University of Oslo
Department of Informatics

Multiple-Input Common-Gate FGUVMOS Transistor and Its Application in Multiple-Valued Logic Circuits

Øyvind Hagen

Cand. Scient. Thesis

May 2006

Acknowledgments

This thesis brings to an end my work for the Cand. Scient. degree at the Department of Informatics, University of Oslo.

I would like to thank my advisor Yngvar Berg for accepting me as his student and for his relentless guidance to the field of active floating-gate circuits.

I would also like to thank Dag T. Wisland for his help with circuit simulations, chip layout and measurement equipment. A thanks to Espen Torstensen for building the test circuit board is also in order. The experiences shared by my fellow graduate students, especially Rene Jensen and Johannes G. Lomsdalen, was invaluable for completing this thesis.

And at last, but not least, I would like to thank my family for their unwavering support.

Department of Informatics, University of Oslo, May 2006.
Øyvind Hagen

Abstract

The demand for reduced area and power consumption have usually been met with improvements in processing techniques, allowing for increased integration and a reduction in the power supply voltage. Some technology improvements have also occurred, such as strained silicon and silicon-on-insulator. But some design techniques also feature a significant reduction in area and power consumption, such the asynchronous design approach. Reducing the amount of interconnects is another approach, for which multiple-valued logic might be an ideal candidate.

This thesis explores the multiple-input common-gate FGUVMOS transistor and the design of multiple-valued logic circuits using this transistor. We examine in detail a UV-programming technique for initializing the floating-gate. There is no need for any extra programming circuitry with this programming method, since it utilizes the supply rail of the nMOS transistor to place a charge on the floating-gate. An important benefit of the floating-gate initialization is a matching of the pMOS and nMOS transistor at a predetermined current level. We also look closer at some of the layout issues concerning FGUVMOS circuits.

We also explore a new area of application for the FGUVMOS transistor, namely multiple-valued logic. The main design parameter of the FGUVMOS transistor–the capacitive division ratios of the coupling capacitors to the floating-gate–is well suited for designing voltage-mode multiple-valued logic circuits. Several multiple-valued logic circuits are examined in detail and several design issues are addressed. Measurements on a fabricated chip are supplied, as well as simulations of the various circuits. And the voltage output functions for the presented circuits are also developed.

Contents

Chapter 1

Introduction

The demand for increased integration–especially in the area of mobile communication and hand-held computing devices–necessitates more computational power per area in order to either reduce the size of the devices or to achieve more functionality. There also exists a need for a reduction in the power consumption to prolong reliable power supply and to ease the cooling requirements, since an increase in device density leads to a higher heat density.

The conventional solutions to these problems have come from various refinements of the processing techniques for integrated circuits. Microelectronic devices have been scaled down mainly through improvements in lithography. The reduction in power supply voltage is merely an obligatory side effect of this downscaling in order to avoid breakdown in the devices. There are, however, some approaches to achieve these goals that do not rely on the refinement of processing techniques.

There have been some improvements in the technology used for fabricating the integrated circuits. Two notable improvements have been strained silicon and SOI (silicon-on-insulator)[1]. Strained silicon enhances the electron/hole mobility by straining the silicon lattice, while SOI reduces the parasitic capacitances present in the devices by adding an insulator on top of the substrate. This combination not only yields an increase in speed, but also reduces the power consumption of the devices.

But there exists solutions to the problem of power consumption that can be found in the design step. Various low-power design methodologies have been able to reduce the power consumption in logic circuit systems beyond what is possible with the traditional static CMOS designs, although static CMOS is a very power efficient technology. Most of the extraneous energy consumption from static CMOS comes from the charging and discharging of the gate and diffusion capacitance due to spurious transitions on the input. However, as mentioned above, the reduction of the power supply and transistor sizes–thereby shrinking the gate and diffusion capacitances–is what affects the overall power consumption the most.

One way of reducing the power supply voltage–which will result in a cubic reduction in power consumption[2]–is through the employment of floating-gate transistors. With such a solution one can lower the voltage supply even further than normally possible with static CMOS. This can be achieved by placing a charge on the floating-gate, which effectively shifts the threshold voltage of the transistors[3]. Ultra low-power applications ($V_{dd} < 1V$) can be achieved through this method. An alternative way of shifting the threshold voltage is through back-gating, i.e. by applying a voltage to the back-gate of the transistor.

While there exists various methods for decreasing the power consumption due to spurious transitions, most of these methods relies on a precharging phase which increases the activity of the devices, thereby consuming more power overall. Another method, which also reduces the overall power consumption, is asynchronous circuit design[4]. Asynchronous circuits have successfully been used to reduce power consumption in commercial integrated circuits[5]. A large part of the reduction in power consumption comes from a reduction in the amount of interconnects used for clock distribution. The resistance and the capacitance of the clock distribution network–coupled with frequent charging and discharging–has made the clock network an ample target for those who seek to reduce power consumption in integrated circuits.

However, with all the methods above for reducing the power consumption, the limited information conveyed by the binary logic system still remains. A large amount of the die area is still being used for interconnects between circuit elements and modules. While an analog electronic system could theoretically have an infinite information density, there might be a problem with reliably detecting and retrieving the information. The advantage of binary logic coupled with two-state logic devices, such as the MOSFET transistor, over an analog electronic system when it comes to robustness is evident, but a middle-ground might be desirable. This middle-ground is provided by MVL (multiple-valued logic), where the information density is larger than for binary logic, while it is still possible to maintain a reasonable level of robustness.

1.1 Multiple-Valued Logic

The main issue with interconnects in modern digital systems is well known. The problem is usually described as twofold with the first problem being the limited number of edge connections. The space allotted for edge connections grows only linearly with length n, while the general die area grows as n^2[6]. This has led to several different package types to accommodate the growing need for more pins[7]. Parallel communication leads to a higher pin count and also suffers the problem of multiple delay paths. These delay paths have to be synchronized in order to ensure the integrity of the overall signal. This is usually achieved by decreasing the speed of all the delay paths to accommodate the slowest one. These problems can be avoided by either serializing the communication or by using MV (multiple-valued) signals[1].

The second problem is the ratio of interconnects to active circuit area on the die. The limited information that is conveyed by a two-level logic system means that a large amount of the die area–one interconnect for each bit that makes up the signal–is used for interconnects, since a larger number of devices must be used to realize complex functions than for MVL (multiple-valued logic) systems[2]. Earlier observations suggest that of a VLSI die area, approximately seventy percent is devoted to interconnects, twenty percent to spacing consisting of insulation and only around ten percent for the actual devices[8]. These observations has led to more compact designs for interconnects such as buses, and increased self-sufficient modularity[6] in an effort to reduce intermodule communication. However, the number of interconnects within these modules increases as a consequence and the problem still remains, although at a different level.

By using multiple-valued signals on the interconnects, a reduction in the routing cost may follow, since interconnects will carry more information without an increase in area cost. Despite this obvious solution to the interconnect problem, multiple-valued logic has not gained wide acceptance. The reason for this mostly stems from the fact that there exists no integrated multi-state device. Efficient and robust two-state devices is the main reason for the prevalence of binary logic.

There are many different methods for designing MVL circuits. The method presented in this thesis uses a floating-gate to construct the MVL circuits. The floating-gate has certain advantageous characteristics with regards to MV signals. With these advantages it might be possible to reduce area and power consumption for the computing devices by using floating-gates to construct the MVL circuits.

[1]The use of MV signals for off-chip communication is not advantageous. The MV signals are usually converted into binary signals before being sent off-chip.

[2]Se section 2.1 for an elaboration on the number of logic functions in a digital system.

1.2 Floating-Gate Circuits

Floating-gate devices have mainly been deployed as memory elements for structures such as EPROM, EEPROM and flash memory[9, 10]. It was not until Shibata and Ohmi introduced the MIFG (multiple-input floating-gate) transistor[11], that circuits using floating-gates as an active circuit element started emerging.

These active floating-gate circuits make use of MIFG transistors as their active devices. The input signals of the MIFG transistor are not coupled directly to the gate, which is the standard configuration for MOS transistors. Instead, a coupling capacitor is used. This configuration allows for several inputs, each coupled to the gate of the MIFG transistor by a separate coupling capacitor.

The signal from each of the coupled inputs is attenuated by the ratio of the coupling capacitor to the sum of all the coupling capacitors[3]. This capacitive division relationship between the various inputs is an important design parameter for MIFG transistors, especially since it is possible to form highly accurate capacitor ratios in integrated circuits[12].

It is also possible to place a permanent charge on the gate[4] using several different programming methods, due to the fact that the gate is isolated. The most common programming method[5] utilized is a combination of hot-electron injection and Fowler-Nordheim tunneling. Another common method for placing a charge on the floating-gate is through UV-light programming. A variant of the latter method is used in this thesis.

1.3 Overview of the Thesis

The main objective of this thesis is to introduce a new method for initializing a FGUVMOS (floating-gate ultra-violet MOS) transistor and demonstrate its use as a MIFG transistor by using it to design MVL circuits.

The thesis is divided into the following sections:

- **Chapter 1** contains an introduction to the motivation for the thesis. It examines the various methods for reducing area and power consumption. It

[3]Note that there are several other capacitances to consider, such as various parasitic capacitances arising from the MOS transistor itself. Se section 3.2 for details.

[4]There will always be some leakage through the gate oxide, but for modern processes in the lower submicron and nanometer range, the leakage will be significantly larger. This is due to the downscaling of the gate oxide along with the rest of the device geometry[13].

[5]This programming method is used in devices such as EEPROM and flash memory.

also has an introduction to multiple-valued logic and floating-gate circuits.

- **Chapter 2** presents the concepts behind multiple-valued logic. An overview of the various design methods used in the construction of multiple-valued logic circuits is presented. Various aspects of multiple-valued logic algebra is examined and the cost and complexity of multiple-valued logic systems are discussed. Different signal representation are examined in more detail and the advantages of MIFG MVL is addressed. The importance of signal integrity is also pointed out.
- **Chapter 3** has floating-gate devices as its topic. The development toward active floating-gate circuits is presented. The FGUVMOS transistor is introduced, its principal design parameter is examined and its equations are shown. The programming technique is presented and implementation considerations are explored.
- **Chapter 4** presents various multiple-valued circuits using FGUVMOS transistors. The basic building blocks for multiple-valued logic circuits are introduced. The various multiple-valued logic circuits with their descriptions and equations are presented. Simulations for the circuits and measurements on a fabricated chip are given.
- **Chapter 5** summarizes the thesis as a whole and gives pointers to where further research on the topics presented in this thesis should the focused.

Chapter 2

Multiple-Valued Logic

2.1 Introduction

Much as the decimal system has dominated our understanding of arithmetic, so has the binary system dominated our understanding of logic. This connection is so strong in fact, that the term logic implies binary logic, even though binary logic is merely a subset of multiple-valued logic[1]. When arithmetic operations on the decimal system are to be performed, one often makes use of binary logic. A radix ten logic system would be more appropriate, since the values would map directly and no information would be discarded, as is the case with binary-coded decimals. The prevalence of binary logic in digital circuits today, stems mainly from the availability of two-state physical devices.

Multiple-valued logic is by no means a new construct. Its origin dates back to the novel work of Post in 1921[14], who published the first paper detailing a functionally complete algebra for any finite radix n, where $n \geq 2$. And there existed practical implementations of multiple-valued logic systems before the solid-state devices entered the arena. Back then, the electromagnetic relays were the principle component within switching systems[15]. Although the simplest ones were binary, there existed multi-state switches. As a result, a wide range of applications with multi-state devices were in use. But with the advent of solid-state devices, switching systems with more than two states were largely lost due to efficient and robust two-state devices, such as the MOSFET transistor.

The latter-day developments makes use of either binary or analog circuits–or a

[1]There are many additional names used to describe multiple-valued logic, such as multi-valued, multivalue and many-valued logic. In the case of radix three logic, the term ternary or trinary logic is preferred.

combination thereof–to construct multiple-valued logic, since a true multi-state device seems elusive. The earlier multiple-valued logic circuits were implemented using discrete BJT or MOS transistors[15]. The implementations could be divided into two broad categories. One was the current-mode approach where the current levels where divided into discrete steps. The other category was the voltage-mode approach, which made use of several different power supply rails. Both design approaches had in common a heavy reliance on resistors, and were therefore not suited for implementation in integrated circuits.

Integrated multiple-valued logic realizations generally favor three commonly used design approaches, closely connected to the signal representation used[6]. Charge used in charge-coupled devices[16] in one of the common design approaches, although voltage is used both as an internal variable and for the external interface. Current used in integrated injection logic[17] is another approach, but again voltage is used both as an internal variable and for the external interface. There also exists a design approach using the voltage as the signal representation. These are usually ternary logic circuits exploiting the difference between threshold voltages in enhancement and depletion devices[18].

2.2 Algebraic Notation

The radix signifies the number of logic levels in a multiple-valued logic system. A higher radix gives you more computational complexity, i.e. the possibility to form a higher number of different logic functions.

Most multiple-valued logic systems are formed from logic values that are a continuous monotonic set of integers. Extending the binary notation–where the set of logic values is given as $\{0,1\}$–in the positive direction is one method of forming a continuous monotonic set of integers. The set of logic values resulting from such an extension is given as $\{0,1,...,r-1\}$ where r is the radix, and is called an unbalanced system (or unsigned system). Unbalanced systems has an even radix, $r = 2n;\ n \in \mathbb{Z}^+$. Balanced systems are formed when the radix is odd, $r = 2n-1;\ n \in \mathbb{Z}^+$. Here the value set is usually given as $\{-\frac{r-1}{2}, -\frac{r-3}{2}, ..., 0, ..., \frac{r-3}{2}, \frac{r-1}{2}\}$, which includes negative integers. The ternary logic value set would then be given as $\{-1,0,1\}$.

For binary logic, the inverted of a logic state is well defined and no ambiguity exists, since each state has the other as its inverted. However, for the superset of MVL, this is not possible. In logic it is normal to define inversion as a reversible state transformation, where a state that is transformed two consecutive times is returned to its original state. Such a unary negation operator for multiple-valued logic can be given as $\overline{x} = (r-1) - x$ where r is the radix[15]. For continuously monotonic balanced systems, this means that the inverted of the pivot point, given as $x = \frac{r-1}{2}$; $\overline{x} = (r-1) - \frac{r-1}{2} = \frac{r-1}{2}$, is in fact itself. The pivot point for a continuously monotonic unbalanced system is not a defined logic state, and consequently no state has itself as the inverted.

The MVL system presented in this thesis has a continuous monotonic and unbalanced logic value set. The logic value set is merely an extension of the binary one

$$\{0, 1, ..., r-1\} \tag{2.1}$$

where r is the radix.

2.3 Radix and Complexity

A general two-input, called A and B, one-output, called F, digital system can form $\max(F)^{(\max(A)\cdot\max(B))}$ different logic functions, assuming the inputs are not correlated. For a binary signal, this amounts to $2^{2*2} = 16$ possible logic functions. With multiple-valued logic, that number increases greatly. For a multiple-valued logic system of radix three, $3^{3*3} = 19683$ possible logic functions can be formed.

We can generalize this for an arbitrary number of inputs, each with their own radix. The requirement of one output is retained. We have n inputs, given as the sequence $\{0, ..., i, ..., n-1\}$. The input radices are given as the sequence $\{r_0, ..., r_i, ..., r_{n-1}\}$ with the output radix given as r_{out}. Each input can take on r_i different values and we assume the inputs are uncorrelated. For each possible combination of input values, the single output can take on r_{out} different output values. The number of possible logic functions in a mixed radix multiple-valued logic system is then given as $r_{out}^{\prod_{i=0}^{n-1} r_i}$.

With so many radices to choose from, one wonders if there are any radices that have a special position. The computational complexity of an MVL system arising from higher radices almost certainly does not come without a cost. This naturally raises the question of which–if any–radix is the optimal. For many logic systems, the radix is given as an inherent characteristic. However, the inherent radix might not be optimal for reducing costs. Hurst has suggested that the cost (or complexity), C, of the system hardware[15] is given as

$$C = k\,(r \cdot d) = k\left[r\frac{\ln N}{\ln r}\right] \tag{2.2}$$

where k is some constant, r is the radix and d is the number of digits in a numerical system necessary to express a range of N numbers, where $N = r^d$. By considering the radix as a real variable and the cost function, C, as a continuously differentiable function, we can take the derivative of (2.2) with respect to the radix

$$\frac{\partial C}{\partial r} = \frac{\partial}{\partial r} k \left[r \frac{\ln N}{\ln r} \right] = k \ln N \frac{\frac{\partial}{\partial r} r \cdot \ln r - \frac{\partial}{\partial r} \ln r \cdot r}{(\ln r)^2} = k \ln N \frac{\ln r - 1}{\ln^2 r}$$

from which we need to find the zeros

$$\frac{\partial C}{\partial r} = 0 \Leftrightarrow k \ln N \frac{\ln r - 1}{\ln^2 r} = 0$$
$$\ln r - 1 = 0 \Leftrightarrow r = e \approx 3$$

and the result indicates an optimal point exists where the cost of increasing the radix is equal to increasing the number of digits, and that ternary logic is the theoretically optimal radix for a logic system with an increasing cost for an increasing radix. The cost function given in (2.2) only holds for the circuits in this thesis that converts between binary and multiple-valued signals. Increasing the radix usually means that another capacitor or transistor has to be added to the circuit. The same applies for increasing the amount of digits.

For the circuits that only process multiple-valued signals, however, the cost of increasing the radix usually means that only a small adjustment has to be made to the capacitor relationships[2]. For this case, where the cost does not increase proportionally with the radix, Hurst has suggested an alternative cost function[15]

$$C = kd = k \left[\frac{\ln N}{\ln r} \right] \tag{2.3}$$

where the cost is decreasing with an increasing radix. The optimal system cost would then have to take into account the ratio of converter circuits to circuits only processing multiple-valued signals.

The power-of-two radices also have a special position in MVL, especially when interacting with binary logic. The power-of-two radices are defined as

$$r = 2^n; \quad n \in \mathbb{Z}^+ \tag{2.4}$$

They are optimal for interacting with binary logic systems since no information is discarded in the conversation between binary and multiple-valued signals. This also means that if the radix in a conversion circuit is between the steps defined by equation (2.4), then there is no cost associated with increasing the radix to the nearest step.

[2]Increasing the number of digits, however, has a cost similar to the conversion circuits.

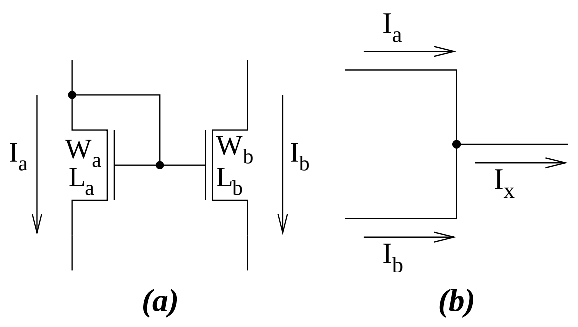


Figure 2.1: ***(a)*** *Scaling can be done by using a current mirror in the current-mode approach. The lengths, L_a and L_b, are equal. The widths, W_a and W_b, are different with the scale factor given as W_b/W_a and $I_b \approx I_a \cdot W_b/W_a$.* ***(b)*** *Addition in the current-mode approach is done by simply connecting the wires together. Here the output current is given as $I_x = I_a + I_b$.*

2.4 Signal Representation

There are many ways to represent digital signals, for instance as voltage, current, charge or frequency. Although with binary logic, voltage is usually chosen as the signal representation, most likely due to the availability of voltage controlled two-state devices, e.g. the MOSFET transistor. However, the choice of signal representation is not so clear when it comes to MVL, since there is no readily available multi-state device. Each of the different signal representations have certain advantages and disadvantages with regards to MVL.

One technique regarded as promising for designing MVL circuits is the CMOS current-mode approach[19]. Here the logic levels are defined as multiples of a base current. Some operations, such as scaling, can be preformed by using a current mirror, as depicted in Figure 2.1. Other operations, such as addition, can be performed even simpler, by just connecting the wires together. The cost effectiveness of several operations in the current-mode approach can contribute to alleviating the areas cost for this design approach.

There are, however, several disadvantages to the current-mode approach[19, 15]. While static CMOS binary circuits have extremely low power dissipation in both of the stable states, the current-mode approach has a rail-to-rail current flow in all states and thus a high amount of static power dissipation. This is due to the biasing of the current mirrors used in the circuits. Since there exists no multi-state devices in the current-mode approach either, there exists a profound problem of signal integrity. Another important disadvantage with the current-mode approach is that the circuit delay is directly proportional to the logic level. The base current–which is usually also the lowest logic level–sets the maximum delay path. The higher logic levels therefore draw a larger amount of current compared to the lower logic levels–increasing the overall power consumption–without providing an overall speed increase. These points leads to inefficiency in area and power dissipation, despite the inherent advantages mention above.

The voltage-mode approach in CMOS MVL have usually consisted in exploiting the difference between voltage thresholds for the depletion and enhancement devices. However, this approach is only suited for low radix operations. By exploiting the capacitive division relationship inherent in MIFG (multiple-input floating-gate) transistors[11], we achieve several advantages. We can match the transistors by placing a charge on the floating-gate, meaning minimum device sizes can be used for both the nMOS and pMOS transistor. Fewer transistors are used, since most of the circuit complexity is place inside the devices themselves. Low-voltage operation is also possible due to the matching of the transistors, however, low noise margins might then become a problem for robust circuit operations. The radix can be of an arbitrary size, but again, low noise margins pose a problem with an increase in the radix.

Unfortunately, also MIFG MVL uses the quantization of an analog signal carrier–voltage in this case. This means that there is a rail-to-rail current flow for all logic states. There are also no stable logic states, meaning there is no inherent signal restoration. Like with the current-mode approach, signal integrity is a prominent issue.

2.5 Signal Integrity

Signal integrity is an important aspect in any electronic system. If the signal information in a circuit can not be reliably retrieved, then the system is of no practical use. There currently exists no readily available multi-state physical device for use in electronic systems. This means that most MVL systems rely on the quantization of an analog signal carrier to represent the MV signal. This gives the system greater sensitivity to various noise sources, such as crosstalk arising from coupling to nearby interconnects or power rail spikes arising from switching circuits that draw large amounts of current. Process variations might also lead to unwanted shifts in the logic levels due to mismatch in the physical devices, such as variations in capacitor sizes.

The single most important task a two-state device performs for binary logic is signal restoration. Every physical device in a binary logic system restores the signal to a pristine state, making noise margins large and increasing system robustness. This is not the case with MVL. Since there is no multi-state device, there is also no automatic signal restoration. A signal restoration circuit will have to be used instead for cascaded systems. The signal restoration circuit should only be used at periodic intervals, dependent upon such characteristics as process variation, noise and radix, due to the area and power consumption cost such a circuit would incur.

2.6 Summary

This chapter has presented an historic overview of the development of MVL circuits. Multiple-valued electromagnetic switches were in use before solid-state devices came about. The methods used for discrete components was precluded from being used for integrated circuits due to a heavy reliance on resistors. We were then presented with the algebraic notation for MVL and the difference between balanced and unbalanced systems was explained. The next topic was the radix and complexity of MVL systems, where two different cost functions were presented. And the number of logic functions possible with a multiple-valued logic system was found. The special power-of-two radices were also discussed. We then looked more in detail at some of the various methods for constructing integrated MVL circuits. The advantages and disadvantages of MIFG MVL was also explored. Lastly, the importance of signal integrity–and the role played by signal restoration devices–was pointed out.

Chapter 3

Floating-Gate UVMOS Devices

3.1 Introduction

Since the first floating-gate structure was reported by Kahng and Sze in 1967[20], the FGMOS (floating-gate MOS) transistor has mainly been used as a non-volatile memory device for digital applications. Various implementations of non-volatile memory devices have evolved, such as EPROM, EEPROM and flash memory[9, 10]. These non-volatile memory devices are normally only available in specialized processes.

The FGMOS transistor has also been used as a non-volatile storage element in analog applications. The charge on the floating-gate has been used to increase accuracy and matching of circuit parameters[21]. It has also been used in neuromorphic circuits and neural-network implementations as a weighted parameter in learning algorithms[22, 23].

The development toward active floating-gate circuits came from computational methods in neuromorphic systems[24]. While the ETANN chip[23] still used the floating-gates for analog storage, the output current from these devices were used by multiplier circuits. Carver Mead presented an adaptive retina circuit, which was the first example of a continuously reconfiguring circuits using FGMOS[25]. Thomsen and Brooke demonstrated the use of electron tunneling in a standard double polysilicon process[26], allowing for experimentation with floating-gate structures in more accessible processes.

An EEPROM addressing structure, called a dual control-gate, was introduced by Heida et. al.[27]. It had two control-gates that were capacitively coupled into a floating-gate, and the capacitors were of equal size for each of the two control-gates. It had one control-gate for the row and another for the column, to address a single cell. When the cell received a signal on both of the control-gates, it would initiate a write cycle using Fowler-Nordheim tunneling.

As a generalization of the dual control-gate EEPROM, Shibata and Ohmi[11] introduced the νMOS (neuMOS, also called neuron-MOS) transistor for use as an active circuit element. The name came from a loose analogy to how synapses in the nervous system works. This was the first time a floating-gate structure had been used as an active circuit element.

Berg et. al. set forth a new programming technique for floating-gate transistors[3]. This method made use of UV-light to set and remove charge from the floating-gate. The programming technique ensured matching of the transistors by placing a charge on the floating-gate. This charge effectively shifted the threshold voltage, allowing for ultra low-power applications.

3.2 Floating-Gate UVMOS Transistor

The FGUVMOS (floating-gate ultra-violet metal-oxide semiconductor) transistor has a structure similar to the νMOS transistor. The main difference lies in the programming technique, which will be discussed later on in section 3.5. The FGUVMOS transistor, shown in Figure 3.1 (a), has multiple inputs, each capacitively coupled to the floating gate. These voltage inputs determine, through capacitive voltage division, the floating-gate potential. In turn, the floating-gate potential modulates the current in the channel. This adds more complexity–both in design and functionality–to the transistors, from which we can hope to achieve overall simpler circuits that will consume less area and power. The capacitive voltage division is the essential operating parameter of the FGUVMOS transistor in relation to its use in multiple-valued logic. The pMOS transistor is not shown, since only the nMOS transistor is different from the standard MOSFET device with the programming method used in this thesis. We can see a basic layout of a FGUVMOS transistor in Figure 3.1 (b).

The process used for the circuits in this thesis is the AMS 0.6μ CUX CMOS process with three metal layers and two polysilicon layers. Any double polysilicon layer process will be adequate, however, most modern submicron and nanometer processes will have excessive gate leakage due to the thinning of the gate oxide, and are therefore not suitable for constructing non-volatile floating-gates[13].

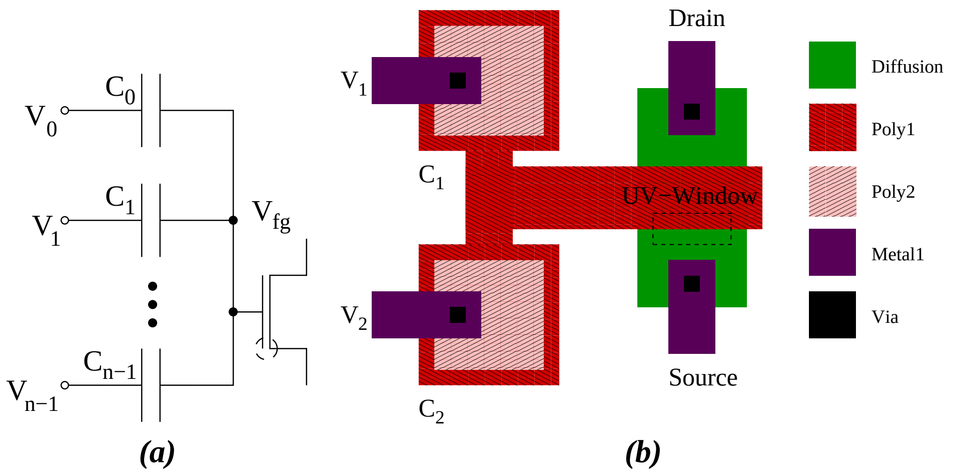


Figure 3.1: ***(a)*** *FGUVMOS nMOS transistor symbol. The symbol shows several input signals capacitively coupled into the floating-gate. The circle enclosing the floating-gate and the source terminal is symbolizing the UV-hole used in the programming of the transistor. The pMOS transistor is not shown since it is not involved in the UV-programming.* ***(b)*** *FGUVMOS transistor layout. The UV-hole encompasses the source diffusion and the polysilicon gate. The coupling capacitors consists of stacked polysilicon layers, forming poly-poly capacitors. The routing between the floating gates and the capacitors are done in the lower polysilicon layer (poly1, which is also used for the gate of the transistor). The upper polysilicon layer is connected to the input node through the metal1 layer.*

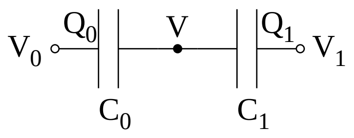


Figure 3.2: *Capacitive division relationship. The floating node, V, has two capacitively coupled, through C_0 and C_1, voltage inputs, V_0 and V_1. The inputs have the two related charges, Q_0 and Q_1*

3.3 Capacitive Voltage Division

To fully understand the operating principle of the FGUVMOS transistor, we will first have to examine the capacitive division relationships employed by it, and how the floating-gate potential is modulated by the capacitively coupled voltage inputs.

In Figure 3.2, we can see a simple setup to determine the floating potential V. We have the relationship $Q = CV$ and assuming that no net charge is stored on the

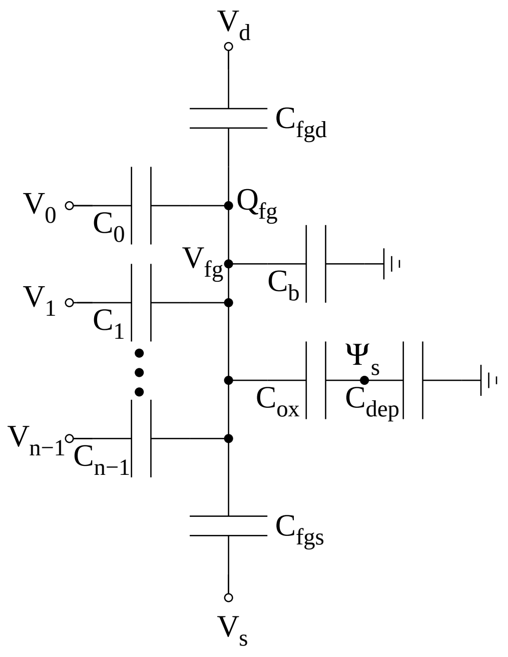


Figure 3.3: *Equivalent circuit used for deriving the capacitive division relationships for the FGUVMOS transistor. All the parasitic capacitances are taken into consideration. Also the capacitively coupled channel surface potential feedback to the floating gate is addressed. Only the nMOS transistor is shown. For the pMOS transistor (given a p-type substrate with a n-well), a charge on the bulk terminal may also have to be taken into consideration, if back-gate modulation techniques are used to fine-tune the device. This applies to the nMOS transistor as well when using a process that employs wells for all diffusions, such as a SOI process.*

node V, we get

$$\begin{aligned} -Q_0 - Q_1 &= 0 \\ -C_0(V_0 - V) - C_1(V_1 - V) &= 0 \\ V &= \frac{C_0}{C_0 + C_1}V_0 + \frac{C_1}{C_0 + C_1}V_1 \end{aligned}$$

due to the law of conservation.

We can translate this to the FGMOS transistor by generalizing for a finite amount of capacitively coupled voltage inputs. The floating node will be the floating-gate, V_{fg}, and the parasitic capacitances are taken into consideration. We then get the setup shown in Figure 3.3[11].

If we in addition assume that a charge is present at the node V_{fg}, we have the charge on the floating-gate, Q_{fg}, given as

$$Q_{fg} = C_{ox}(V_{fg} - \Psi_s) + C_b V_{fg} - \sum_{i=0}^{n-1} C_i(V_i - V_{fg}) \\ - C_{fgs}(V_s - V_{fg}) - C_{fgd}(V_d - V_{fg}) \tag{3.1}$$

where Ψ_s is the channel surface potential. We then rearrange to get the floating-gate potential

$$V_{fg} = \frac{C_{ox}}{C_T^*}\Psi_s + \frac{Q_{fg}}{C_T^*} + \sum_{i=0}^{n-1} \frac{C_i}{C_T^*} V_i + \frac{C_{fgs}}{C_T^*} V_s + \frac{C_{fgd}}{C_T^*} V_d \tag{3.2}$$

where $C_T^* = C_{prc} + C_T$, with $C_T = \sum_{i=0}^{n-1} C_i$ and $C_{prc} = C_{ox} + C_b + C_{fgs} + C_{fgd}$. Most of these parasitic capacitances can be ignored for hand-calculations–which we will do for all the calculation in this thesis. But the gate capacitance might effect the size of any capacitor shunting the output and the floating-gate. Specifically, the size of this shunting capacitor would have to be reduced to account for the gate capacitance.

3.4 FGUVMOS Transistor Equations

The FGUVMOS transistors are programmed in an inverter configuration to satisfy an output of $V_{out} = V_{dd}/2$ when the input is $V_{in} = V_{dd}/2$ in order to ensure a symmetrical equilibrium point, and thus a matching of the pMOS and NMOS transistor. The current at the equilibrium point[1]–which is determined by the input voltage during the programming of the circuit–is given as I_{bec}.

The equations for the weak inversion region are chosen as a matter of convenience, but weak inversion is not a prerequisite for circuit operations. The current equations for the multiple-input FGUVMOS transistor can be expressed as[28, 29, 30]

$$I_{ds,n} = I_{bec} \prod_{i=0}^{n-1} exp\left(\frac{1}{\eta U_T}(V_i - V_{dd}/2)k_i\right) \tag{3.3}$$

$$I_{ds,p} = I_{bec} \prod_{i=0}^{n-1} exp\left(\frac{1}{\eta U_T}(V_{dd}/2 - V_i)k_i\right) \tag{3.4}$$

[1]It is natural to connect the equilibrium point to the pivot point, which was discussed in section 2.2.

where $U_T = \frac{kT}{q}$ is the thermally exited voltage where k is Boltzmann's constant, q is the electron charge and T is the temperature in Kelvin, η is the slope factor of the transistor and $k_i = \frac{C_i}{C_T}$ with $C_T = \sum_{i=0}^{n-1} C_i$. η is usually between 1 and 2, depending on the current level, while $U_T = 25mV$ at room temperature $(300K)$. It is important to note that these equations are simplifications that do not include the effect of channel shortening or velocity saturation, and should therefore only be used in connection with long channel devices. It should also be noted that the floating-gate voltage will always remain at the programmed equilibrium point. The equations above merely describe voltage differences on the coupling capacitors that modulate the equilibrium current.

3.5 Programming Technique

An FGUVMOS circuit always comes with a stacked height of two. One pMOS transistor stacked on one nMOS transistor. This leaves us with an inverter as the fundamental circuit topology, although one might also use several nMOS or pMOS transistors in parallel. Although this might seem as a restrictive design limitation, complex circuit functions can still be achieved through the capacitive division relationships on the inputs. In addition, the capacitive division relationships design methodology lends itself as a natural choice for certain circuits such as multiple-valued logic circuits.

Fowler-Nordheim tunneling (electron tunneling) and hot electrons (electron injection) are traditionally used to remove and add charge to the floating-gate[31], respectively. The UV-programming method instead uses shortwave UV (ultraviolet) light in the UV-C range ($254nm$ to be exact). This wave length is in common use in UV-erasers/programmers, normally used for programming or erasing ROMs and EPROMs. When the source and gate region is exposed to UV-light, a UV-activated conductance is created across the separating oxide. UV-activated conductances is a convenient way to simplify the underlying model.

It should also be noted that this programming method ensures calibration of the floating-gate[32]. The floating-gate is set to a known quantity. This is different from the combination of tunneling and injection, which cannot set the floating-gate to a known quantity, thus requiring UV exposure to calibrate.

The earlier UV-programming method[33] of the FGUVMOS transistors used a split-gate configuration, as seen in Figure 3.4. This allowed for separate tuning of the current-level since the pMOS and the nMOS transistor could have a different charge on the floating-gate. This technique was a good match for ultra low-power applications since the threshold could be shifted. However, there were problems

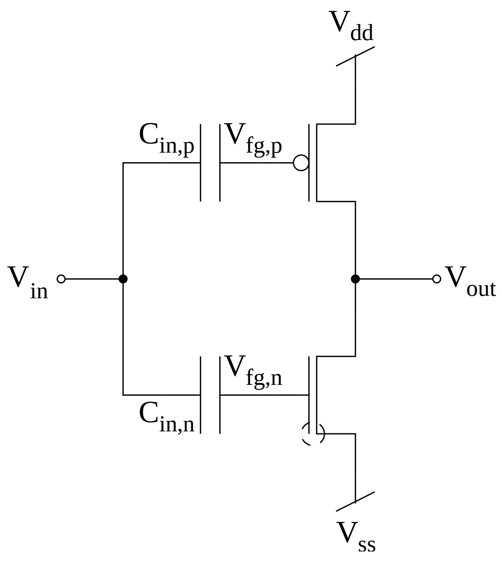


Figure 3.4: *A split-gate FGUVMOS inverter. The split-gate inverter has a separate floating-gate for the nMOS and pMOS transistor. This allows for different charges to be placed on the floating-gates and the threshold can be effectively shifted. It also allows for the current level to be tuned by setting different charges on the floating-gates.*

with the programming of the pMOS transistor due to its weaker workfunction, resulting in a large workfunction difference between the nMOS and the pMOS transistor. By instead using a common-gate configuration–where the pMOS and nMOS transistor share the floating-gate–these problems are solved. However, one ends up with one less tuning parameter, and therefore the current level cannot be arbitrarily set, like in the case with a split-gate configuration. This makes the common-gate topology less useful for ultra low-power applications, but requires less area to implement. It is also easier to program since the programming step only involves the nMOS transistor.

A normal biased common-gate FGUVMOS inverter circuit is shown in Figure 3.5 (a). This circuit is in the operational mode. In this mode, there is no conductive connection between the source diffusion and the floating-gate. When UV-light is applied to the circuit, however, an increase in conductivity occurs. This increase in conductivity is called photoconductivity and is modeled using UV-activated conductances[34], as seen in Figure 3.5 (b). The only desirable UV-activated conductances is the conductive connection between the source diffusion and the floating-gate, $G_{fgs,n}$. While metal shielding is in place to ensure illumination of only the desired area, some UV-light will be reflected under the metal shielding. This stray UV-light gives rise to several unwanted UV-activated conductances.

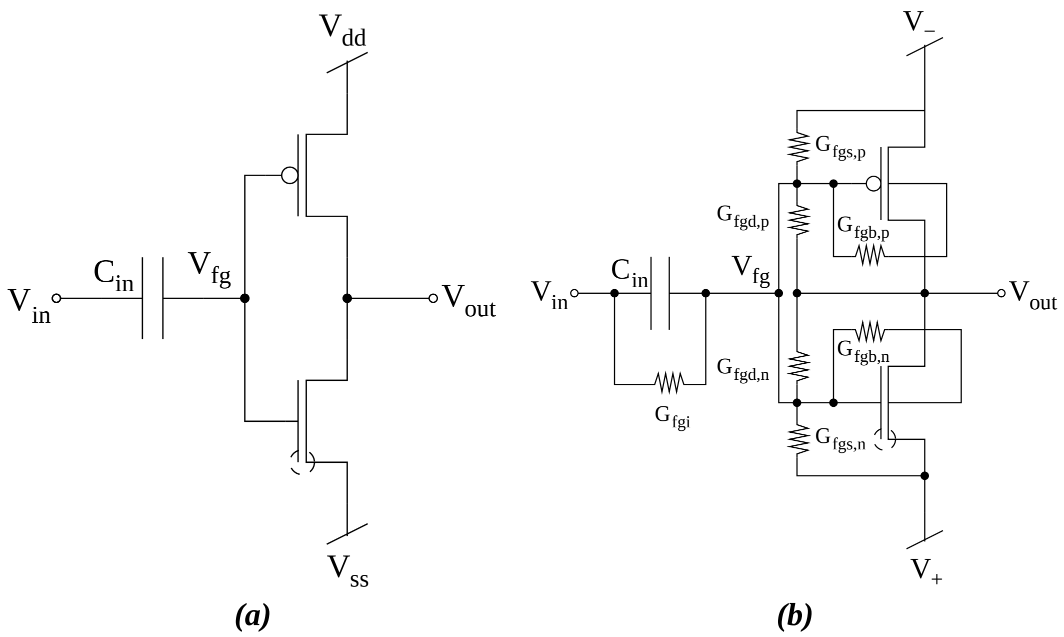


Figure 3.5: ***(a)*** *The operative mode (or normal biased mode) of the common-gate FGUVMOS inverter. Since most of the natural UV-C light is stopped in the outer atmosphere, it is not strictly necessary to shelter the UV-hole during normal operation. However, it is a prudent measure to undertake in case of artificial UV-C sources. In any case, such a protective measure is necessary during the programming of the FGUVMOS circuit.* ***(b)*** *The programming mode of the FGUVMOS circuit. When UV-light is applied, the UV-holes will allow for UV-activated conductances to be "created". Only $G_{fgs,n}$ is the wanted UV-activated conductance. All the other UV-activated conductances are considered parasitic. The parasitic UV-conductance associated with the pMOS transistor for a common-gate configuration will be significantly smaller than for a split-gate configuration. This is because the distance to the UV-hole is larger with the common-gate configuration.*

The energy of the UV-light is given as $E = hv = \frac{hc}{\lambda}$, where h is Planck's constant ($4.14 \cdot 10^{-15} eVs$), v is the frequency of the radiation, c is the speed of light ($3 \cdot 10^{8} m/s$) and λ is the wavelength of the radiation[35]. For UV-C radiation ($\lambda = 254nm$), we then get $E = \frac{hc}{\lambda} = \frac{4.14 \cdot 10^{-15} eVs \cdot 3 \cdot 10^{8}}{254 \cdot 10^{-9} m} \approx 4.9eV$ for the photon energy. In the Si-SiO_2-Si structure making up the connection between the source diffusion and the floating-gate, the SiO_2 presents a $4eV$ barrier for the electrons[34]. We can see from the above calculations that UV-C radiation imparts enough energy to allow the electrons to surmount that barrier. The excited electrons in the valence band of the Si layer then enters the conduction band in the SiO_2 layer. The electrons are then swept through the oxide layer by the voltage gradient, as can be seen in Figure 3.6.

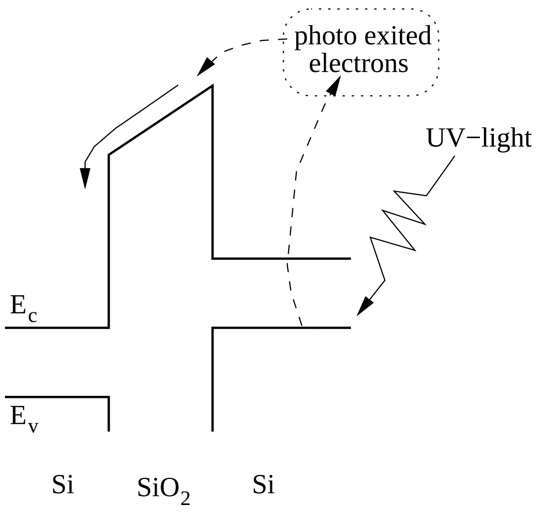


Figure 3.6: *Energy band explanation of the UV-programming procedure. A Si-SiO_2-Si sandwich is shown, as well as the energy levels of the valence band, E_v, and the conductance band, E_c. Energy from the UV-light will excite the electrons in the Si valence band and cause some of them to enter the SiO_2 region. There they will support a current flow through the SiO_2 region due to the voltage gradient.*

The programming procedure consists of applying the desired switching voltage on the inputs and a higher potential on V_{ss} than on V_{dd}. This leads to the source and drain changing place on the transistors, thus giving us a low output impedance. Unlike most programming methods, this technique does not require any programming circuitry since the floating-gates are programming from the V_{ss} rail.

The programming technique entails the following steps:

1. **Decide upon the supply voltage, V_{dd}.** This is the supply voltage that will be used in the normal operating mode. It will vary between applications.
2. **Apply $V_{dd}/2$ to all external inputs.** When the programming is over, all internal and external nodes will have reached $V_{dd}/2$.
3. **Apply the programming voltages, V_- at V_{dd} and V_+ at V_{ss}, to the supply rails.** Instead of using extra programming circuitry, which would be used when programming from the gate, we use the supply rails instead. This gives reduced area overhead in the circuit designs.
4. **Apply the UV-light.** The nMOS transistor has a UV-hole to allow the UV-light to excite electrons in the Si valence band to the SiO_2

conductance band. The applied electric field will sweep the electrons through the oxide layer.

5. **Terminate the programming by removing the UV-light when the outputs converge to $V_{dd}/2$.** The circuit is now ready to be operated in the normal biased mode and perform its desired function.

The programming method ensures matching of the nMOS and pMOS transistors. One benefit of this is that the minimum size for the transistors can be used. There is no need to scale the pMOS transistor to compensate for charge mobility. The transistor must, however, be scaled to achieve the current level desired. The matching of the nMOS and pMOS transistors is achieved by the charge being placed on the floating-gate via the UV-activated conductances. This charge forces an equilibrium point–a voltage point where the nMOS and pMOS currents match–which is decided by the voltage set on the coupling capacitors to the floating-gate. The current level, however, is decided through normal device sizing means.

The programming time for an inverter circuit is normally on the order of minutes to tens of minutes. The programming time can be reduced by increasing the desired UV-activated conductances. This can be achieved by either increasing the luminance of the UV-light source or by moving the light source closer to the desired exposure point, however, an increase in temperature might pose a problem. The UV-activated conductances can also be increased by increasing the source-gate perimeter[2]. Another way to reduce the programming time is by minimizing the parasitic conductances. This can be done through effective shielding. One might expect some undercutting when the hole in the passivation layer is etched. Therefore the hole must be larger than the intended exposure area[3]. Combined with the diffraction of the UV-light from the hole in the passivation layer down to the desired exposure point, there is a need for further shielding of the transistor. This shielding is done in the upper metal layer for reasons of convenience. The shield should cover the entire transistor, with a hole cut out only over the intended exposure area. Common-gate circuits should have an advantage here over split-gate circuits, since there is a larger distance from the exposure point to several of the parasitic UV-activated conductances. This means that the escape routes for the programming currents–which is the role parasitic UV-activated conductances play–should be less effective. It is also important to note that the programming time is independent of the number of circuits, since all of the circuits are programmed simultaneously.

[2]See section 3.6 for a discussion on transistor topologies.

[3]This is especially important if wet etching is used. With dry etching, this might not pose much of a problem.

3.6 Layout Considerations

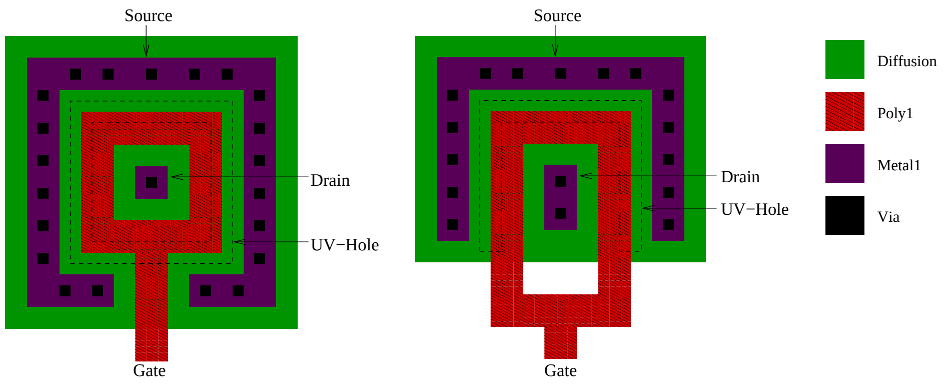
While the desired level of circuit performance is normally attainable during simulation, there are several factors that have to be taken into account to achieve the same level of performance with a processed circuit. These factors are mostly affected by the layout of the individual circuit element and the overall circuit topology. It is important that great care is taken when laying out sensitive analog circuits. The considerations presented here are those that are especially pertinent to–or have the greatest impact on the performance of–the circuits presented in this thesis. Many of the reduced performance characteristics comes from process variations in the basic devices. However, there are ways of minimizing those variations, even to the point of making them a non-issue. There are two main devices for FGUVMOS that has to be considered when laying out the circuits in this thesis, namely the transistors and the capacitors.

3.6.1 Transistors

The programming method used in this thesis, which is described in section 3.5, ensures matching of the stacked pMOS and nMOS transistor. There is no need to take into account the difference between carrier mobility in the nMOS and pMOS transistor, and minimum sizes for the transistors may be used. However, the nMOS transistor is being used in the programming of the circuit. It is therefore prudent to examine how the programming performance can be enhanced by tuning various transistor characteristics.

One important factor is the reduction of the programming time. This reduction can be achieved by increasing the size of the UV-activated conductances. The UV-activated conductance involved in the programming, as can be seen in Figure 3.5, is the source-gate conductance. It is therefore possible to increase the conductance by increasing the source-gate perimeter[34]. There exists several transistor topologies to achieve this goal, such as u-shaped or ring transistors, which can be seen in Figure 3.7.

An noticeably reduction in programming time can be achieved by utilizing one of these transistor topologies. The ring transistor has the largest source-gate perimeter, and therefore takes the shortest time to program. However, the minimum transistor size is larger with a ring transistor due to the larger gate perimeter, thus yielding a larger area consumption. One added benefit with the ring transistor is the greater distance from the UV-hole to the gate-substrate boundary, meaning that the parasitic UV-activated conductance connecting the floating-gate to the



(a) ***(b)***

Figure 3.7: ***(a)*** *Ring transistor layout. The ring transistor gives the largest source-gate perimeter relative to transistor width. It also has the added benefit of the smallest drain capacitance area to width ratio, increasing the speed of circuit operations due to less loading capacitance.* ***(b)*** *U-shaped transistor layout. While the ratios are not as beneficial as for the ring transistor, smaller minimum size transistors can be created with the U-shaped transistor.*

substrate, $G_{fgb,n}$, will be smaller than for a regular or the U-shaped transistor.

3.6.2 Capacitors

Traditional CMOS circuits are concerned with transistor matching and scaling, due to the difference in carrier mobility. FGUVMOS circuits, on the other hand, is more concerned with the matching of capacitors due to the inherent matching of the transistors. It is therefore important that proper consideration is given to the layout of the capacitors in FGUVMOS circuits. The most important consequence for not giving capacitor layout proper consideration is reduced noise margins, leading to problems with reliably detection of logic levels, especially in cascaded systems.

Several possibilities exists for making capacitors in a standard CMOS process[12]. Junction capacitors are made by exploiting a reverse biased junction. MOS capacitors are constructed by connecting the drain and source, using the channel as one electrode and the gate as the other, with the thin-oxide as the dielectric. Poly-poly capacitors are made by stacking two polysilicon layers using the interlayer oxide as the dielectric. The performance of the various capacitors can be characterized by the amount of process variation expected, the voltage coefficient, temperature coefficient and area consumption. The poly-poly capacitors usually have small

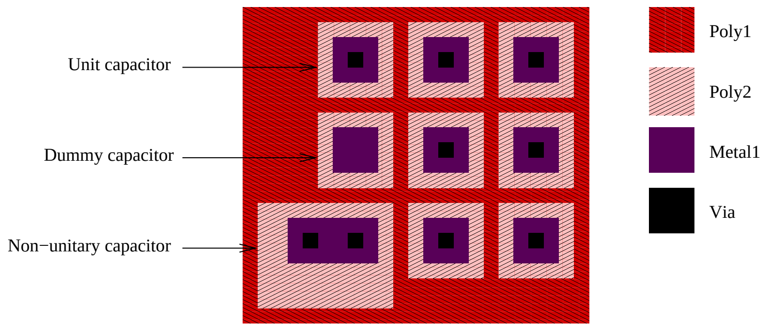


Figure 3.8: *Unit capacitor array. The unit capacitor array consists of unit capacitors in a square (or as near to a square as possible) with dummy capacitors to complete the array. A non-unitary capacitor should be used if the capacitance desired is not an integer multiple of the unit capacitance. A ring of dummy capacitors can be used as shielding against horizontal fringing fields emanating from nearby leads. These shielding capacitors should have the electrodes connected together and they should be grounded.*

process variation and a low voltage coefficient, but sufferers from large area consumption due to the thickness of the interlayer oxide, which is used as the dielectric. Some processes offers other dielectrics with a higher dielectric constant than SiO_2 for poly-poly capacitors, meaning smaller areas can be used to get the same value of capacitance. Both the junction capacitor and the MOS capacitor have a high voltage coefficient, meaning they should only be operated within a small voltage differences from the biasing voltage, if a consistent capacitance is desired. Since rail-to-rail signals can be expected–and a high degree of matching is needed–in FGUVMOS circuits, poly-poly capacitors seems to be the best choice for constructing capacitances.

Polysilicon capacitors usually exhibit a much lower voltage and temperature dependency than alternative capacitances. Integrated capacitors may experience significant process variations leading to tolerances as high as $\pm 20 - 30\%$. However, they can be matched to another capacitor with good accuracy (as high as $\pm 0.1\%$) without using any trimming, provided that some guidelines are followed[12, 36].

To achieve good matching when laying out capacitors, one should always try to make use of arrays of unit capacitors, as can be seen in Figure 3.8. A unit capacitor should be decided upon, with all other capacitors being a multiple of that size. Capacitors with a size that is not a multiple of the unit capacitor size needs to add a non-unitary capacitor to complete the capacitance. The capacitors should be laid out in a square fashion–with dummy capacitors completing the square–in order to minimize periphery-to-area mismatch.

An important source of mismatch is the base structure that the capacitor is laid upon. A nonuniform base structure can lead to variations in parasitics for the different capacitors. Surface discontinuities, such as those found in the thick oxide near the transistors, causes variations in the topography of the capacitor dielectrics. It would be desirable to keep the overall size of circuits to a minimum by placing the capacitor arrays close to the transistors. However, to achieve good matching, the capacitors should be placed well away from the thick oxide, and instead be placed over the field oxide. One should also place the circuits closer to the middle of the die area to avoid the bulk of the variation due to process stress on the devices. To minimize the process stress even further, one should cut of the corners on the capacitors[37]. And the optimal capacitor sizes, found to be around $20 - 50\mu m$[38, 39][4], should be used for the unit capacitor size.

There are also several ways to minimize the effect of noise through various layout techniques. To minimize noise under operation, no leads should be run over or close to the capacitors. Guardbars should be used to avoid substrate noise, especially if binary or high frequency analog circuits are present in the nearby area. One can also enclose the perimeter of the capacitor with dummy capacitors to reduce noise from any nearby lateral fields, and shielding can be used to avoid any noise from the vertical direction. The leads from and to the capacitors should be of equal width and laid out in a similar fashion to match the parasitics.

3.7 Noise Margins

The noise margins describes the amount of noise that can be present on the input of a circuit without a shift in the output level occurring. The upper and lower limits of the noise margins are shown in Figure 3.9. In the ideal case, the upper noise margin would equal the lower noise margin, and the transition between logic levels would be instantaneous. However, since the transition is actually non-instantaneous, there is a indeterminate region between the upper and the lower noise margin. The indeterminate region can be decreased by increasing the gain of the circuit in order to get a more abrupt transition.

It is desirable to have high gain inverters to improve signal detection–thereby improving the noise margins of the circuit. Unfortunately, high gain affects the speed of operation, as high gain FGUVMOS inverters require large coupling capacitance–since this increases the capacitive division ratio–to deliver a larger amount of the signal to the gate. The traditional way of increasing the length of the transistors will also effect the gain, but comes at the cost of reduced speed

[4]Although this is process dependent.

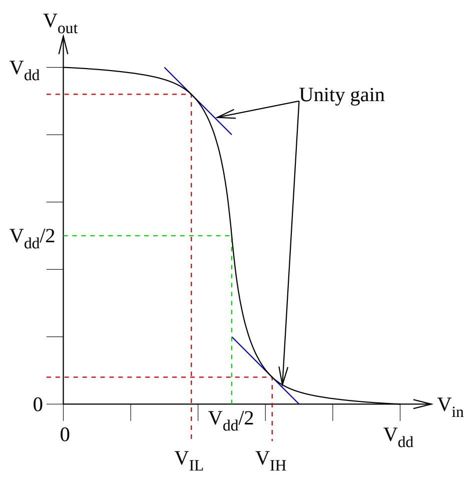


Figure 3.9: *Noise margins for an inverter circuit. The lower and upper noise margins, V_{IL} and V_{IH} respectively, are usually found by determining the unity gain at either end of the voltage swing. The output voltage in the region in between the lower and the upper noise margins is considered indeterminable. Thus no valid logic level should be greater than V_{IL} or less than V_{IH} (while obviously being greater than V_{IL}). The noise margin of the inverter can be increased by increasing the gain (making the transition more abrupt, thus increasing V_{IL} and decreasing V_{IH}).*

of operation. Both of these methods decrease the speed of operation. The first method by increasing the demand for current from the previous circuit element, while the second will reduce the supply of current to the next circuit element. Increasing the width of the transistor in order to compensate for the lack of available current, will only result in an increased area cost for the circuit, and a decrease in the gain and noise margins.

Although most of the problems regarding noise margins are due to process variances, signal noise might also affect the performance of the circuits. One of the prevalent noise sources in mixed signal circuits[5] is ground noise. Due to the resistance in the substrate[6], ground loops may form if attention is not payed to proper grounding methods[40]. Since higher frequency signals follow the path of least inductance, return currents will flow in as close proximity as possible to the signal currents. It is therefore important to separate the analog and digital portions of the circuit. No digital leads should flow near an analog circuit. Since the process stress is smaller–which in turn leads to less process variation–closer to the center of the die, as mentioned in section 3.6, the analog circuits relying on ca-

[5]The multiple-valued logic system presented in this thesis is in essence an analog system.
[6]This is not relevant for circuits based on SOI technology.

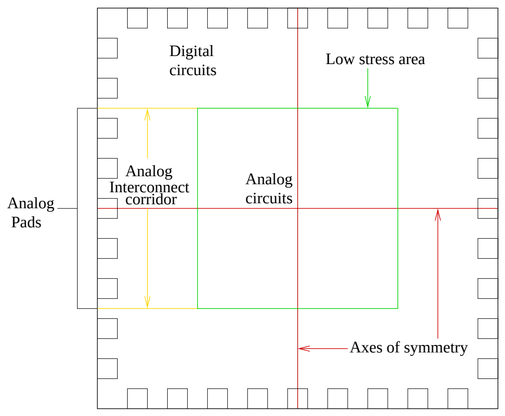


Figure 3.10: *Die layout for reduction in noise and process variation. The analog circuits (which are dependent on capacitor and transistor matching) should be placed in the low stress area near the middle of the die. Especially sensitive circuits should be placed on the axes of symmetry. A corridor of analog interconnects to the pads should be established where no digital interconnects are allow to cross.*

pacitor relationships should be placed here. The digital circuits can be placed closer to the die periphery. Another problem is power rail spikes resulting from massive amounts of current beeing drawn by the synchronous digital circuits during switching. Separate power supply rails should be provided for the analog and digital circuits. The ideal layout of the die can be seen in Figure 3.10.

3.8 Summary

In this chapter we started with chronicling the development toward active floating-gate devices. The FGUVMOS transistor was then presented with both the transistor symbol and a conceptual layout. We then examined closer the principal design parameter for FGUVMOS circuits, namely the capacitive voltage division. The transistor equations were then presented. These equations will lay the groundwork for developing the circuits in the next chapter. The programming technique–involving UV-activated conductances–was described, as well as the differences between the split-gate and common-gate configuration. Various methods for reducing the programming time was discussed. The layout methods for the FGUVMOS circuits was considered next. We saw that the ring transistor had several advantages over a regular transistor and that unit capacitor arrays would minimize capacitor mismatch. The importance of shielding with regards to parasitic UV-activated conductances was pointed out. Lastly, an explanation of circuit noise margins was given, and we looked at how various noise sources might have a negative impact on FGUVMOS circuit performance.

Chapter 4

FGUVMOS MVL Circuits

4.1 Introduction

All of the FGUVMOS multiple-valued logic circuits presented in this thesis are based on a few fundamental building blocks. These building blocks are the FGUVMOS binary inverter and the FGUVMOS multiple-valued inverter. Although it might seem restrictive to base all FGUVMOS MVL circuits on only these two building blocks, most of the complexity of the circuit design is actually placed into the FGUVMOS devices themselves. The main design feature of FGUVMOS MVL is the capacitive division relationships, and we will therefore make substantial use of this design parameter in order to realize the functions performed by the circuits presented in this chapter. From these fundamental building blocks, a binary to multiple-valued converter, a multiple-valued to binary converter and a multiple-valued full-adder will be presented.

Conversion between binary and multiple-valued signals are more efficient if some care is taken when choosing which radix to employ. By choosing a radix that is a power of two, as given in equation (2.4), no information will be left unused or discarded when converting between binary and multiple-valued logic systems. It would be natural to choose n in equation (2.4) to represent the number of bits in the binary system.

The circuits presented in this thesis are therefore all of radix eight, $r = 2^3 = 8$, although higher intermittent radix will be present in some of the circuit topologies. These higher intermittent radices will also be a power-of-two radix, thus satisfying equation (2.4). By using a different topology, it might be possible to remove these higher intermittent radix signals from the design all together, but possibly at the cost of increased area and power consumption.

Even though all of the circuits presented here are of radix eight, it is entirely feasible to use the same techniques for even higher radices. The use of a higher radix will, however, stress the noise margins of the MVL steps even further, making reliable signal detection difficult. It will also leave little room for capacitance mismatch, especially for cascaded systems. This can, however, be countered by increasing the dynamic range of the MVL circuits by increasing the supply voltage. Signal restoration circuits will have to be employed for cascaded systems to achieve reliable signal detection.

All of the simulations were performed in spectreS inside Cadence. The simulation setup for the different circuits can be seen in appendix A. The Matlab scripts used for the simulations can be found in appendix D.1. The measurements were performed on a chip fabricated in the AMS $0.6\mu m$ CMOS CUX process with three metal layers and two polysilicon layers. The layout of the circuits can be seen in appendix C. The equipment used to take the measurements–as well as the overall setup of the measurement equipment–can be seen in appendix B. The measurements were performed in Matlab (using a GPIB interface to the instruments) with the scripts in appendix D.2.

4.2 Binary Inverter

The binary inverter, which can be seen in Figure 4.1, is one of two fundamental building blocks in FGUVMOS MVL. The binary inverter consists of a pMOS transistor stacked upon an nMOS transistor, thus forming an inverter configuration. The nMOS transistor has a UV-hole covering the source diffusion and the gate in order to allow for programming of the circuit. Multiple voltage inputs are coupled to the floating-gate, each through a separate capacitor.

The response of the binary inverter to a piecewise linear voltage input is shown in Figure 4.2. We assume here that the inverter only has one input called V_{in}. The similarities between the standard static CMOS binary inverter and the FGUVMOS binary inverter are clearly shown. The main difference between them lies in the fact that the FGUVMOS binary inverter has multiple inputs available.

However, that difference is an important factor which makes the operating principles of the FGUVMOS binary inverter more akin to a capacitive threshold logic device[41]. Here the output is the result of a threshold operation. This is similar to how neurones in the nervous system will only fire if a preset threshold is exceed by the accumulated potential coming in from the synapses. This threshold operation can clearly be seen from the equation for the voltage output of the binary

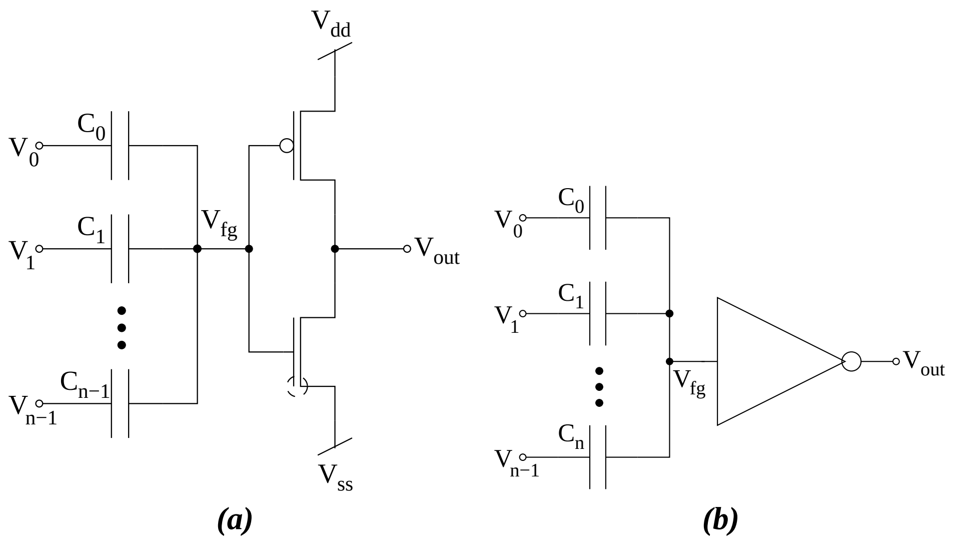


Figure 4.1: ***(a)*** *FGUVMOS binary inverter transistor schematics. The FGUVMOS binary inverter consists of an ordinary binary inverter with one or more voltage inputs with a capacitor signifying the capacitive coupling on the inputs.* ***(b)*** *FGUVMOS binary inverter symbol.*

inverter, given as

$$V_{out} = \begin{cases} 0 & \text{if} \quad \sum_{i=0}^{n-1} k_i V_i > \frac{V_{dd}}{2} \sum_{i=0}^{n-1} k_i \\ V_{dd} & \text{if} \quad \sum_{i=0}^{n-1} k_i V_i < \frac{V_{dd}}{2} \sum_{i=0}^{n-1} k_i \end{cases} \tag{4.1}$$

from which we can see that that if the combined weighted potential of the input voltages exceed a given threshold, here given as $V_{dd}/2$, the state of the output voltage will be altered. The term $\sum_{i=0}^{n-1} k_i = 1$, so the right-hand side of the inequality can be written simply as $V_{dd}/2$, which is the switching voltage all of the FGUVMOS circuits in this thesis are programmed to satisfy. However, as we shall see later on, these extra unity terms eases the solving of the equations used to find the capacitive division factors. Now, it should be mentioned that equation (4.1) is merely a simplification of the operating characteristics of the binary inverter that is adequate for the purpose of hand-calculations. A more complex model would use standard static CMOS binary inverter equations with the input voltage given in equation (3.2).

An important task in any electronic system is the ability to reliably detect the information contained within the signal, i.e. to decode the signal. As stated in section 2.5, this task is not as easily accomplished for MVL as for binary logic. The task of detecting the MVL steps in FGUVMOS MVL circuits is left up to the binary inverter. This detection process is depicted in Figure 4.3 and shows the response of the binary inverter to an MV (multiple-valued) signal.

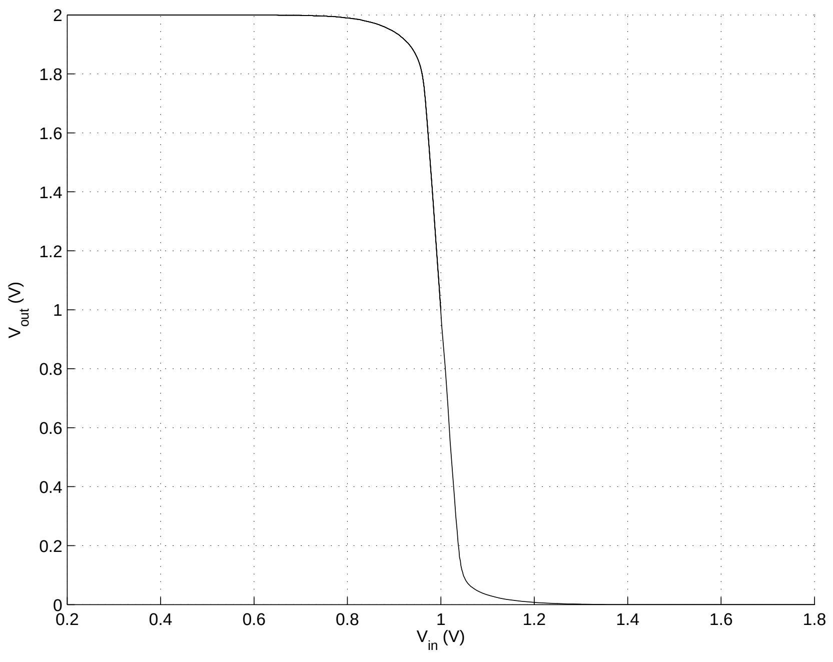


Figure 4.2: *This simulation shows the response of the FGUVMOS binary inverter to a linear input voltage. As can be seen from the figure, the response is similar to a standard static CMOS binary inverter. However, the FGUVMOS binary inverter has the advantage of being able to accept multiple inputs. The gain of the inverter is of special interest, since it directly affects the ability to detect multiple-valued signals.*

The binary inverter is able to discriminate signals above $V_{dd}/2$ from signals below $V_{dd}/2$ due to the threshold operation shown in equation (4.1). The accuracy of the discrimination depends upon two things. First of all, if the noise margins of the steps in the MV signal are high, then a higher degree of accuracy can be achieved. This stems from the simple fact that the gain of the binary inverter is not actually infinite, unlike what the abstraction of binary logic dictates. The operation of the binary inverter is highly analog in nature, as is evident in its response to a linear input voltage, shown in Figure 4.2. There are two ways to increase the noise margins. One can either increase the supply voltage[1], which will result in increased power consumption, or decrease the radix, thereby decreasing the available information contained in the signal.

The other main route that can be taken to increase the ability to discriminate, is to increase the gain of the inverter. This will reduce the input voltage range around

[1]It might not be possible to increase the supply voltage due to the possibility of device breakdown.

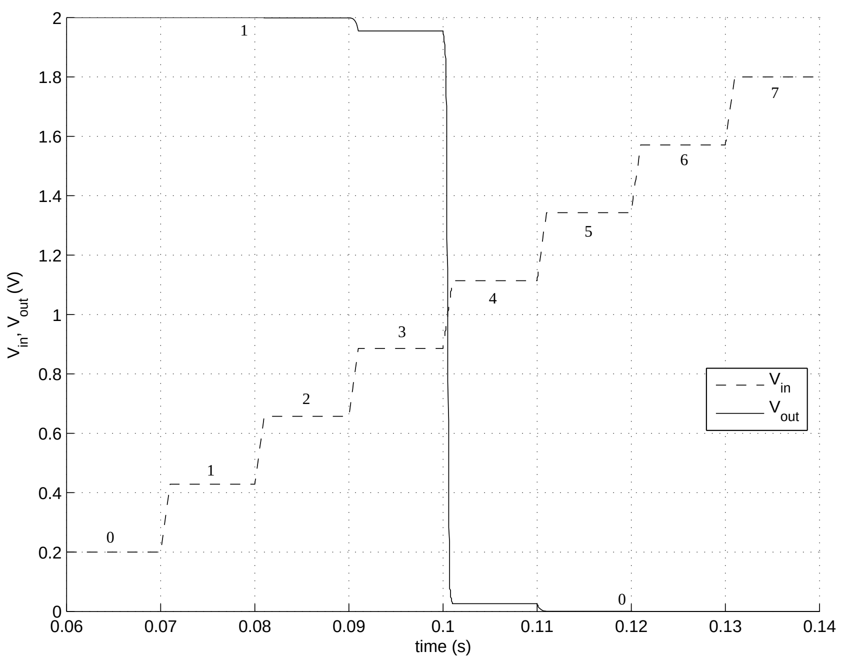


Figure 4.3: *In this simulation we can see the response of the FGUVMOS binary inverter to an MV signal. We can clearly see that the FGUVMOS binary inverter has the ability to detect the MSB of an MV signal on the form given in equation* (2.4). *We are also able to see how the gain of the FGUVMOS binary inverter affects the output signal for MVL steps around $V_{dd}/2$. The finite gain of the binary inverter causes the output signals not to extend all the way to the rails for these MVL steps.*

$V_{dd}/2$ where the output signal does not reach the rails, which was discussed in section 3.7. Therefore a high gain binary inverter is the key to achieving a higher noise margin without reducing the radix or increasing the supply voltage. There are two ways to achieve an increase to the gain of the binary inverter. One can either increase the length of the transistors–thereby reducing the output conductance and increasing the area consumption for the transistor–to get a higher gain directly. Or one can increase the coupling capacitor in relation to the MOS gate capacitance to deliver more of the signal to the gate, i.e. one can increase the capacitive division factor for the input signal.

Note that for high speed operations, the RC delay, involving the output conductance of the transistor and the capacitive load on the output, will have to be taken into consideration. This RC delay is a limiting factor which will reduce the noise margins if the circuits is operated above a certain speed. An increase in the width–which increases the available current–will solve this problem at the cost of in-

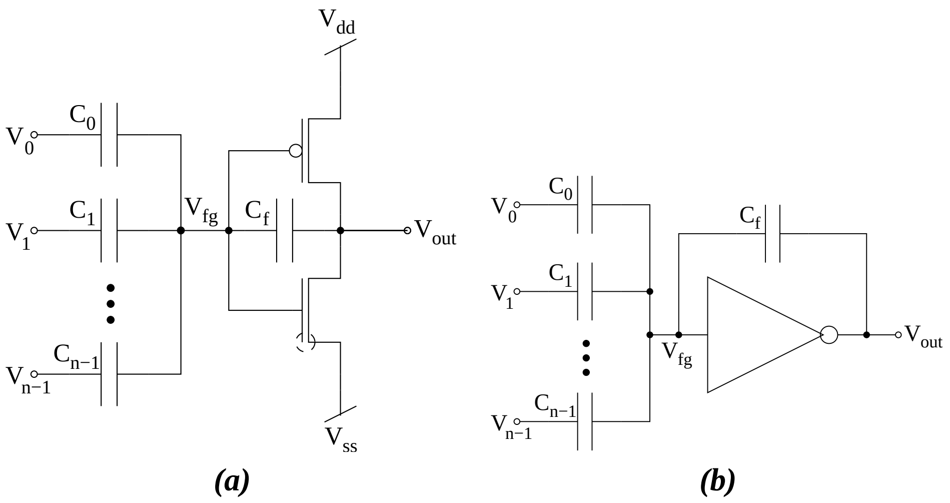


Figure 4.4: ***(a)****FGUVMOS multiple-valued inverter transistor schematic. The analog inverter consists of an FGUVMOS binary inverter with the addition of a feedback capacitor. This feedback capacitor ensures an extended linear region–at the cost of gain–by operating both transistors in saturation.* ***(b)*** *FGUVMOS multiple-valued inverter symbol.*

creased area and power consumption and a decrease in gain and noise margins.

4.3 Multiple-Valued Inverter

FGUVMOS MVL operations primarily rely on the multiple-valued inverter due to its highly linear operation over a large dynamic range. A single input multiple-valued inverter ideally delivers $V_{out} = V_{dd} - V_{in} = V_{in}^*$, i.e. the voltage output is the complement of the voltage input, which is the definition of an analog inverter. It also conforms to the definition of the multiple-valued unary negation operator seen in section 2.2. However, as the output approaches the rails, one of the transistors will enter the linear operating region–the pMOS transistor when we approach the V_{dd} rail and the nMOS transistor when we approach the V_{ss} rail–while the other transistor stays in the saturated region. This leads to a degradation of the linear performance–which in terms of multiple-valued logic circuits will lead to a compression of the dynamic range for the logic values near the rails–and consequently nonuniform voltage steps. It is therefore necessary to take appropriate measures in order to avoid this situation. Shibata[11] switched the pMOS and nMOS transistors to achieve the desired effect. Although a large linear dynamic range is achieved in this manner, it comes at some significant disadvantages. The decreased speed of the circuits and lack of gain control is two major concerns, and

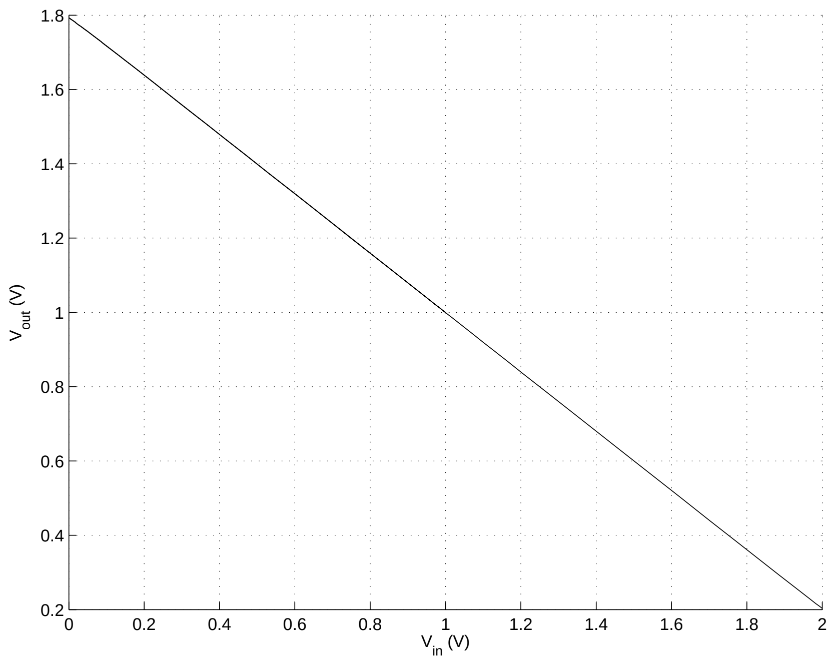


Figure 4.5: *A simulation showing the response of the FGUVMOS analog inverter to a linear voltage input. As can be seen from the figure, the output does not extend all the way to the rails, meaning the gain is less then $A_v = -1$. This is done on purpose do avoid the area near the rails where one of the transistors enters the linear operating region.*

another solution is therefore desirable.

Most operational amplifiers employ negative feedback to extend the linear dynamic range, and this is also the solution used by the multiple-valued inverter, which is shown in Figure 4.4. The multiple-valued inverter consists of a pMOS transistor stacked upon an nMOS transistor, forming an inverter configuration. It also has multiple capacitively coupled inputs connecting to a floating-gate. The only difference between the binary inverter and the multiple-valued inverter is the feedback capacitor. This feedback capacitor gives the multiple-valued inverter variable gain control, which can be used to avoid the problems of degradation of linear performance and is therefore a vital design parameter.

The only characteristic that separates an analog inverter, which produces the complement of the input on the output, and the FGUVMOS multiple-valued inverter, are the multiple inputs of the multiple-valued inverter. Since the multiple-valued inverter is not a physical multi-state device, as described in section 2.4, the idea of multiple discrete steps of logic values is merely an abstraction. The physical nature of the multiple-valued inverter is analog, thus the output signal is in reality

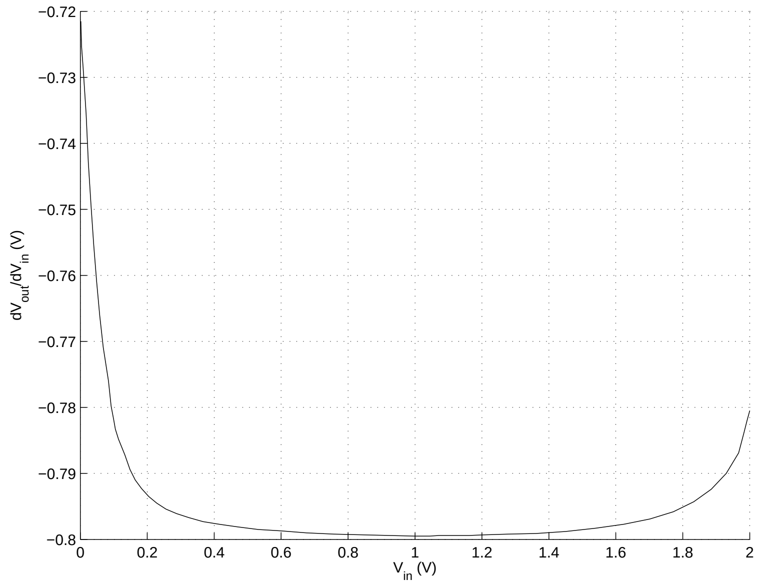


Figure 4.6: *A simulation of the voltage gain, A_v, of the FGUVMOS analog inverter. The voltage gain is fairly linear over the range of input voltages.*

continuous, both in time and value, as can be seen in Figure 4.5. Here we assume only one voltage input, called V_{in}, is capacitively coupled to the floating-gate.

The feedback capacitor from the output to the floating-gate allows for the gain to be arbitrarily set. The amplification can, in the ideal case, be tuned via $A_v = \frac{\sum_{i=0}^{n-1} C_i}{C_f}$. This allows for tuning the size of the linear dynamic range, seen in Figure 4.6, which shows the gain of the multiple-valued inverter for a piecewise linear input voltage.

The voltage output of the multiple-valued inverter can be found by setting $I_{ds,n} = I_{ds,p}$ and utilizing equations (3.3) - (3.4)[2]

$$I_{ds,n} = I_{ds,p}$$
$$\Updownarrow$$
$$I_{bec} \prod_{i=0}^{n-1} exp\left(\frac{1}{\eta U_T}(V_i - V_{dd}/2)k_i\right) \cdot exp\left(\frac{1}{\eta U_T}(V_{out} - V_{dd}/2)k_f\right)$$
$$= I_{bec} \prod_{i=0}^{n-1} exp\left(\frac{1}{\eta U_T}(V_{dd}/2 - V_i)k_i\right) \cdot exp\left(\frac{1}{\eta U_T}(V_{dd}/2 - V_{out})k_f\right)$$

[2]It should be noted that it is not a prerequisite that the transistors are operated in the weak inversion region.

$$\Updownarrow$$

$$\begin{aligned}
&\sum_{i=0}^{n-1} \frac{1}{\eta U_T}(V_i - V_{dd}/2)k_i + \frac{1}{\eta U_T}(V_{out} - V_{dd}/2)k_f \\
=&\sum_{i=0}^{n-1} \frac{1}{\eta U_T}(V_{dd}/2 - V_i)k_i + \frac{1}{\eta U_T}(V_{dd}/2 - V_{out})k_f
\end{aligned}$$

where the sums can be written out as follows

$$\begin{aligned}
&(V_0 - V_{dd}/2)k_0 + ... + (V_i - V_{dd}/2)k_i + ... + (V_{n-1} - V_{dd}/2)k_{n-1} + (V_{out} - V_{dd}/2)k_f = \\
&(V_{dd}/2 - V_0)k_0 + ... + (V_{dd}/2 - V_i)k_i + ... + (V_{dd}/2 - V_{n-1})k_{n-1} + (V_{dd}/2 - V_{out})k_f
\end{aligned}$$

and the terms can be grouped according to the different voltages

$$\begin{aligned}
2k_f V_{out} &= \left(\sum_{i=0}^{n-1} k_i + k_f\right) V_{dd} - 2\sum_{i=0}^{n-1} k_i V_i \\
V_{out} &= \left(\frac{\sum_{i=0}^{n-1} k_i}{k_f} + 1\right) V_{dd}/2 - \frac{\sum_{i=0}^{n-1} k_i V_i}{k_f}
\end{aligned}$$

The amplification factor, A_v, for the multiple-valued inverter is given as

$$A_v = \frac{\sum_{i=0}^{n-1} k_i}{k_f} \tag{4.2}$$

By substituting the equation for the gain into the expression for the voltage output, we get the final expression for the voltage output of the multiple-valued inverter

$$V_{out} = \left(\frac{1 + A_v}{2}\right) V_{dd} - \frac{\sum_{i=0}^{n-1} k_i V_i}{k_f} \tag{4.3}$$

where $k_i = C_i/C_T$, $k_f = C_f/C_T$ and $C_T = \sum_{i=0}^{n-1} C_i + C_f$ in the ideal case.

If the multiple-valued inverter has only one voltage input, called V_{in}, and we have unity gain, $A_v = k_{in}/k_f = 1$, then equation (4.3) reduces to

$$\begin{aligned}
V_{out} &= \left(\frac{1 + A_v}{2}\right) V_{dd} - \frac{k_{in} V_{in}}{k_f} \\
V_{out} &= V_{dd} - V_{in} = V_{in}^* \qquad (4.4)
\end{aligned}$$

and we get the analog inversion of the input. Equation (4.4) clearly show that the multiple-valued inverter is merely an analog inverter with multiple inputs and a tunable gain, which is given in equation (4.2). Both of these features of the

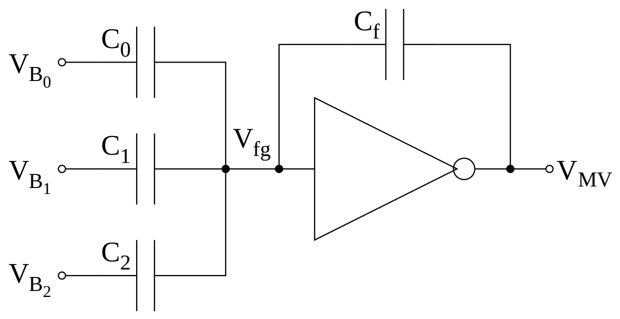


Figure 4.7: *FGUVMOS radix eight BMVC schematic. The BMVC is nothing more than the multiple-valued inverter building block with the correct capacitive division factors in place to perform the conversion operation. The number of capacitively coupled inputs equal $log_2 r$, where r is a radix on the form given by equation* (2.4).

multiple-valued inverter can be used as design parameters for constructing FGUVMOS MVL circuit, as we shall see in the course of this chapter.

Both of the fundamental building blocks are in an inverter configuration. This can lead to a problem of redundant inversions in cascaded systems if this is not taken into consideration in the overall circuit design. It should be noted that there is currently no overall design method for FGUVMOS MVL. The capacitance relationships in the building blocks can be decided by solving the equations presented in this section, and the previous one, through linear algebra. The topology of the circuit, however, has to be found through other means. Although we will see in the course of this chapter that finding the topology for arithmetic operations is a straight forward task.

4.4 Binary to Multiple-Valued Converter

Since the domain of digital circuitry mainly consists of binary circuits, there is a need to convert between FGUVMOS MVL and binary logic[3]. This is the purpose of the BMVC (binary to multiple-valued converter)–to provide an interface from the binary circuits to the FGUVMOS MVL circuits–which is essential for accomplishing integration with existing binary circuitry.

Sending multiple-valued signals off-chip has some desirable effects, such as a reduction in the number of interconnects on the PCB and a reduced number of

[3]The focus here is standard static CMOS binary logic.

pins on the chip[4]. It does, however, require a more sophisticated output buffer–to provide enough drive strength at high speeds–than for binary circuits. For binary circuits, it is common to simply use a pair of inverters as an output buffer. For high speeds or large loads, the output buffer will consists of several pairs of inverters, with increasing width toward the pad. This will further increase the drive current, although a delay will incur in the signal propagation. For multiple-valued circuits, signal levels have to be maintained. One way of maintaining signal levels is through a linear circuit capable of providing large amounts of drive current. Multiple-valued inverters, with increasing width in order to provide the necessary current, is one option. Another option is an operational amplifier in a source follower configuration. In both cases, linearity is the key to robust performance.

As can be seen from the circuit in Figure 4.7, the BMVC uses capacitive division relationships to achieve its operating function. It is also clear from the schematic that the BMVC is nothing more than one of the fundamental building blocks–namely the multiple-valued inverter–with the correct capacitive division factors in place. To determine these capacitive division factors–thereby finding the capacitances for the coupling capacitors–we will first need to develop the output voltage function for the BMVC. We will also have to examine the gain of the BMVC–which relates to the linear dynamic range of the output signal–before the capacitor relationships can be determined.

The voltage output of the BMVC can be obtained by rewriting equation (4.3)

$$V_{MV} = \left(\frac{1 + A_v}{2}\right) V_{dd} - \frac{\sum_{i=0}^{n-1} k_i V_i}{k_f} \tag{4.5}$$

The measurement setup for the BMVC can be seen in appendix B. For the measurement of the linear performance of the BMVC, we can assume the circuit only has one voltage input[5], called V_{Bn}. A piecewise linear voltage signal is applied to the input, resulting in the output voltage, V_{MV}, shown in Figure 4.8. To assure an even spread of the MVL steps–thus a uniform size for all the logic levels–a high degree of linearity over the entire dynamic range is needed. Figure 4.9 examines the linearity closer by looking at the deviance from ideal linearity. The deviance is small–on the order of a few mV–and is probably due to mismatch in the capacitor sizes and possibly some contributions from the various parasitic capacitances depicted in Figure 3.3.

The gain of the BMVC is purposely set below unity, which can be seen in Figure

[4]However, since most of the digital circuits of today are binary in nature, it is advantageous to send the signals off-chip in a binary form.

[5]This can be achieved for a multiple-input FGUVMOS device by connecting the inputs together, which was done for the measurements in this thesis.

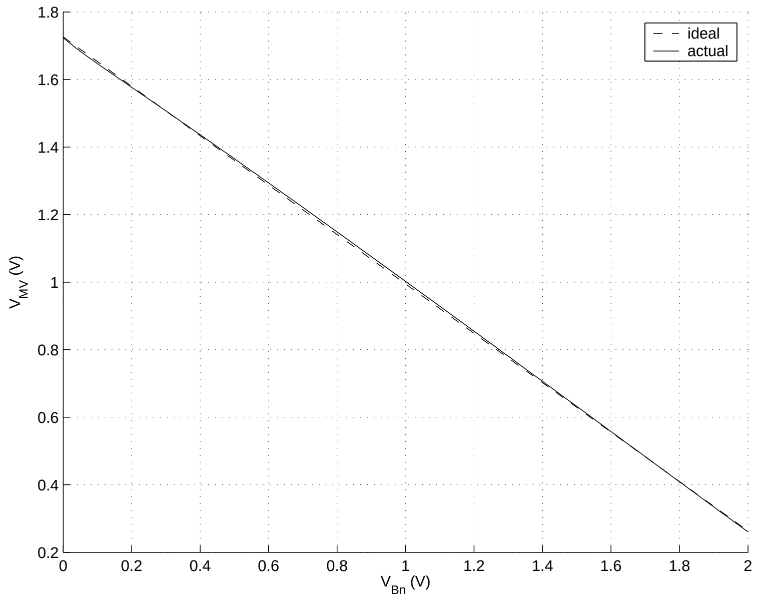


Figure 4.8: *The measurement of the response of the BMVC to a piecewise linear voltage input is shown as the solid line. The ideal voltage response is depicted as the dashed line and is match to a first order to the amplification factor of the BMVC.*

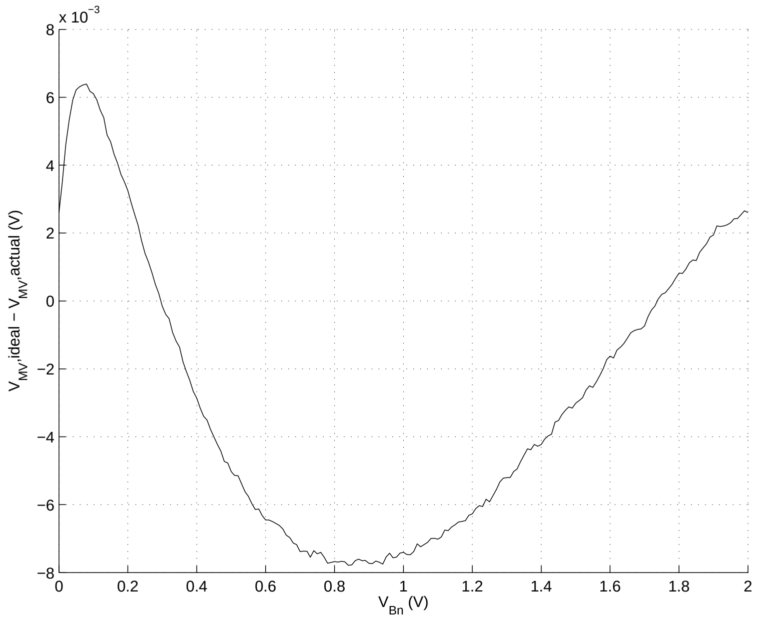


Figure 4.9: *The deviance from the ideal response for the measured response of the BMVC to a piecewise linear voltage input.*

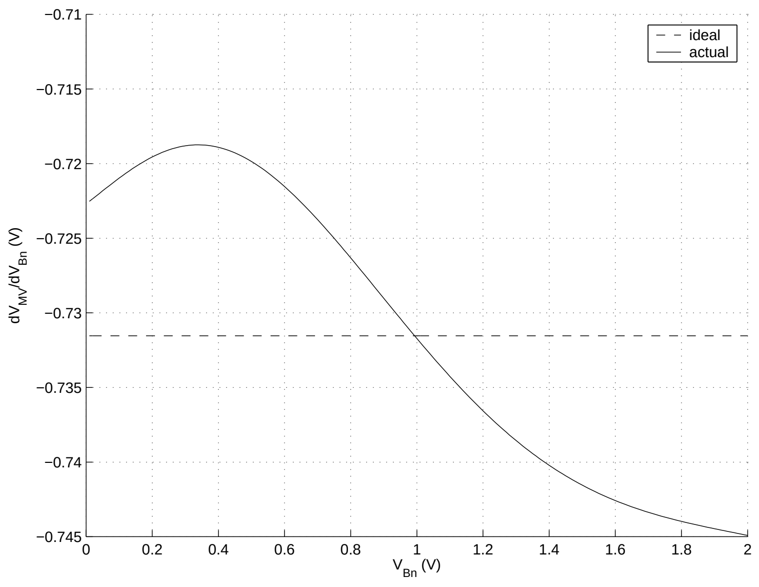


Figure 4.10: *The voltage gain of the measured response of the BMVC to a piece-wise linear voltage input is shown as the solid line. The line is smoothed using cubic splines to better show the trend. The dashed line shows the amplification factor of the ideal response.*

4.10, to avoid discrepancies near the rails. The cause of these discrepancies is the transition of the transistors from the saturated region to the linear operating region. While a lower gain reduces the dynamic range of the circuit, it is necessary to avoid compression of the dynamic range of the logic levels near the rails in order for the MVL steps to be of uniform size. Shibata[11] switched the nMOS and pMOS transistors to achieve this effect. A better method, as employed by the BMVC, is to introduce a feedback capacitor to get a tunable voltage gain. By adjusting the size of the feedback capacitor, C_f, we can adjust the size of the linear dynamic range by assuring that the two transistors remain in the saturated region for a greater range of output voltages.

Several measures might be taken to avoid the non-linear behavior–which leads to a compression of the MVL steps near the rails–associated with unity gain. This non-linear behavior is caused by one of the transistors leaving the saturated region–and entering the linear region–when the output signal approaches the rail. By limiting the input voltage range–while maintaining unity gain–the input voltage will not approach the rails. Neither will the output voltages due to the unity gain. However, this necessitates limiting the voltage range of the input signal. A natural place to perform such a voltage range limiting operation would be as a part of the binary to multiple-valued conversion[6]. If the voltage amplification, A_v, is set below unity, this would impose such a voltage range limiting operation. The amplification would have to be set sufficiently low as to impede either of the transistors from leaving the saturated region.

In general, the voltage gain of a circuit is expressed as $A_v = V_{out}/V_{in}$. We can define the output voltage range, V_{DR}, for which the transistors operate in the saturated region–thereby assuring a linear output voltage over this voltage range. The input voltage is binary, therefore the gain becomes $A_v = V_{DR}/V_{dd}$, assuming $V_{ss} = 0V$. As stated, the output voltage range should only include the transistor operating region where both transistors are in saturation. This can be achieved by choosing the boundaries of the output voltage range to match the saturation voltages for the transistors, thus $V_{DR} = V_{dsat,p} - V_{dsat,n}$. For the switching point voltage–where both the transistors are in saturation–we have $V_{in} = V_{out} = V_{dd}/2$, due to the programming of the circuit. The saturation voltages–the voltage points where the transistors enters the saturated region–is given as

$$V_{dsat,p} = V_{dd} - V_{gs} - V_{t,p} = V_{dd}/2 - V_{t,p} \tag{4.6}$$

$$V_{dsat,n} = V_{gs} - V_{t,n} = V_{dd}/2 - V_{t,n} \tag{4.7}$$

which gives us the following dynamic range for which both the transistors are in saturation

$$\begin{aligned} V_{DR} &= V_{dsat,p} - V_{dsat,n} = V_{dd}/2 - V_{t,p} - (V_{dd}/2 - V_{t,n}) \\ &\Updownarrow \\ V_{DR} &= -V_{t,p} + V_{t,n} \end{aligned} \tag{4.8}$$

For the AMS 0.6μm CUX CMOS process, a value of $A_v = 0.8$ was found to be appropriate.

By assuming unity gain, we can more easily solve the equations for finding the appropriate capacitor sizes for the coupling capacitors and the feedback capacitor. To transform the circuit from unity gain, $A_{v,unity}$, back to the original gain, A_v, we can multiply the gain of the BMVC, given by equation (4.2), by the actual gain desired

$$A_{v,unity} \cdot A_v = \frac{\sum_{i=0}^{n-1} k_i}{k_f} \cdot A_v = \frac{\sum_{i=0}^{n-1} A_v \cdot k_i}{k_f} = \frac{\sum_{i=0}^{n-1} k_i}{k_f/A_v} \tag{4.9}$$

which means the correct capacitance for the original gain, A_v, can be retrieved from a unity gain circuit by either multiplying all of the input coupling capacitances by A_v or by dividing the feedback capacitance by A_v.

[6]The BMVC is not the only place in FGUVMOS MVL where binary to multiple-valued signal conversions occur, as we shall see when we examine the multiple-valued full-adder.

B_2	B_1	B_0	$\overline{MV}$
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Table 4.1: *Truth table for the radix eight BMVC. The first three columns lists the binary input. The last column lists the logic values for the multiple-valued output. Note that the output is inverted.*

The capacitances for the BMVC can now be found by assuming unity gain, $A_v = 1$. The actual size of the capacitances for the BMVC in this thesis, which has a gain of $A_v = 0.8$, can be retrieved as described above. We first start by writing out the sums of the complement of equation (4.5), taking into account the statements made above

$$V_{MV}^{*} = k_0 V_{B_0} + k_1 V_{B_1} + ... + k_{n-1} V_{B_{n-1}} \tag{4.10}$$

where $k_i = \frac{C_i}{C_T}$ and $C_T = \sum_{i=0}^{n-1} C_i + C_f$ in the ideal case[7].

More specifically, the radix eight BMVC has three input capacitances

$$V_{MV}^{*} = k_0 V_{B0} + k_1 V_{B1} + k_2 V_{B2} \tag{4.11}$$

where the capacitive division relationships are unknown and have to be decided upon to achieve the desired output function. The truth table of the radix eight BMVC, listed in Table 4.1, shows the desired logic levels for the multiple-valued output compared to the binary input.

The measured response of the BMVC to a three-bit binary signal going from 000 to 111 in logic value is shown in Figure 4.11. As we can see from the figure, the output logic levels are evenly distributed over the entire dynamic range, thus ensuring a uniform size for the logic value steps in the MV signal. The output signal does not swing from rail to rail, thus demonstrating the effect of the reduced gain of the BMVC, which is needed to ensure a linear performance for the entire dynamic range. This reduction in the gain of the BMVC will lead to a reduction in the size of the MVL steps, making the circuit more sensitive to noise or device mismatch resulting from the processing step. The inverting property of the BMVC

[7]We disregard the parasitic capacitances in the ideal case.

V_{B_2}	V_{B_1}	V_{B_0}	V^*_{MV}	$V^*_{MV_*}$
0	0	0	0	0.200
0	0	2	$\frac{1}{7}V_{dd}$	0.429
0	2	0	$\frac{2}{7}V_{dd}$	0.657
0	2	2	$\frac{3}{7}V_{dd}$	0.886
2	0	0	$\frac{4}{7}V_{dd}$	1.114
2	0	2	$\frac{5}{7}V_{dd}$	1.343
2	2	0	$\frac{6}{7}V_{dd}$	1.571
2	2	2	V_{dd}	1.800

Table 4.2: *Voltage table for the radix eight BMVC. The three first columns lists the binary input voltages. The next to last column lists the multiple-valued output voltages for a unity gain BMVC. The last column lists the multiple-valued output voltages for a BMVC with a gain of $A_v = 0.8$, which is the gain found adequate for the AMS $0.6\mu m$ CUX CMOS process used in this thesis. Note that the output of the BMVC is inverted.*

is also clearly shown in the figure. There is, however, a small deviance from the ideal response. This can further be seen in Figure 4.12 where the difference between the measured response and the ideal response is shown. Although the largest deviation is around $10mV$, this is less than $1/20$th the size of the MVL step size and therefore only of concern for cascaded systems.

While the logical values are given as the sequence $\{0, ..., i, ..., 7\}$ for a radix eight FGUVMOS MVL circuit, the actual voltage levels, assuming unity gain, are given as $\{0, ..., \frac{i}{r-1}V_{dd}, ..., V_{dd}\}$. Table 4.2 relates the binary input voltages to the multiple-valued output voltage. From this table we should be able to determine the capacitance–or at least the relationship between the capacitor values–for the coupling capacitors.

The might be a way of intuitively grasping the capacitor relationships for the BMVC. The binary input B_2 is twice as significant as the input B_1, which in turn is twice as significant as B_0. Coupled with the linear operation of the BMVC, one would come to the conclusion that the capacitors should probably conform to the same relationship. This naturally leads to $C_2 = 2C_1 = 2C_0$. However, a more formal approach may be appreciated.

Since we have three unknown variables, namely C_T, C_1 and C_2[8], we will need three equations to find the capacitance values of the circuit. It should be noted that certain values of the input greatly eases the solving of these equations. The first (starting from zero), second and fourth entry in Table 4.2 are used to construct

[8] C_0 is chosen as the base capacitance, and therefore be of an arbitrary size.

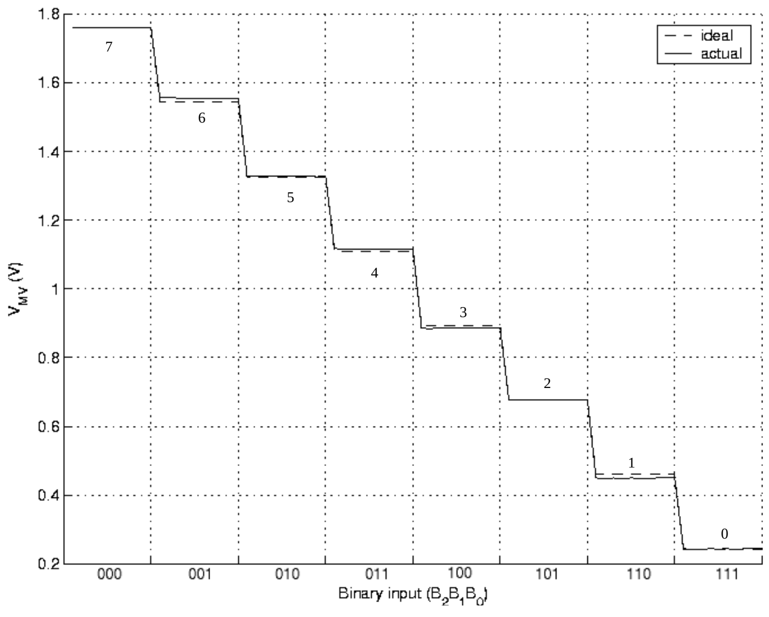


Figure 4.11: *Measurement of the response of the BMVC to a binary voltage input signal. The binary input is given in Table 4.1. The x-axis lists the logic values for the binary inputs, since the waveforms of the binary input signals are of little interest.*

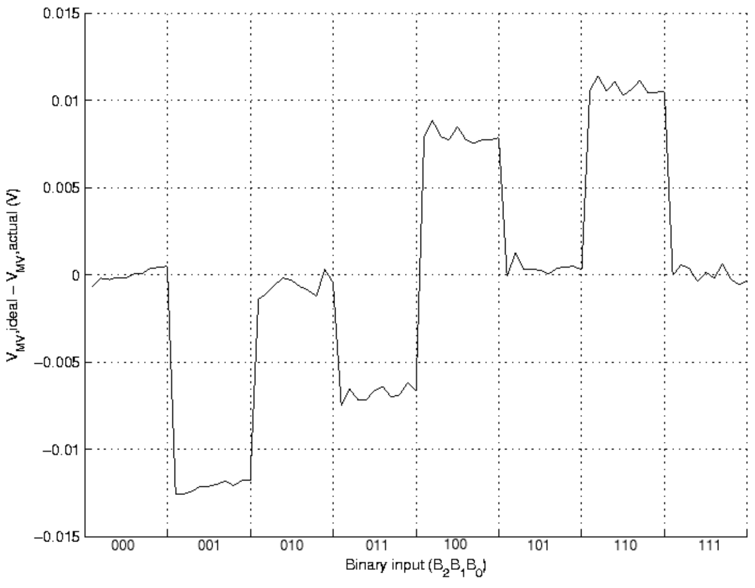


Figure 4.12: *Deviance from ideal response for the measured response of the BMVC to a binary voltage input signal.*

the equations

$$\begin{aligned}
\text{I:}\quad & \frac{1}{7}\cdot 2 = k_0\cdot 2 + k_1\cdot 0 + k_2\cdot 0 \Leftrightarrow k_0 = \frac{1}{7}\\
\text{II:}\quad & \frac{2}{7}\cdot 2 = k_0\cdot 0 + k_1\cdot 2 + k_2\cdot 0 \Leftrightarrow k_1 = \frac{2}{7}\\
\text{III:}\quad & \frac{4}{7}\cdot 2 = k_0\cdot 0 + k_1\cdot 0 + k_2\cdot 2 \Leftrightarrow k_2 = \frac{4}{7}
\end{aligned}$$

We can see that the total sums up to one, $k_0 + k_1 + k_2 = \frac{1}{7} + \frac{2}{7} + \frac{4}{7} = 1$. By substituting $k_i = \frac{C_i}{C_T}$, we get

$$\text{I:}\quad k_0 = \frac{C_0}{C_T} = \frac{1}{7} \Leftrightarrow C_T = 7C_0 \tag{4.12}$$

$$\text{II:}\quad k_1 = \frac{C_1}{C_T} = \frac{2}{7} \Leftrightarrow C_1 = 2C_0 \tag{4.13}$$

$$\text{III:}\quad k_2 = \frac{C_2}{C_T} = \frac{4}{7} \Leftrightarrow C_2 = 4C_0 \tag{4.14}$$

which means that once we have determined the base capacitor, C_0–which is usually set to the minimum size capacitor for the process at hand to reduce area consumption–the rest of the capacitor sizes are given in the equations above. Although it might be necessary to adjust the capacitor sizes for a non-unity gain BMVC–which was shown in equation (4.9).

The cost of higher radices increase in a linear fashion. For each increment in a power-of-two radix–as specified in equation (2.4)–one extra capacitor has to be added to the converter. This capacitance will have to be twice the size of the current MSB-capacitance. The feedback capacitor would also have to be increased with twice the size of the current MSB-capacitance. This type of cost increase–which also holds for the converter circuit presented in the next section–is in accordance with the first cost function presented in section 2.3, equation (2.2).

4.5 Multiple-Valued to Binary Converter

A method for converting from multiple-valued signals to binary signals is needed to complement the BMVC. We may need to convert to binary signal levels before going off-chip to communicate with binary peripheral circuits. The need for interaction with binary modules in a mixed signal integrated circuit is another relevant

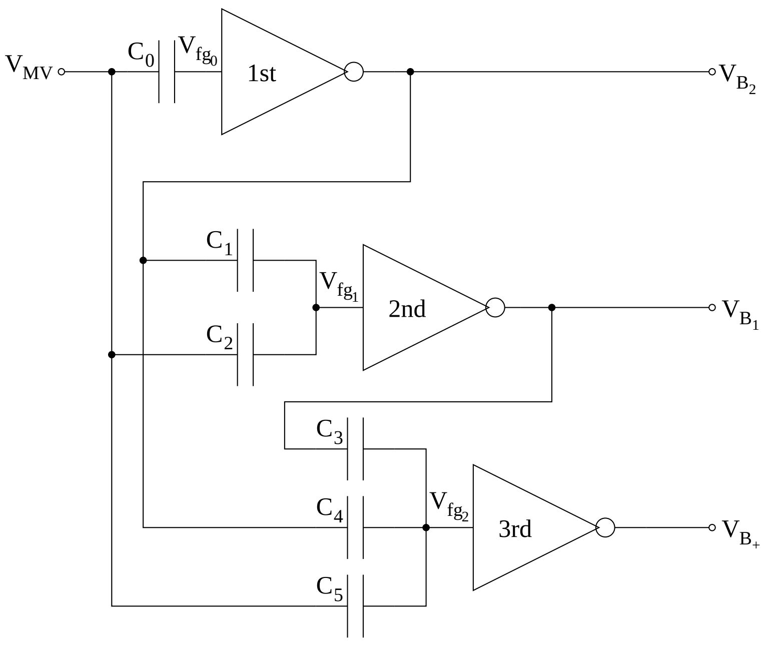


Figure 4.13: *FGUVMOS radix eight MVBC schematic. The circuit has one multiple-valued input, called V_{MV} and $log_2 r$ binary outputs–with r defined in equation* (2.4)*–called V_{B_i}. The MVBC has $log_2 r$ stages. Each stage is responsible for converting one of the $log_2 r$ bits in the binary output. The FGUVMOS binary inverter in the first stage can be exchanged with a standard static CMOS binary inverter (as can all single input FGUVMOS binary inverters). However, the benefits of the UV-programming will then be lost.*

need. With a multiple-valued to binary converter, we will have a comprehensive interface between binary logic and FGUVMOS MVL.

The FGUVMOS MVL circuit that accomplishes this task is called the MVBC (multiple-valued to binary converter). As can be seen from Figure 4.13, the MVBC only makes use of the FGUVMOS binary inverter. This is an obvious choice, considering that a single input binary inverter actually preforms a one-bit AD (analog-to-digital) conversion. Since this is a radix eight MVBC–which results in a three-bit binary output–there are three stages in the circuit. Each stage consists of one binary inverter responsible for converting one of the bits in the binary output. The simulation of the MVBC in response to a multiple-valued input signal is shown in Figure 4.14.

As previously mentioned, the binary inverter preforms a one-bit AD operation, due to the fact that the switching point is programmed to $V_{dd}/2$. Thus, assuming ideal behavior, the output voltage of the single input binary inverter, with the input

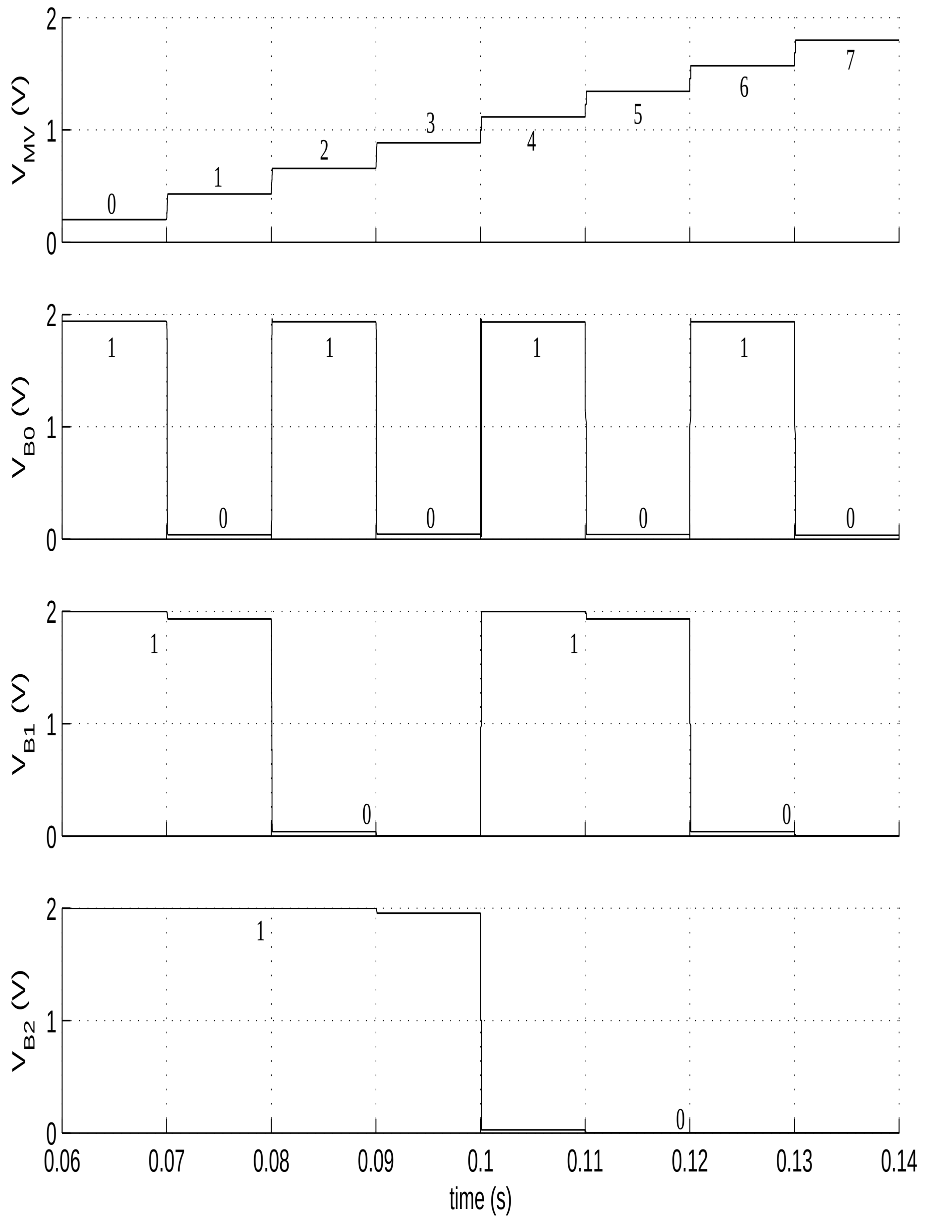


Figure 4.14: *A simulation of the response of the MVBC to a multiple-valued input signal. The deviance from the rail in the voltage outputs is due to the finite voltage gain of the binary inverters. If a rail-to-rail voltage signal is desired, then a binary buffer will most likely have to be added to the binary outputs.*

V_{MV} and the output V_B, is given as

$$V_B = \begin{cases} 0 & \text{if} \quad V_{MV} > \frac{V_{dd}}{2} \\ V_{dd} & \text{if} \quad V_{MV} < \frac{V_{dd}}{2} \end{cases}$$

according to equation (4.1) with the input voltage called V_{MV} and the output voltage called V_B. The voltage and logic value of the input are related in the following manner

$$V_{MV} = \frac{MV}{r-1} V_{dd}$$

where MV is the logic value of the multiple-valued input signal. We can then express the function of the single input binary inverter in terms of logic values in the following manner

$$B = \begin{cases} 0 & \text{if} \quad MV > \frac{r-1}{2} \\ 1 & \text{if} \quad MV < \frac{r-1}{2} \end{cases}$$

where B is the logic value of the binary output signal. We can see that the binary inverter will convert the MSB (most significant bit) in an MV signal on the form given in equation (2.4), since it can detect whether an MV signal is lesser or greater than $\frac{r-1}{2}$. Note that the output will be inverted due to the inverting property of the circuit.

The conversion operation, in terms of logic values, for the radix eight MVBC is given in Table 4.3. The conversion procedure consists of several stages. At the first stage, the AD conversion can be done directly using one single input binary inverter, since the transition point for the MSB already lies at $V_{dd}/2$. This can be seen in Table 4.4, which lists the voltages corresponding to the logic levels given in Table 4.3. We can assume that the multiple-valued signal has a rail-to-rail dynamic range in order to make the equations a little easier to solve.

The transition of the next bit, MSB-1, does not only lie at $V_{dd}/2$. According to Figure 4.15, which combines Table 4.3 and Table 4.4, the set of transition points is given as

$$\left\{ \frac{3}{14} V_{dd}, \frac{V_{dd}}{2}, \frac{11}{14} V_{dd} \right\}$$

Since the binary inverters are programmed to switch at $V_{dd}/2$, we need to shift the MV signal so the relevant transition points will be seated at $V_{dd}/2$. For the transition point $\frac{3}{14} V_{dd}$ we need to add $\frac{4}{14} V_{dd}$. For the transition point $\frac{11}{14} V_{dd}$ we

MV	B_2	B_1	B_0
0	1	1	1
1	1	1	0
2	1	0	1
3	1	0	0
4	0	1	1
5	0	1	0
6	0	0	1
7	0	0	0

Table 4.3: *Truth table for the radix eight MVBC. The first column lists the multiple-valued input signal. The three last columns lists the binary output signals. Note that the outputs are inverted.*

V_{MV}	V_{MV*}	V_{B_2}	V_{B_1}	V_{B_0}
0	0.200	2	2	2
$\frac{1}{7}V_{dd}$	0.429	2	2	0
$\frac{2}{7}V_{dd}$	0.657	2	0	2
$\frac{3}{7}V_{dd}$	0.885	2	0	0
$\frac{4}{7}V_{dd}$	1.114	0	2	2
$\frac{5}{7}V_{dd}$	1.343	0	2	0
$\frac{6}{7}V_{dd}$	1.571	0	0	2
V_{dd}	1.800	0	0	0

Table 4.4: *Voltage table for the radix eight MVBC. The first column lists the multiple-valued input for a non-limited voltage signal. The next column list a voltage limited multiple-valued input signal which is used by the circuits in this thesis. The three last columns lists the binary output voltages. Note that the outputs are inverted.*

need to subtract $\frac{4}{14}V_{dd}$. And for the transition point $V_{dd}/2$, no shifting of the MV signal should occur.

The voltage shift on the floating-gate can be found by looking at the following part of equation (3.3)

$$k_1(V_1 - V_{dd}/2) + k_2(V_2 - V_{dd}/2)$$

For this stage we have two signals available, namely the MV signal, V_{MV}, and the MSB signal, V_{B_2}. For the transition point $V_{MV} = \frac{3}{14}V_{dd}$, the MSB is given as $V_{B_2} = V_{dd}$ according to Figure 4.15. The shift in the floating-gate voltage for this transition point is then given as

$$k_1(V_{dd} - V_{dd}/2) + k_2(V_{MV} - V_{dd}/2)$$

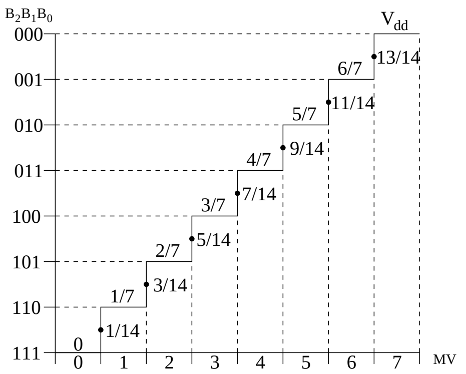


Figure 4.15: *A combination of Table 4.3 and Table 4.4 in order to determine the transitions in the binary output signals for the different transition points in the multiple-valued input signal. The logic levels of the multiple-valued input signal is given along the x-axis and the logic levels of the binary output signal is given along the y-axis. The multiple-valued input voltages are given along the staircase, along with the voltage for the transition points, marked by the bullet points. These last two sequences of values should all be multiplied by V_{dd}.*

$$= k_1 V_{dd}/2 + k_2(V_{MV} - V_{dd}/2)$$

which adds $k_1 V_{dd}/2$ to the MV signal. For the transition point $V_{MV} = \frac{11}{14}V_{dd}$, the MSB is given as $V_{B_2} = 0V$. This gives us a shift in the floating-gate voltage of

$$\begin{aligned} &k_1(0V - V_{dd}/2) + k_2(V_{MV} - V_{dd}/2) \\ &= -k_1 V_{dd}/2 + k_2(V_{MV} - V_{dd}/2) \end{aligned}$$

and $k_1 V_{dd/2}$ is subtracted from the MV signal. For the last transition point, $V_{MV} = V_{dd}/2$, the MSB is also at the transition point and is given as $V_{B_2} = V_{dd}/2$. The shift in the floating-gate voltage then becomes

$$\begin{aligned} &k_1(V_{dd}/2 - V_{dd}/2) + k_2(V_{MV} - V_{dd}/2) \\ &= k_2(V_{MV} - V_{dd}/2) \end{aligned}$$

and the MV signal is not shifted. Table 4.5 lists the floating-gate voltage on the second stage binary inverter based on the transition points. The pattern of voltage shifts of MV signal in this table coincides with the assumptions made above. However, in order for the MV signal to be shifted by the correct amount, the correct capacitor values will first have to be found.

The same procedure of shifting the MV signal can also be employed by the LSB stage. Here the transition point set is given as

$$\left\{\frac{1}{14}V_{dd}, \frac{3}{14}V_{dd}, \frac{5}{14}V_{dd}, \frac{V_{dd}}{2}, \frac{9}{14}V_{dd}, \frac{11}{14}V_{dd}, \frac{13}{14}V_{dd}\right\}$$

V_{MV}	V_{B_2}	V_{fg_1}
$\frac{3}{14}V_{dd}$	V_{dd}	$k_1\frac{V_{dd}}{2} + k_2\left(V_{MV} - \frac{V_{dd}}{2}\right)$
$\frac{V_{dd}}{2}$	$\frac{V_{dd}}{2}$	$k_2\left(V_{MV} - \frac{V_{dd}}{2}\right)$
$\frac{11}{14}V_{dd}$	$0V$	$-k_1\frac{V_{dd}}{2} + k_2\left(V_{MV} - \frac{V_{dd}}{2}\right)$

Table 4.5: *MVBC second stage transition voltages. The first column lists the transition points for the multiple-valued signal. The next column lists the value of the MSB signal at those transition points. The last column lists the shifts in the floating-gate voltage of the second stage binary inverter.*

V_{MV}	V_{B_2}	V_{B_1}	V_{fg_2}
$\frac{1}{14}V_{dd}$	V_{dd}	V_{dd}	$k_3\frac{V_{dd}}{2} + k_4\frac{V_{dd}}{2} + k_5\left(V_{MV} - \frac{V_{dd}}{2}\right)$
$\frac{3}{14}V_{dd}$	V_{dd}	$\frac{V_{dd}}{2}$	$k_4\frac{V_{dd}}{2} + k_5\left(V_{MV} - \frac{V_{dd}}{2}\right)$
$\frac{5}{14}V_{dd}$	V_{dd}	$0V$	$-k_3\frac{V_{dd}}{2} + k_4\frac{V_{dd}}{2} + k_5\left(V_{MV} - \frac{V_{dd}}{2}\right)$
$\frac{V_{dd}}{2}$	$\frac{V_{dd}}{2}$	$\frac{V_{dd}}{2}$	$k_5\left(V_{MV} - \frac{V_{dd}}{2}\right)$
$\frac{9}{14}V_{dd}$	$0V$	V_{dd}	$k_3\frac{V_{dd}}{2} - k_4\frac{V_{dd}}{2} + k_5\left(V_{MV} - \frac{V_{dd}}{2}\right)$
$\frac{11}{14}V_{dd}$	$0V$	$0V$	$-k_4\frac{V_{dd}}{2} + k_5\left(V_{MV} - \frac{V_{dd}}{2}\right)$
$\frac{13}{14}V_{dd}$	$0V$	$0V$	$-k_3\frac{V_{dd}}{2} - k_4\frac{V_{dd}}{2} + k_5\left(V_{MV} - \frac{V_{dd}}{2}\right)$

Table 4.6: *MVBC third stage transition voltages. The first column lists the transition points for the multiple-valued signal. The next column lists the value of the MSB signal at those transition points. The third column lists the value of the MSB-1 signal at the multiple-valued signal transition points. The last column lists the shifts in the floating-gate voltage of the third stage binary inverter.*

Again we need to shift the transition points in the MV signal to $V_{dd}/2$. The voltage shift on the floating gate of the third stage binary inverter is given as the following part of equation (3.3)

$$k_3(V_3 - V_{dd/2}) + k_4(V_4 - V_{dd}/2) + k_5(V_5 - V_{dd}/2)$$

For the third stage binary inverter we have three signals available, namely the MSB signal, V_{B_2}, the MSB-1 signal, V_{B_1}, and the MV signal, V_{MV}. Table 4.6 lists the floating-gate voltage shifts on the third stage inverter based on the transition point set given above.

Again the correct pattern for shifting the MV signal has been found. By using Figure 4.15–which includes the voltages at the potential transition points–we can also find the correct capacitor values. From equation (4.1) for the binary inverter, we can see that by setting

$$\sum_{i=0}^{n-1} k_i V_i = \frac{V_{dd}}{2}\sum_{i=0}^{n-1} k_i \tag{4.15}$$

we can find the correct capacitor relationships–which sets the voltage threshold switching point–needed to get the desired operating function from the circuit. By setting the correct capacitor values, we are not actually altering the switching point of the binary inverter–which is already programmed to be $V_{dd}/2$. We are instead weighting the voltage inputs in such a manner that the effective switching point, with regards to the inputs, is shifted.

The capacitor value of the first stage in the MVBC can be set freely. For capacitances that can be set freely, the minimum capacitance is usually chosen. The justification is reduced area consumption, although choosing a larger capacitance will increase the gain of the stage. For the second stage conversion, we can set the capacitance for the MSB input, C_1, freely. We can then express the capacitance of the MV signal as a scale factor of C_1. For the transition $V_{MV} = \frac{1}{7}V_{dd}$ to $V_{MV} = \frac{2}{7}V_{dd}$, the MSB signal is V_{dd}. The switching voltage for this transition is $V_{MV} = \frac{3}{14}V_{dd}$. The capacitance division relationships for the first stage is then, according to equation (4.15)

$$
\begin{aligned}
k_1 V_{B_2} + k_2 V_{MV} &= \frac{V_{dd}}{2}\,(k_1 + k_2) \\
k_1 V_{dd} + k_2 \frac{3}{14} V_{dd} &= \frac{V_{dd}}{2} k_1 + \frac{V_{dd}}{2} k_2 \\
(1 - \frac{1}{2}) k_1 &= (\frac{1}{2} - \frac{3}{14}) k_2 \\
k_2 = \frac{7}{4} k_1 &\Leftrightarrow C_2 = \frac{7}{4} C_1
\end{aligned}
$$

We can now verify that the capacitor ratios are correct with regards to the voltage shifts on the floating-gate. The capacitor ratios for the second stage is given as

$$
\begin{aligned}
C_T &= C_1 + C_2 = C_1 + \frac{7}{4} C_1 = \frac{11}{4} C_1 \\
k_1 &= \frac{C_1}{C_T} = \frac{C_1}{\frac{7}{11} C_1} = \frac{4}{11} \\
k_2 &= \frac{C_2}{C_T} = \frac{\frac{7}{4} C_1}{\frac{7}{11} C_1} = \frac{7}{11}
\end{aligned}
$$

By using equation (3.2), and disregarding the parasitic capacitances, we get

$$
V_{fg_1} = \sum_{i=0}^{1} k_i V_i = k_1 V_1 + k_2 V_2 = \frac{4}{11} V_{dd} + \frac{7}{11}\frac{3}{14} V_{dd} = V_{dd}/2
$$

and we can see that the first switching point at $\frac{3}{14}V_{dd}$ is in fact shifted to $V_{dd}/2$. The other transition points can be developed in a similar manner.

For the last stage of the conversion, we will first look at the transition from $V_{MV} = \frac{2}{7}V_{dd}$ to $V_{MV} = \frac{3}{7}V_{dd}$. Here the MSB signal is given as $V_{B_2} = V_{dd}$ and the MSB-1 signal is given as $V_{B_1} = V_{dd}$, according to Figure 4.15. The switching point voltage for this transition is given as $V_{MV} = 1/14 \cdot V_{dd}$. For the second equation–we only need two since the capacitance for the MSB-1 signal input can be set freely–we look at the transition from $V_{MV} = 0V$ to $V_{MV} = \frac{1}{7}V_{dd}$. The MSB signal is given as $V_{B_2} = V_{dd}$ with the MSB-1 signal given as $V_{B_1} = 0V$ for this transition. Here the switching point voltage is given as $V_{MV} = \frac{5}{14}V_{dd}$. We then set this into equation (4.15), and we get

$$\begin{aligned}
\text{I:}\quad & k_3V_{B_1} + k_4V_{B_2} + k_5V_{MV} = \frac{V_{dd}}{2}(k_3 + k_4 + k_5) \\
& k_3V_{dd} + k_4V_{dd} + k_5\frac{1}{14}V_{dd} = \frac{V_{dd}}{2}k_3 + \frac{V_{dd}}{2}k_4 + \frac{V_{dd}}{2}k_5 \\
& \frac{1}{2}k_3 + \frac{1}{2}k_4 = (\frac{1}{2} - \frac{1}{14})k_5 \\
\text{II:}\quad & k_3V_{B_1} + k_4V_{B_2} + k_5V_{MV} = \frac{V_{dd}}{2}(k_3 + k_4 + k_5) \\
& k_3 \cdot 0V + k_4V_{dd} + k_5\frac{5}{14}V_{dd} = \frac{V_{dd}}{2}k_3 + \frac{V_{dd}}{2}k_4 + \frac{V_{dd}}{2}k_5 \\
& \frac{1}{2}k_4 - \frac{1}{2}k_3 = (\frac{1}{2} - \frac{5}{14})k_5 \\
\text{I:}\quad & k_5 = \frac{7}{6}(k_3 + k_4) \\
\text{II:}\quad & \frac{1}{2}(k_4 - k_3) = \frac{1}{7}k_5 \\
& \frac{1}{2}(k_4 - k_3) = \frac{1}{6}(k_3 + k_4) \\
& k_4 = 2k_3 \Leftrightarrow C_4 = 2C_3 \\
\text{I:}\quad & k_5 = \frac{7}{6}(k_3 + 2k_3) \\
& k_5 = \frac{7}{2}k_3 \Leftrightarrow C_5 = \frac{7}{2}C_3
\end{aligned}$$

This gives the radix eight MVBC the following set of capacitor values

$$C_0 = C_1 = C_3 = C_{min} \tag{4.16}$$

$$C_2 = \frac{7}{4}C_1 \tag{4.17}$$

$$C_4 = 2C_3 \tag{4.18}$$

$$C_5 = \frac{7}{2}C_3 \tag{4.19}$$

where C_{min} can be set arbitrarily, although it is usually set to the minimum size allowed by the process in question in order to minimize the area used by the circuit.

The procedure of shifting the transition points to $V_{dd}/2$ holds for even higher radices. For the nth bit stage, the $n - 1$ former bits are added or subtracted from the floating-gate in such a manner as to shift the transition points of the MV signal to $V_{dd}/2$. Eventually, however, we are bound to run into fan-out problems with higher radices. With higher radices, we also get a large amounts of input coupling capacitors that all have to be matched. The size of the MV signal input coupling capacitor in the LSB stage will also be a problem. For a radix eight circuit, however, none of these issues presents a problem.

4.6 Multiple-Valued Full-Adder

The FGUVMOS multiple-valued full-adder, which can be seen in Figure 4.16, takes two MV input signals, V_A and V_B. Since this circuit is a full-adder, as opposed to a half-adder, it also takes a carry-in input, $V_{C_{in}}$. This carry-in input is a binary signal, causing the circuit to have a mixed radix input. The output is the arithmetic sum, V_S, of the two multiple-valued signals, as well as the binary carry-in signal, and also has the same radix as the MV input signals. Since the output sum signal has the same radix as the MV input signals, there is also a need for a carry-out signal, $V_{C_{out}}$. As with the carry-in signal, this signal is also binary.

The carry-out signal can be coupled to the carry-in input of another full-adder to form a $(log_2 r \cdot m)$-bit ripple adder, where r is the radix as given in equation (2.4) and m is the number of full-adder stages. The amount of stages–and thus the delay caused by cascading–is less than for a standard binary ripple adder, assuming $r > 2$.

The full-adder has been simulated and verified for the complete set of possible combinations, which is $8 * 8 * 2 = 128$. However, only a selected value range, given by $B = 3;\ \ A = 0 \rightarrow 7;\ \ C_i = 0 \rightarrow 1$, is shown to better illustrate the operating function of the full-adder. The three topmost binary signals shown in Figure 4.17 are the signal inputs to a BMVC which generates the multiple-valued input V_A. The V_S multiple-valued output is feed into an MVBC and the resulting binary outputs are shown as the three last signals. Figure 4.18 shows the multiple-valued inputs and outputs of the full-adder, as well as the internal signal, V_Z, and the binary carry-in and carry-out signals.

The input signals, including the carry-in signal, are added together in the first

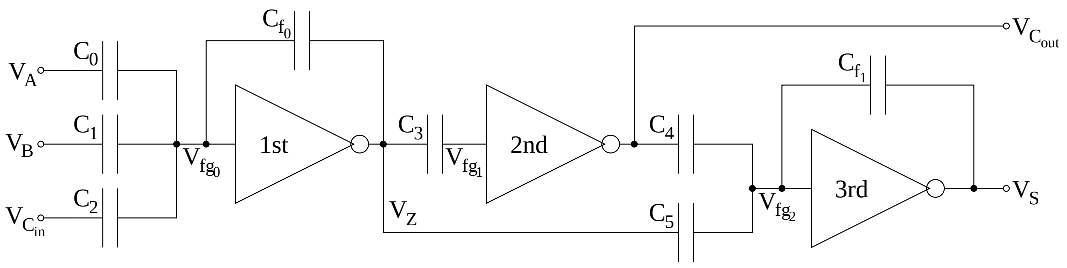


Figure 4.16: *FGUVMOS MV full-adder schematic. The circuit has two multiple-valued inputs, V_A and V_B, and one binary input signal, which is the carry-in signal $V_{C_{in}}$. The outputs consists of the multiple-valued sum, V_S, and the binary carry-out signal, $V_{C_{out}}$. The full-adder consists of three stages. The first stage adds the input signals together, and we get a higher radix signal on the internal node, V_Z. The next stage generates the carry-out signal, $V_{C_{out}}$. The last stage converts the internal signal, V_Z, to a lower radix signal, V_S, by removing the carry-out portion of the signal. The output sum signal has the same radix as the multiple-valued input signals.*

stage of the full-adder. This addition operation is defined as

$$\overline{Z} = A + B + C_i \tag{4.20}$$

due to the inverting property of the multiple-valued inverter. In general, the inverted of an MV signal, with a value set as given in equation (2.1), can be found by taking the complement. Thus we have $\overline{x} = (r-1) - x$ and in order to find the non-inverted function, we first have to determine the radix of the internal signal, Z. The radix can be found by examining the maximum values that can be assumed by the input signals, which, for an MV signal on the form described above, is $r-1$. Thus the maximum value that can be assumed by the output signal from the first stage is

$$\begin{aligned}&\text{max}(\overline{Z}) = \text{max}(A) + \text{max}(B) + \text{max}(C_i) = \\ &(r-1) + (r-1) + 1 = 2r - 1\end{aligned}$$

which gives us a radix of $2r$ for the internal sum signal Z. The non-inverted output then becomes

$$Z = (2r-1) - (A + B + C_i) \tag{4.21}$$

where r is the radix of the MV input signals, which are assumed to be of equal radix. In order to find the coupling capacitances, we first need to define the addition operation in terms of input and output voltages. The input voltages to the

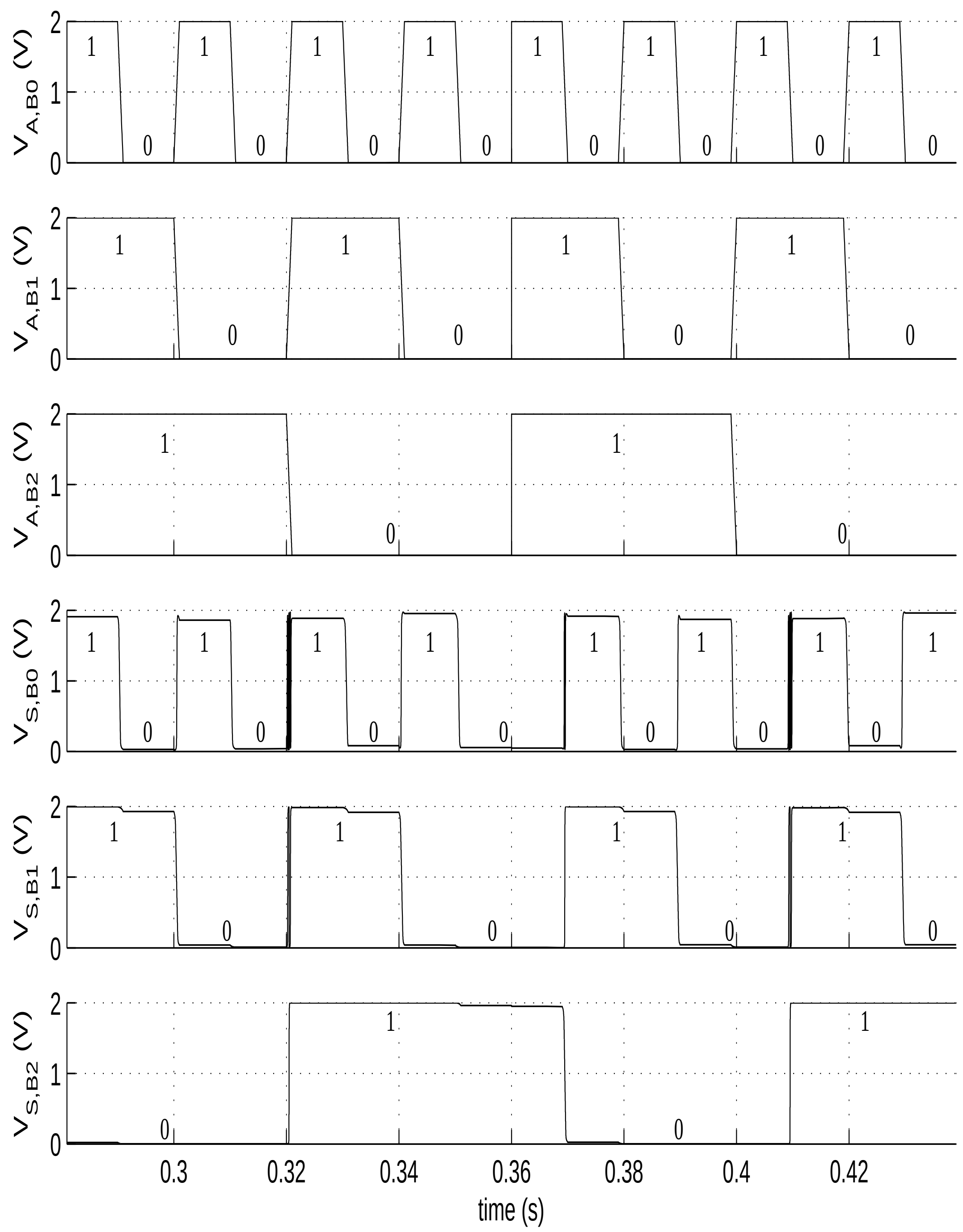


Figure 4.17: *A simulation of the response of the FGUVMOS MV full-adder showing the binary signals of the input and output converters. The V_B signal is held fixed at $B = 3$. This figure only shows the binary signal inputs to the BMVC that generates the V_A input signal, and the binary output signals from the MVBC, whose input is the V_S signal. The carry-in and carry-out signals are shown in the next figure, although it should be noted that they would have to be inverted before they can be used in conjunction with the binary signals shown here.*

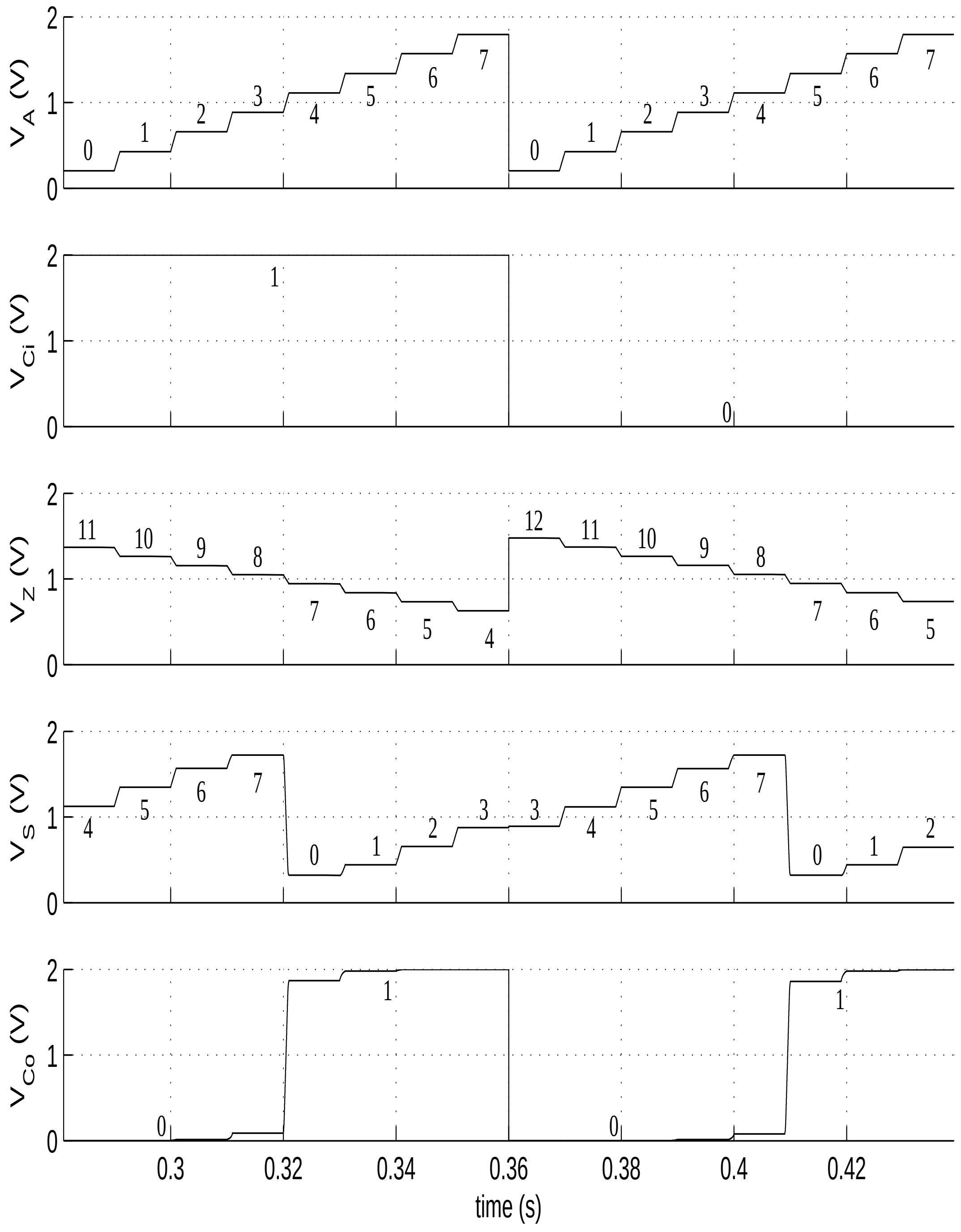


Figure 4.18: *A simulation of the response of the FGUVMOS MV full-adder showing the MV signals. The V_B signal is held fixed at $B = 3$. This figure shows the MV input and output signals, V_A and V_S. The internal radix $2r$ signal, V_Z, is also included. The binary carry signals, $V_{C_{in}}$ and $V_{C_{out}}$, are shown as well.*

full-adder and the output voltage of the first stage, V_Z, can be expressed in terms of the logic values in the following manner

$$V_A = \frac{A}{r-1}V_{dd} \Leftrightarrow A = \frac{r-1}{V_{dd}}V_A \tag{4.22}$$

$$V_B = \frac{B}{r-1}V_{dd} \Leftrightarrow B = \frac{r-1}{V_{dd}}V_B \tag{4.23}$$

$$V_{C_{in}} = C_i V_{dd} \Leftrightarrow C_i = \frac{V_{C_{in}}}{V_{dd}} \tag{4.24}$$

$$V_Z = \frac{Z}{2r-1}V_{dd} \Leftrightarrow Z = \frac{2r-1}{V_{dd}}V_Z \tag{4.25}$$

where r is the radix of the MV input signals. Equation (4.21) can then be expressed in terms of voltages instead of logic levels by using equations (4.22) - (4.25) as follows

$$\begin{aligned} Z &= (2r-1) - (A+B+C_i) \\ \frac{2r-1}{V_{dd}}V_Z &= (2r-1) - \left(\frac{r-1}{V_{dd}}V_A + \frac{r-1}{V_{dd}}V_B + \frac{1}{V_{dd}}V_{C_{in}}\right) \\ V_Z &= V_{dd} - \left(\frac{r-1}{2r-1}V_A + \frac{r-1}{2r-1}V_B + \frac{1}{2r-1}V_{C_{in}}\right) \end{aligned} \tag{4.26}$$

By using a multiple-valued inverter, we can implement equation (4.26). We could find the capacitive division relationships by constructing a series of equations–based on a truth table for the adder–to find the unknown division factors. However, we have equation (4.26), with separate terms for each voltage input, which we can use instead. By setting this equation equal to equation (4.3), we get

$$V_Z = V_{out}$$

$$V_{dd} - \left(\frac{r-1}{2r-1}V_A + \frac{r-1}{2r-1}V_B + \frac{1}{2r-1}V_{C_{in}}\right) = \frac{1+\frac{\sum_{i=0}^{2}k_i}{k_f}}{2}V_{dd} - \left(\frac{k_0}{k_f}V_A + \frac{k_1}{k_f}V_B + \frac{k_2}{k_f}V_{C_{in}}\right)$$

Assume we have unity gain (we will check that this is correct later on), we then get

$$\frac{1}{2r-1}[(r-1)V_A + (r-1)V_B + V_{C_{in}}] = \frac{k_2}{k_f}(\frac{k_0}{k_2}V_A + \frac{k_1}{k_2}V_B + V_{C_{in}})$$

and we can then see which terms corresponds. We then set the corresponding terms equal to each other

$$\frac{k_0}{k_2} = (r-1) \Leftrightarrow k_0 = (r-1)k_2 \Leftrightarrow C_0 = (r-1)C_2$$

$$\frac{k_1}{k_2} = (r-1) \Leftrightarrow k_1 = (r-1)k_2 \Leftrightarrow C_1 = (r-1)C_2$$
$$\frac{k_2}{k_f} = \frac{1}{2r-1} \Leftrightarrow k_f = (2r-1)k_2 \Leftrightarrow C_f = (2r-1)C_2$$

from which we get $\sum_{i=0}^{2} k_i = (2r-1)C_2$ and we have unity gain according to equation (4.2). The C_2 capacitor can be set freely and is usually set to the minimum capacitor size available for the process in question, $C_2 = C_{min}$.

We can see from the first stage in the full-adder that mixed radix input is possible and that the output can have a higher radix than any of the input signals. In general, it is possible to mix any radix desired on the input of the binary and multiple-valued inverter. The output radix of the binary inverter will always be binary, while the output radix of the multiple-valued inverter will either be lower, equal or higher than any of the input radices. The output radix of a FGUVMOS fundamental building block is given as

$$r_{out} = 1 + \max\left(1, \frac{C_f}{C_0}\cdot(r_0-1), ..., \frac{C_f}{C_i}\cdot(r_i-1), ..., \frac{C_f}{C_{n-1}}\cdot(r_{n-1}-1)\right) \tag{4.27}$$

assuming an MV signal on the form given by equation (2.1). The number of inputs is n and r_i is the radix associated with the ith input. Since the binary inverter does not have a feedback capacitor, $r_{out} = 2$ and the output has a binary radix. For the first stage in the full-adder, the output radix is given as $r_{out} = 1 + \frac{2r-1\cdot C_2}{r-1\cdot C_2}(r-1) = 2r$, which coincides with what was found earlier.

The carry-out signal, C_o, is generated in the second stage of the full-adder. This signal should be generated when $\overline{Z} > (r-1)$ and conversely should not be generated when $\overline{Z} < (r-1)$. However, the internal signal has a radix of $2r$, and the expression for the carry-out signal generation should take this into consideration. By also accounting for the inverting property of the first stage, we get

$$C_o = \begin{cases} 0 & \text{if} \quad Z > \frac{2r-1}{2} \\ 1 & \text{if} \quad Z < \frac{2r-1}{2} \end{cases} \tag{4.28}$$

for when the carry-out signal should be generated in terms of logical values. When we take into account equation (4.25), we can then express this in terms of voltages

$$V_{C_{out}} = \begin{cases} 0 & \text{if} \quad V_Z > \frac{V_{dd}}{2} \\ V_{dd} & \text{if} \quad V_Z < \frac{V_{dd}}{2} \end{cases} \tag{4.29}$$

and the carry-out signal can therefore be converted directly using a binary inverter and the capacitor, C_3, can be chosen freely. The FGUVMOS binary inverter in this stage can be exchanged with a standard static CMOS binary inverter, however, the benefits from the UV-programming will then be lost.

The output sum signal, called S, is produced in the last stage of the full-adder. This output signal has the same radix as the multiple-valued input signals A and B. The output sum signal can be generated by adding all of the input signals together and then remove the portion that constitutes the carry-out signal. The carry-out signal has a logical value of either 1 or 0 since it has a binary radix. However, the value signified is either r or 0. The output sum signal can therefore be defined in the following manner

$$S = (A + B + C_i) - rC_o \tag{4.30}$$

The voltage output signals from the full-adder, V_S and $V_{C_{out}}$, can be expressed in terms of logical values in the following manner

$$V_S = \frac{S}{r-1}V_{dd} \Leftrightarrow S = \frac{r-1}{V_{dd}}V_S \tag{4.31}$$

$$V_{C_{out}} = C_o V_{dd} \Leftrightarrow C_o = \frac{V_{C_{out}}}{V_{dd}} \tag{4.32}$$

since the output sum signal should have the same radix as the multiple-valued input signals. When we also take into account equations (4.22) - (4.24), we can express equation (4.30) in terms of voltages

$$\begin{aligned} \frac{r-1}{V_{dd}}V_S &= \left(\frac{r-1}{V_{dd}}V_A + \frac{r-1}{V_{dd}}V_B + \frac{V_{C_{in}}}{V_{dd}}\right) - \frac{r}{V_{dd}}V_{C_{out}} \\ V_S &= \left(V_A + V_B + \frac{1}{r-1}V_{C_{in}}\right) - \frac{r}{r-1}V_{C_{out}} \end{aligned} \tag{4.33}$$

The expression for the output sum signal can be implemented using a multiple-valued inverter. The capacitive division relationships for the multiple-valued inverter can be found by setting equation (4.33) equal to equation (4.3)

$$\begin{aligned} V_S &= V_{out} \\ V_S &= \left(\frac{1+A_v}{2}\right)V_{dd} - \left(\frac{k_4}{k_f}V_{C_{out}} + \frac{k_5}{k_f}V_Z\right) \end{aligned}$$

and we can substitute V_Z as defined in equation (4.26) and V_S as defined in equation (4.33)

$$V_S = \left(\frac{1+A_v}{2}\right)V_{dd} - \frac{k_4}{k_f}V_{C_{out}} - \frac{k_5}{k_f}\left(V_{dd} - \frac{r-1}{2r-1}V_A - \frac{r-1}{2r-1}V_B - \frac{1}{2r-1}V_{C_{in}}\right)$$

$$\Updownarrow$$

$$\left(V_A + V_B + \frac{1}{r-1}V_{C_{in}}\right) - \frac{r}{r-1}V_{C_{out}}$$
$$= \left(\frac{1+A_v}{2} - \frac{k_5}{k_f}\right)V_{dd} + \frac{k_5}{k_f}\frac{r-1}{2r-1}\left(V_A + V_B + \frac{1}{r-1}V_{C_{in}}\right) - \frac{k_4}{k_f}V_{C_{out}}$$

The corresponding terms on either side then becomes clear. There is, however, one exception. The term proceeding V_{dd} on the right-hand side does not have any equivalent term on the left side. This right-hand term will then have to resolve to zero in order for both sides of the equation to match up. Since this term includes all of the unknown variables, we will try to resolve all of the other terms first by setting them equal to each other

$$\frac{k_4}{k_f} = \frac{r}{r-1} \Leftrightarrow k_4 = \frac{r}{r-1}k_f \Leftrightarrow C_4 = \frac{r}{r-1}C_f \tag{4.34}$$

$$\frac{k_5}{k_f}\frac{r-1}{2r-1} = 1 \Leftrightarrow k_5 = \frac{2r-1}{r-1}k_f \Leftrightarrow C_5 = \frac{2r-1}{r-1}C_f \tag{4.35}$$

and the last capacitor value, C_f, can be set freely. We can now examine the last unresolved term–the term preceding V_{dd} on the right-hand side–which must resolve to zero. This term can be resolved, by substituting A_v as defined in equation (4.2)

$$\frac{1+A_v}{2} - \frac{k_5}{k_f} = \frac{1 + \frac{k_4}{k_f} - \frac{k_5}{k_f}}{2} = \frac{\frac{r-1}{r-1} + \frac{r}{r-1} - \frac{2r-1}{r-1}}{2} = 0$$

and we see that the term resolves to zero and all the terms of the equation match up.

One last noteworthy point is the amplification of the last stage. Since we want the multiple-valued output sum signal to be of radix r, and the input signals to the last stage, V_Z and $V_{C_{out}}$, were not of radix r, then clearly a non-unity amplification is needed. We can express this statement in the following manner

$$\frac{S}{\max(S)}A_v = \frac{S}{r-1}$$
$$\Updownarrow$$
$$A_v = \frac{\max(S)}{r-1} = \frac{\max(Z) + r\cdot\max(C_o)}{r-1} = \frac{(r-1)+(r-1)+1+r\cdot 1}{r-1} = \frac{3r-1}{r-1}$$

and we can see that the sum of the input signals on the floating-gate on the last stage of the full-adder has a radix of $3r$. Thus a gain of $A_v = \frac{3r-1}{r-1}$ is necessary to get an output signal of radix r. We can also see that the gain for the last stage, as defined by equation (4.2), is $A_v = \frac{\sum_{i=4}^{5} k_i}{k_f} = \frac{3r-1}{r-1}$ by using equations (4.34) - (4.35).

The size of the voltage input coupling capacitors are radix dependent due to the binary carry-in signal. If the carry-in signal had been multiple-valued, all of the input coupling capacitors would have been of equal size and they would all have been radix independent. The carry-out signal generating stage is radix independent. And the last stage also only has minimal changes in capacitor values with an increase in the radix, suggesting that the second cost function in section2.3, equation (2.3), might be appropriate for this circuit. One hurdle for higher radix in this circuit is the radix $2r$ intermittent signal. A different topology might avoid this higher intermittent radix signal.

4.7 Summary

We started this chapter with discussing which radices were used for the circuits in this thesis, and what tools were used to measure and simulate them. We then presented the first of the fundamental building blocks, the FGUVMOS binary inverter. We showed the equation for the output voltage and discussed ways of improving signal detection. We then looked at the other fundamental building block, the FGUVMOS multiple-valued inverter. We discussed the problem of compression of the voltage levels near the rails and how the tunable gain provided by the feedback capacitor deals with this issue. We then developed equations for the voltage output and the tunable gain. We then moved on to the BMVC. We discussed the need for converting between binary and multiple-valued signals. We saw that the equation for the voltage output was similar to the multiple-valued inverter and several measurements on a fabricated chip were shown. We then discussed the gain of the BMVC and the voltage limiting operation it performs. At last we developed the capacitor relationships for the BMVC. We then presented the complement of the BMVC, namely the MVBC. We discussed how the binary inverter preforms a one-bit AD operation. Next, we showed some simulations for the circuit. We looked at switching point sets and how the voltage on the floating-gate was effectively shifted using the capacitive division relationships. We subsequently found the capacitor values for the circuit and discussed how this method of conversion would scale. Lastly, we presented the FGUVMOS multiple-valued adder. We show some simulations and discussed how to find the inverted of a FGUVMOS multiple-valued circuit and which output radix we would get with a mixed radix input. We then developed the voltage equations for every stage in the circuit, as well as the capacitor sizes for each stage.

Chapter 5

Conclusion And Further Work

5.1 Summary

The main topics in this thesis have been the multiple-input common-gate FGUVMOS transistor and the design of multiple-valued logic circuits using this transistor. We have focused our attention on the common-gate programming method for the FGUVMOS transistor and its characteristics. We have also seen how the FGUVMOS transistor is an ideal candidate for constructing multiple-valued logic circuits using the voltage-mode approach.

5.1.1 Common-Gate FGUVMOS Transistor

We have presented a new UV-programming technique for initializing the FGUVMOS transistor by using only the nMOS transistor in the programming step. This solved the problems associated with the split-gate programming technique due to the workfunction differences between the nMOS and pMOS transistor. There are, however, some drawbacks to the new programming technique. We loose the ability to arbitrarily set the current level, meaning ultra-low power applications are no longer a viable candidate for the utilization of the common-gate FGUVMOS transistor.

The common-gate programming method also comes with several advantages. A reduction in programming time might be forthcoming, since only the nMOS transistor is involved in the programming step. There is also a saving in areas cost since the floating-gate is shared by the nMOS and pMOS transistor–cutting the coupling capacitors needed in half.

5.1.2 Multiple-Valued Logic Application

We have also looked at a new area of application for the FGUVMOS transistor, namely multiple-valued logic. The main design parameter of the FGUVMOS transistor–the capacitive division relationships between the coupling capacitors to the floating-gate–is well suited for designing voltage-mode multiple-valued logic circuits. Along with the multiple voltage inputs, a tunable gain which allows for conversion between radices and the new programming technique, they make the common-mode FGUVMOS transistor an ideal candidate for multiple-valued logic.

The FGUVMOS MVL circuits have, unfortunately, one major drawback. There is no inherent signal restoration built into the FGUVMOS devices. Thus signal restoration circuits have to be employed at periodic intervals–depending on process variations, noise, radix and similar characteristics–to ensure robust circuit operation. This drawback is shared with other multiple-valued logic circuits using different design approaches, since there currently exists no physical multi-state device for solid-state integrated circuits.

5.1.3 Layout Issues

The topic of proper layout of the transistors and capacitors, and the effect it has on circuit performance both in the operational and programming mode, has been thoroughly discussed in this thesis. These discussions are a result of problems with programming the circuits on the fabricated chip. This is also the reason why measurements are only presented for the BMVC.

While the exact cause of the problems is unknown, there are some strong indications. First of all, since we were able to get measurements from the BMVC, this suggest that the problem is not with the programming method. The problems more likely resides with the regular shaped transistors, greatly increasing the significance of the parasitic UV-activated conductances. Another relevant issue might be the internal contacts on the floating-gate, especially those that were in proximity to the transistor. This might have lead to some gate leakage, however, nothing was found in the available literature to support this assumption.

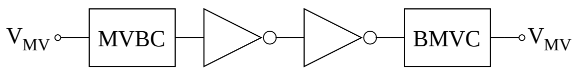


Figure 5.1: *Straight forward signal restoration scheme. The multiple-valued signal is first converted to binary by an MVBC and is then converted back to a multiple-valued signal by the BMVC. An extra digital buffer can be added to ensure rail-to-rail binary signals.*

5.2 Further work

A signal restoration scheme is needed to ensure robust circuit operation, especially for cascaded systems. Such a scheme is needed because there is no inherent signal restoration built into the FGUVMOS transistor, since the FGUVMOS transistor is not a multi-state device. There is one straight forward way to regenerate the signal. We can convert the MV signal to binary then back to a multiple-valued signal using a chain of circuits consisting of an MVBC and a BMVC. To ensure a rail-to-rail binary signal after the conversion preformed by the MVBC, one might want to include a binary signal buffer. The entire setup can be seen in Figure 5.1. Although such a signal restoration scheme will work, the area cost is significant. It should therefore only be used at periodic intervals dependent on process variations, noise, radix and similar characteristics.

Another way of solving the signal restoration problem, might be to embed a regeneration capability into the FGUVMOS multiple-valued inverter itself. This might be done by adding several different power rails to the circuit. These extra power rails can be generated locally by stacking transistor with their gate and drain connected together. The appropriate power rail can then be selected and shunted to the voltage output using a pass-transistor. The overall suggested topology is depicted in Figure 5.2. The pass transistor selection circuit consists of several FGUVMOS binary transistors were only one pass transistor can be selected at any given time. This is done by shifting the switching threshold of the binary inverters depending on the output from the FGUVMOS multiple-valued inverter. Again, there is a significant increase in area cost, and this signal restoration scheme should only be used at intermittent intervals in a cascaded system.

While we showed a method for determining the capacitor relationships through linear algebra, no generalized method for designing FGUVMOS circuits (as well as FGUVMOS MVL circuits) exists. A numerical method for selecting the capacitor values–maybe also taking into consideration the gate capacitance–is a starting point. However, also the topology of the circuits needs to be taken into consideration. There is no method for determining which topology a FGUVMOS circuit should have to performed a given function.

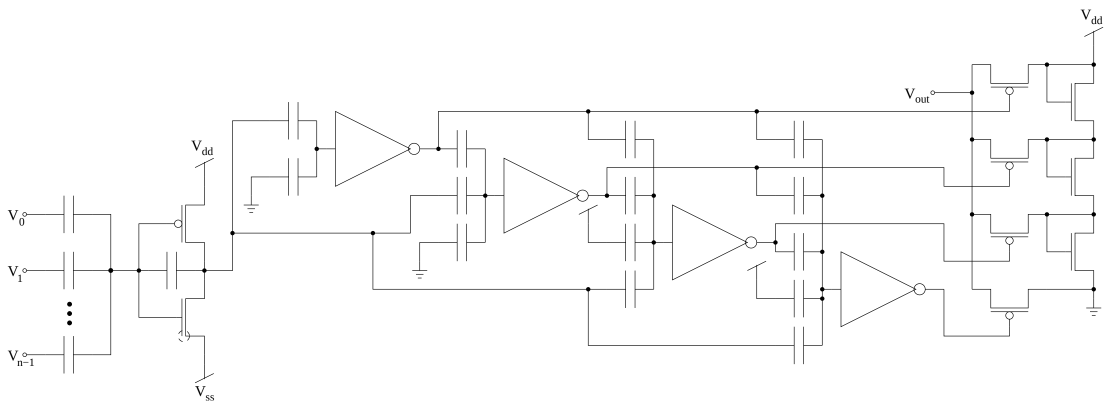


Figure 5.2: *Suggested topology for an FGUVMOS multiple-valued inverter with embedded signal restoration for a radix four multiple-valued logic system. The signal is regenerated by selecting one of several power rails, which are connected to the output by a pass transistor. The problem lies in selecting the appropriate capacitor ratios for the FGUVMOS binary inverters.*

There is a need to investigate the possible causes for gate leakage in the FGU-VMOS transistors. Several layout techniques should be explored to see what the consequences are not only to gate leakage, but also programming time. To be on the safe side, FGUVMOS transistors should have no internal contacts on the floating-gate and they should have a ring topology.

The FGUVMOS MVL circuits presented in this thesis preform only a small selection of the functions necessary to form a comprehensive logic system. One should explore further the relation between Post algebra and FGUVMOS MVL. Also, a more in depth look at the various multiple-valued logic operations[42] should be taken, and equivalent FGUVMOS MVL circuits should be constructed. A universal logic gate–such as an FGUVMOS MVL multiplexer–should also be constructed. A good starting point might be the pass transistor selection circuit in Figure 5.2.

Bibliography

[1] S. ichi Takagi, T. Mizuno, T. Tezuka and A. Kurobe. **Strained-Si-On-Insulator (Strained-SOI) MOSFETs – Concepts, Structures And Device Characteristics**. *IEICE Transactions On Electronics*, vol. E84-C, no. 8, pp. 1043–1050, 2001.

[2] N. Weste and K. Eshraghian. **Principles Of CMOS VLSI Design**. Addison-Wesley, 2nd ed., 1994.

[3] Y. Berg and T. S. Lande. **Programmable Floating-Gate MOS Logic For Low-power Operation**. *Proceedings of the 1997 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 3, pp. 1792–1795, June 1997.

[4] C. H. V. Berkel, M. B. Josephs and S. M. Nowick. **Scanning the Technology: Applications of Asynchronous Circuits**. *Proceedings of the IEEE*, vol. 87, no. 2, pp. 223–233, February 1999.

[5] H. van Gageldonk, K. van Berkel, A. Peeters, D. Baumann, D. Gloor and G. Stegmann. **An Asynchronous Low-Power 80C51 Microcontroller**. *Fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 96–107, April 1998.

[6] K. C. Smith. **The Prospects for Multivalued Logic: A Technology and Applications View**. *IEEE Transactions on Computers*, vol. 30, no. 9, pp. 619–634, September 1981.

[7] C. Patel, S. Ogitani, P. Kohl, K. Martin and J. Meindl. **An Analysis of the Gap between PWB Technology and Chip I/O Interconnect Technology, and a New Wafer-Level Batch Packaging Concept**. *Proceedings of the 1999 International Symposium on Microelectronics*, pp. 611–619, 1999.

[8] T. T. Dao. **Recent Multi-Valued Circuits**. *Proccedings of the 1981 IEEE Compcon*, pp. 194–203, January 1981.

[9] F. Masuoka, R. Shirota and K. Sakui. **Reviews And Prospects Of Non-Volatile Semiconductor Memories**. *IEICE transactions*, vol. 74, no. 4,

pp. 868–874, 1991.

[10] S. Lai. **Flash Memories: Where We Were And Where We Are Going**. *IEEE International Electron Devices Meeting, San Francisco*, pp. 971–974, 1998.

[11] T. Shibata and T. Ohmi. **A Functional MOS Transistor Featuring Gate-Level Weighted Sum and Threshold Operations**. *IEEE Transactions on Electron Devices*, vol. 39, no. 6, pp. 1444–1455, June 1992.

[12] A. Hastings. **The Art Of Analog Layout**. Prentice Hall, 2001.

[13] D. J. Frank, R. H. Dennard, E. Nowak, P. M. Solomon, Y. Taur and H.-S. P. Wong. **Device Scaling Limits Of Si MOSFETs And Their Application Dependencies**. *Proceedings of the IEEE*, vol. 89, no. 3, pp. 259–288, March 2001.

[14] E. L. Post. **Introduction to a General Theory of Elementary Propositions**. *American Journal of Mathematics*, vol. 43, pp. 163–185, 1921.

[15] S. L. Hurst. **Multiple-Valued Logic - Its Status And Its Future**. *IEEE Transactions On Computers*, vol. 33, no. 12, pp. 1160–1179, December 1984.

[16] H. G. Kerkhoff and M. L. Tervoert. **Multiple-Valued Logic Charge-Coupled Devices**. *IEEE Transactions on Computers*, vol. 30, no. 9, pp. 644–652, 1981.

[17] T. T. Dao, E. J. McClusky and L. K. Russell. **Multivalued Integrated Injection Logic**. *IEEE Transactions on Computers*, vol. 26, no. 12, pp. 1233–1241, 1977.

[18] E. J. McClusky. **Logic Design of MOS Ternary Logic**. *IEEE Proceedings of the 10th International Symposium on Multiple-Valued Logic*, pp. 1–5, June 1980.

[19] E. Dubrova. **Multiple-Valued Logic In VLSI: Challenges And Opportunities**. *Proceedings of NORCHIP 99*, Oslo, Norway, 1999.

[20] D. Kahng and S. M. Sze. **A Floating-Gate And Its Application To Memory Devices**. *The Bell System Technical Journal*, vol. 46, no. 4, pp. 1288–1295, 1967.

[21] L. R. Carley. **Trimming Analog Circuits Using Floating-Gate Analog MOS Memory**. *IEEE Journal of Solid-State Circuits*, vol. 24, no. 6, pp. 1569–1575, 1989.

[22] C. A. Mead. **Analog VLSI And Neural Systems**. Addison-Wesley, 1989.

[23] M. Holler, S. Tam, H. Castro and R. Benson. **An Electronically Trainable Artificial Neural Network (ETANN) With 10240 'Floating-Gate' Synapses**. *in Proceedings of the 1989 International Conference on Neural Networks*, vol. 2, pp. 191–196, June 1989.

[24] B. A. M. Paul Hasler and C. Diorio. **Floating-Gate Devices: They Are Not Just For Digital Memories Any More**. *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2, pp. 388–391, June 1999.

[25] C. A. Mead and M. Ismail. **Analog VLSI Implementation Of Neural Systems**. Kluwer Academic Publishers, 1989.

[26] A. Thomsen and M. A. Brooke. **A Floating Gate MOSFET With Tunneling Injector Fabricated Using Standard Double Polysilicon CMOS Process**. *IEEE Electron Device Letters*, vol. 12, pp. 111–113, 1991.

[27] K. Hieda, M. Wada, T. Shibata and H. Iizuka. **A New EEPROM Cell With Dual Control Gate Structure**. *Digest of Technical Papers of the 1983 Symposium on VLSI Technology*, pp. 114–115, September 1983.

[28] Y. Berg, Ø. Næss, S. Aunet and M. Høvin. **A Novel Floating-Gate Multiple-Valued Signal To Binary Signal Converter**. *9th International Conference on Electronics, Circuits and Systems*, vol. 2, pp. 575–578, September 2002.

[29] Ø. Næss. **Continuous-Time Filter Design Using Floating Gate Circuits**. Master's thesis, University of Oslo, Department of Informatics, 1999.

[30] E. A. Vittoz. **Analog VLSI Signal Processing: Why, Where And How?** *Journal of VLSI Signal Processing*, vol. 8, pp. 27–44, 1994.

[31] P. Hasler, C. Diorio, B. A. Minch and C. Mead. **Single Transistor Learning Synapse With Long Term Storage**. *Proceedings of the 1995 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 3, pp. 1660–1663, April 1995.

[32] W. P. Millard, Z. K. Kalyjian and A. G. Andreou. **Calibration And Matching Of Floating Gate Devices**. *Proceedings of the 2000 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, pp. 389–392, May 2000.

[33] Y. Berg, T. S. Lande and Øivind Næss. **Programming Floating-Gate Circuits With UV-Activated Conductances**. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 1, pp. 12–19, January 2001.

[34] R. G. Benson and D. A. Kerns. **UV-Activated Conductances Allow For Multiple Time Scale Learning**. *IEEE Transactions on Neural Networks*, vol. 4, no. 3, pp. 434–440, May 1993.

[35] B. G. Streetman and S. Banerjee. **Solid State Electronic Devices**. Prentice Hall, 2000.

[36] M. J. McNutt. **Systematic Capacitance Matching Errors And Corrective Layout Procedures**. *IEEE Journal Of Solid-State Circuits*, vol. 29, no. 5, pp. 611–616, May 1994.

[37] B. A. Minch. **Floating-Gate Techniques For Assessing Mismatch**. *IEEE Symposium on Circuits and Systems*, vol. 4, pp. 385–388, May 2000.

[38] J.-B. Shyu, G. C. Temes and K. Yao. **Random Errors In MOS Capacitors**. *IEEE Journal Of Solid-State Circuits*, vol. 17, no. 6, pp. 1070–1076, December 1982.

[39] J.-B. Shyu, G. C. Temes and F. Krummenacher. **Random Error Effects In Matched MOS Capacitors And Current Sources**. *IEEE Journal Of Solid-State Circuits*, vol. 19, no. 6, pp. 984–955, December 1984.

[40] H. W. Ott. **Noise Reduction Techniques In Electronic Systems**. John Wiley and Sons, 1988.

[41] H. Özdemir, A. Kepkep, B. Pamir, Y. Leblebici and U. Çilingiroğolu. **A Capacitive Threshold-Logic Gate**. *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 1141–1150, August 1996.

[42] K. C. Smith. **Multiple Valued Logic: A Tutorial And Appreciation**. *Computer*, vol. 21, no. 4, pp. 17–27, April 1988.

List of Figures

List of Tables

Appendix A

Simulation Setup

All the simulations were preformed in spectreS inside Cadence. The following series of figures shows the schematics of the circuits that were simulated, followed by the simulation setup for the circuits. The former is necessary to fully appreciate the simulation setup.

All the V_{ss} terminals are for simulation purposes coupled to ground. The programming is simulated by placing a charge on the floating-gate which results in an output of $V_{dd}/2$ when the input is set to $V_{dd}/2$. This is checked in the simulations by using a piecewise linear input signal of $V_{dd}/2$. If the output is also $V_{dd}/2$, then the actual circuits simulation can be preformed correctly.

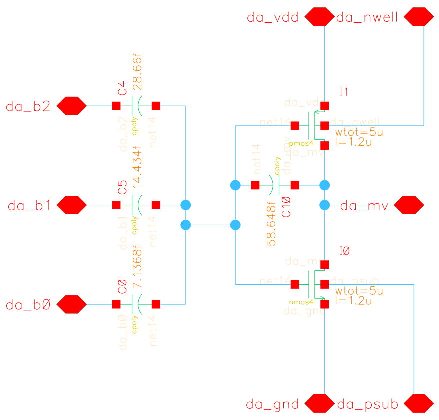


Figure A.1: *BMVC schematic from Cadence. The circuit has three binary inputs,* da_b0 - da_b2*, and one multiple-valued output,* da_mv*. The capacitor values are listed above the capacitors. One should also note that the nMOS and pMOS transistor are of equal size.*

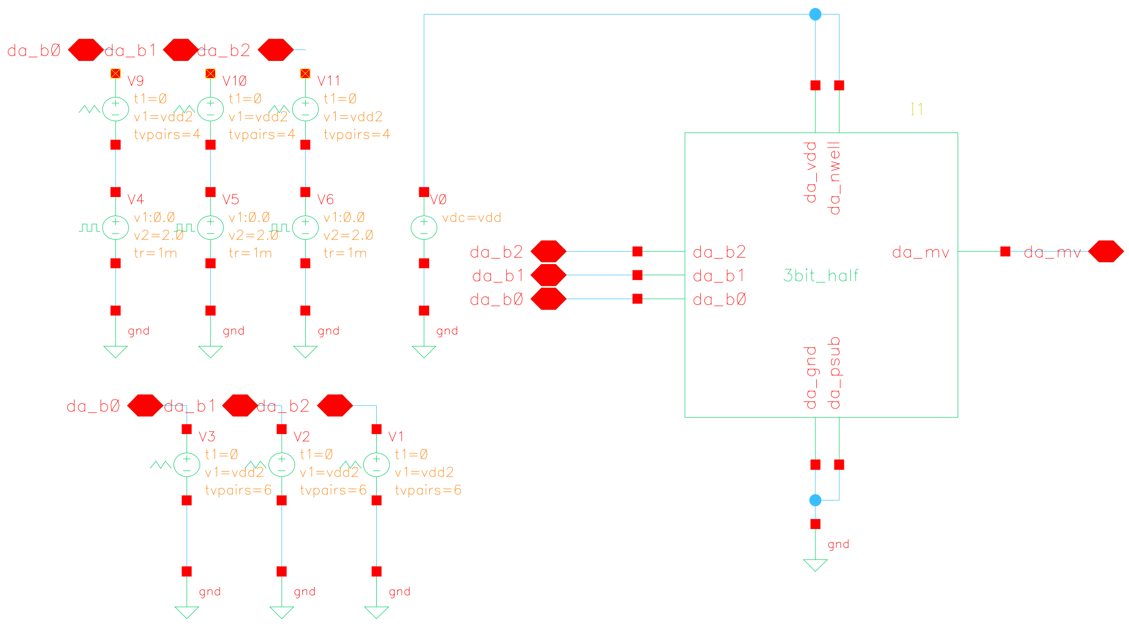


Figure A.2: *BMVC simulation setup from Cadence. Two sets of signal sources are used. One for the linear voltage input simulation (the set at the bottom) and the second for the binary signal inputs (the set at the top). The binary voltage inputs uses the piecewise linear voltage source to check the floating-gate charge.*

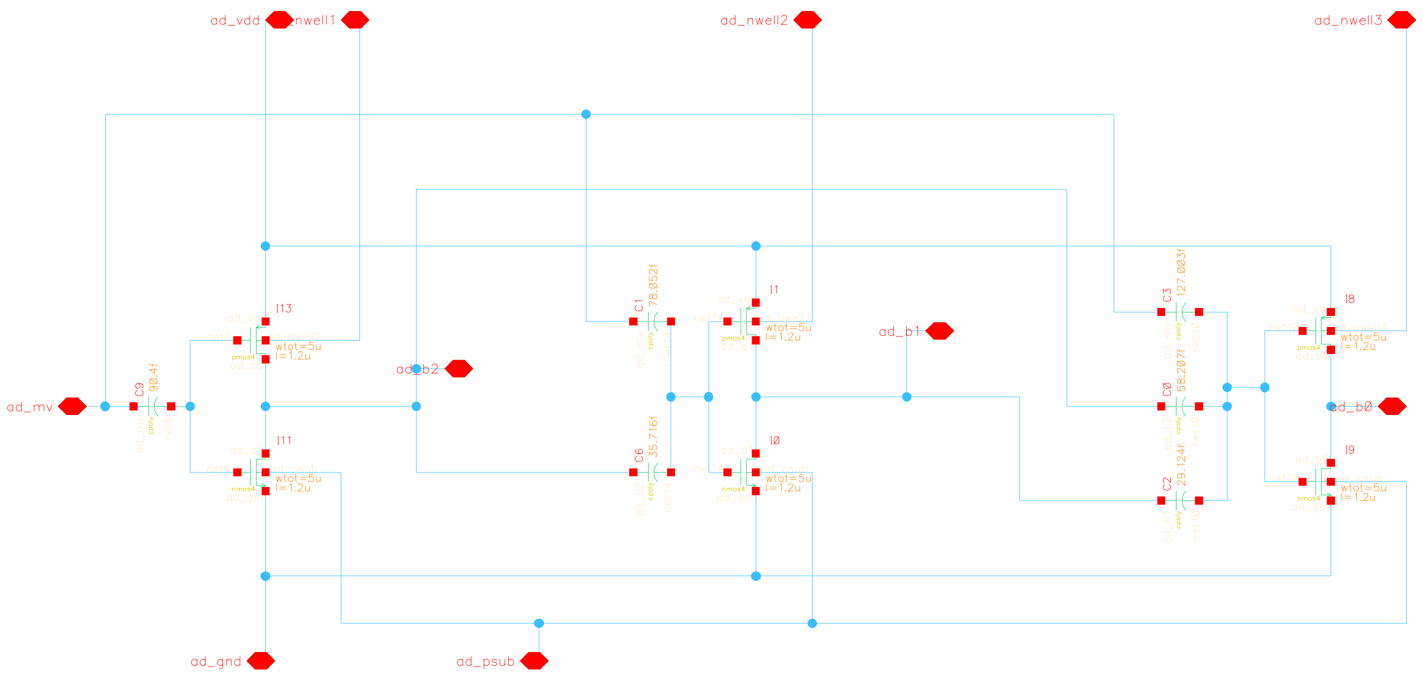


Figure A.3: *MVBC schematic from Cadence. The circuit has one multiple-valued input,* ad_mv*, and three binary outputs,* $ad\_b0 - ad\_b2$*. The capacitor values are listed above the capacitors. One should note that all of the nMOS and pMOS transistors are of equal size.*

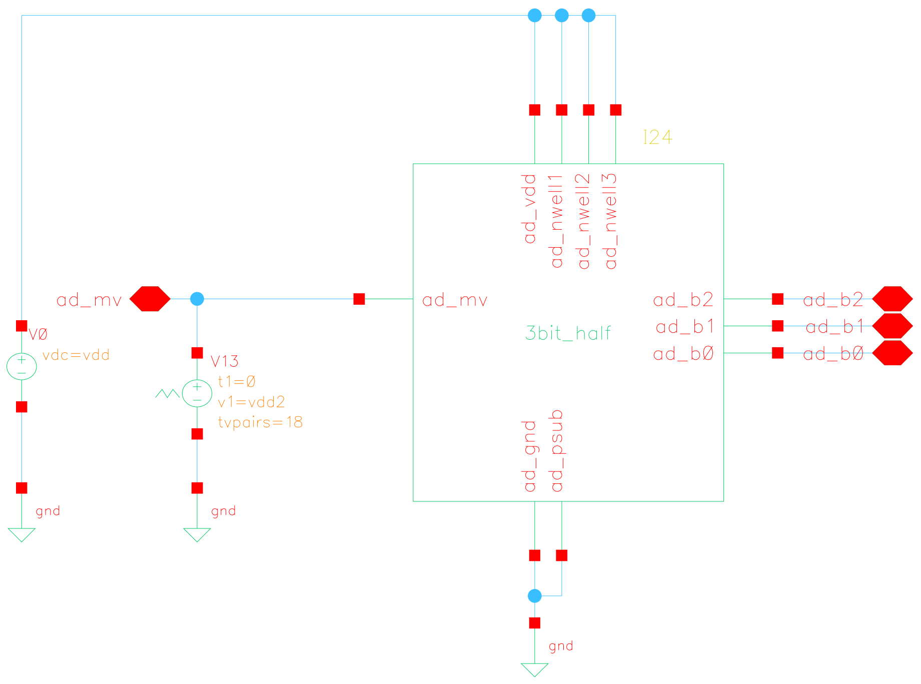


Figure A.4: *MVBC simulation setup from Cadence. The multiple-valued input signal is constructed using a linear piecewise voltage source, which has eighteen steps going from* $0.2V$ *to* $1.8V$*, including the steps necessary to check the floating-gate charge.*

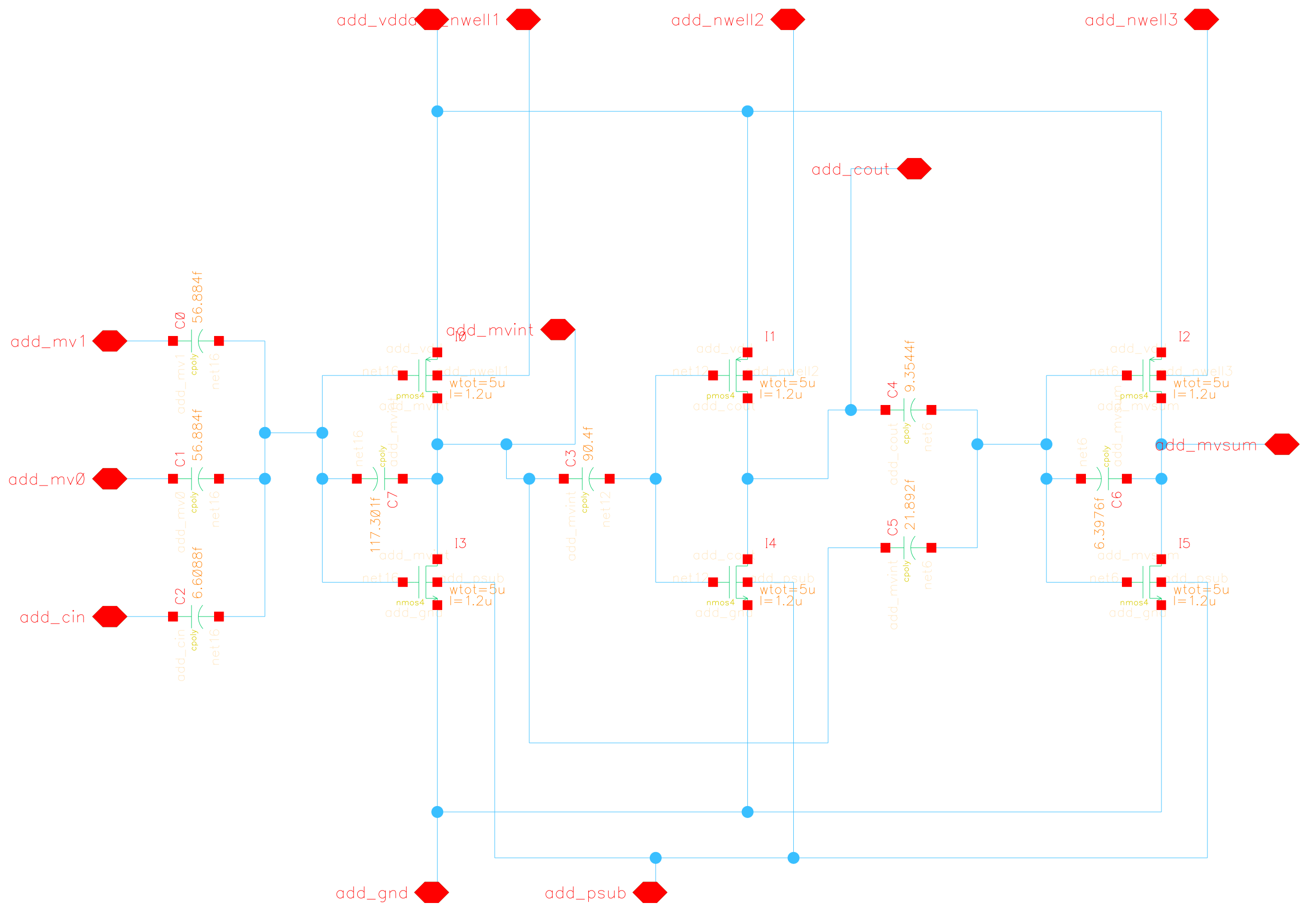


Figure A.5: *MV full-adder schematic from Cadence. The circuits has two multiple-valued inputs of the same radix,* add_mv0 *and* add_mv1*. It also has one binary input, called* add_cin*. There is one multiple-valued output of the same radix as the multiple-valued inputs,* add_mvsum*, and one binary output, called* add_cout*. The capacitor values are listed above capacitors. Note that all of the transistors in the circuit are of the same size.*

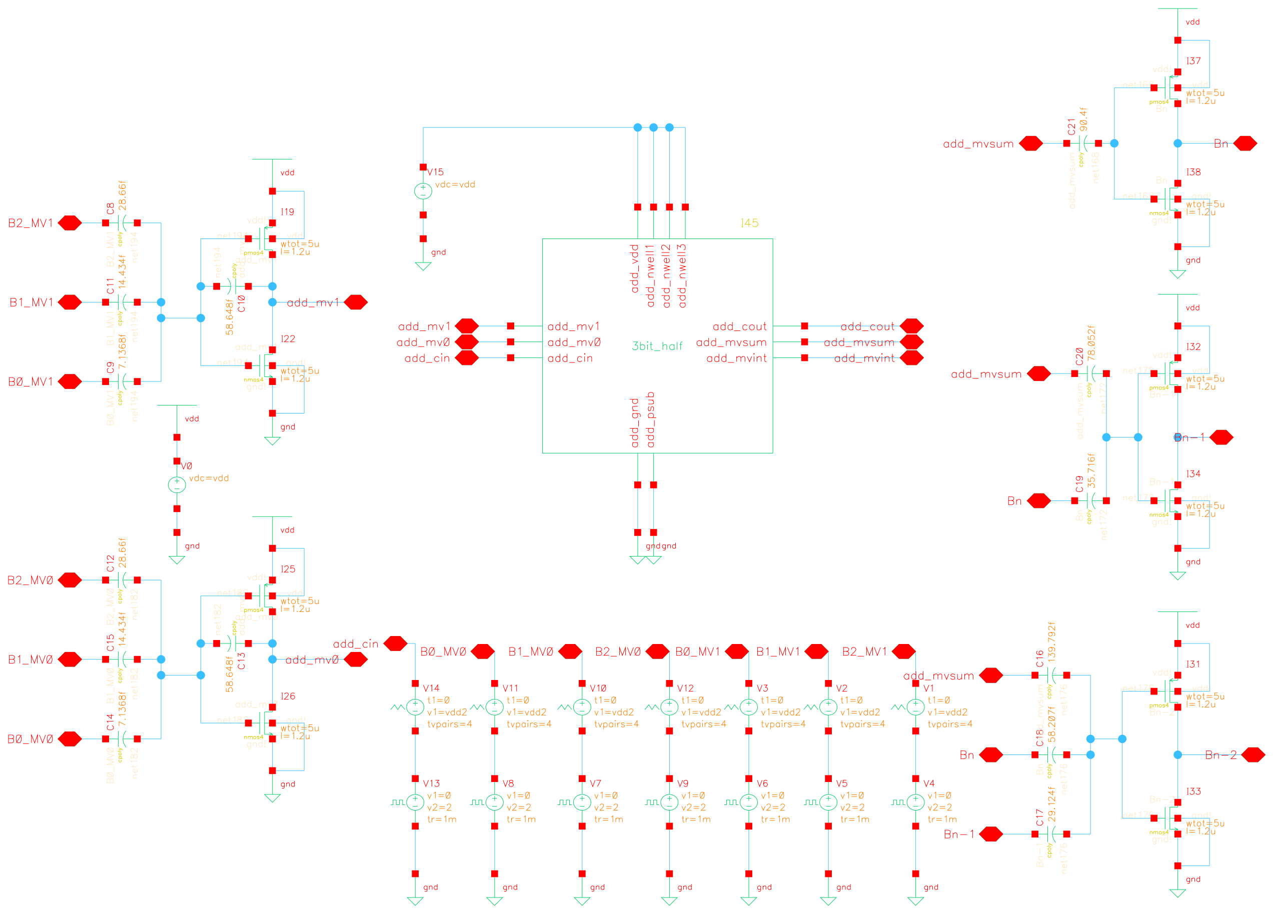


Figure A.6: *MV full-adder simulation setup from Cadence. The two circuits on the left (both are BMVCs) generate the multiple-valued signal inputs using the voltage sources at the bottom of the figure. The left most voltage source is used to generate the binary carry-in signal. The three circuits on the right (which constitutes the three stages in an MVBC) takes the multiple-valued sum signal and converts it to binary.*

Appendix B

Measurement Setup

B.1 Measurement Equipment

Various different instruments were used to measure and to drive the circuits. The instruments were programmable from Matlab via a GPIB network connected to a SUN workstation. The Keithly 617, seen in Figure B.1 (c), was used to measure the voltage output of the BMVC. The Keithly 236, which can be seen in Figure B.1 (b), was used to measure the BMVC current. The last instrument used was the Keithley 213, which was used to drive the circuits.

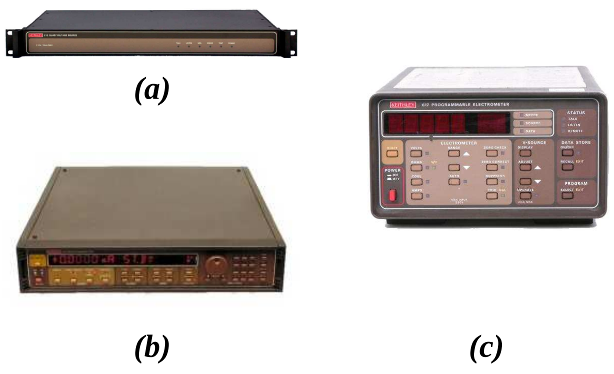


Figure B.1: ***(a)*** *Keithley 213. A quadruple voltage source.* ***(b)*** *Keithley 236. A voltage source/amperemeter or current source/voltmeter.* ***(c)*** *Keithley 617. A programmable multimeter with a single voltage source.*

B.2 Measurement Schematic

The measurement setup for the BMVC is shown in Figure B.2. The current is measured at the source terminal, which due to the programming technique is not connected to the substrate. This might cause some body effect during normal biased operation, unless the routing resistance is kept to a minimum[1]. The same applies for the n-well contacts of the pMOS transistor. They are not connected to the V_{dd} supply rail in order to allow for back-gating.

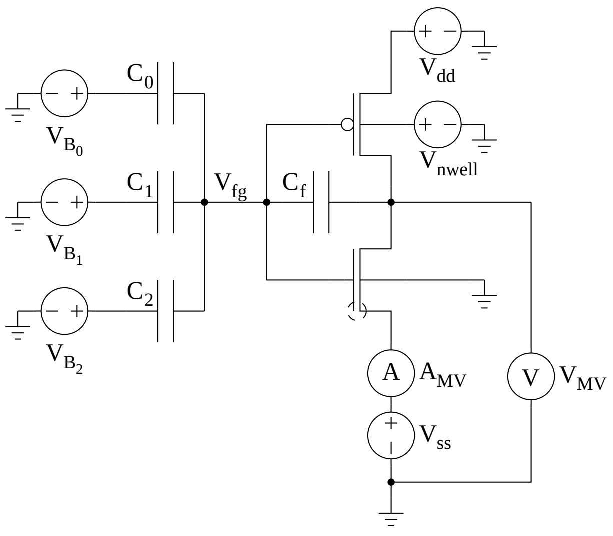


Figure B.2: *Measurement setup of the BMVC. The voltage output and the current through the nMOS transistor is measured. There is also a voltage source connected to the back-gate of the pMOS transistor, which can be used to fine-tune the circuit. Also note that the V_{dd} terminal is not connected to the n-well and the V_{ss} terminal is not connected to ground.*

[1]This is not a special circumstance for FGUVMOS circuits. Routing resistances for signal ground should always be kept to a minimum to avoid the formation of ground loops.

Appendix C

Chip Layout

C.1 Layout

The overall layout of the die and padring–as well as the layout of the circuits–were created using Virtuoso, the layout editor inside Cadence. The overall die layout, including the padring, can be seen in Figure C.1. The layout was developed in accordance with the layout guidelines for the AMS $0.6\mu m$ CUX CMOS process. The circuits were not constructed using the guidelines presented in section 3.6, and are thus more vulnerable for process variations which might cause device mismatch, especially for the capacitors. One can also clearly see the the internal contacts on the floating-gate and that the transistors does not have a ring topology. This is a problem which is discussed further in section 5.1.3. One noteworthy detail all the circuit layouts have in common, is the metal shielding covering the nMOS transistor. The layout of the binary to multiple-valued converter is shown in Figure C.2. In Figure C.3, we can see the layout of the multiple-valued to binary converter. And the layout of the multiple-valued full-adder can be seen in Figure C.4.

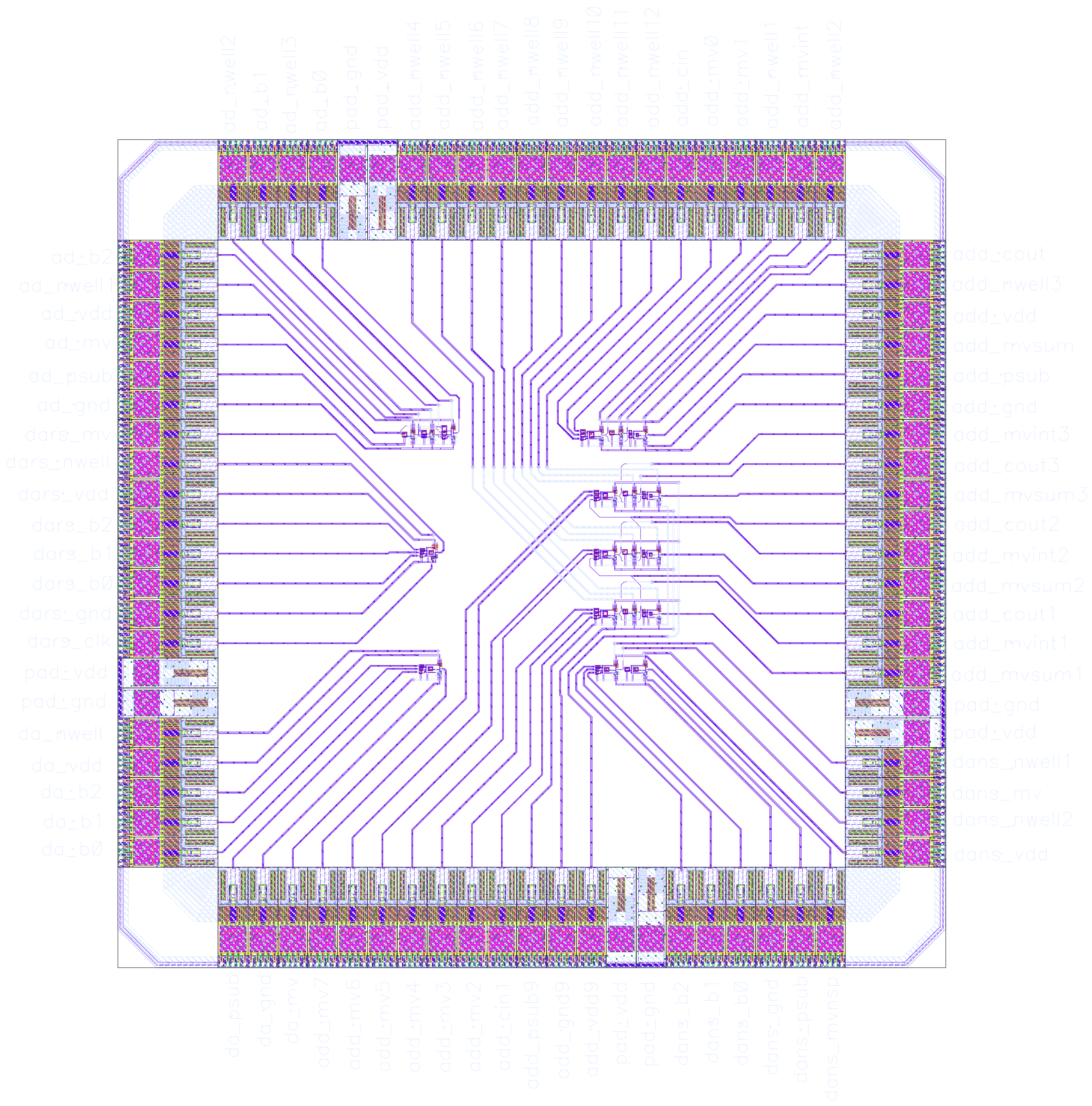


Figure C.1: *Overall die layout with pad descriptions. The BMVC circuit is located at the bottom left corner. In the upper left corner, the MVBC circuit can be found. And in the upper right corner, the multiple-valued full-adder is situated.*

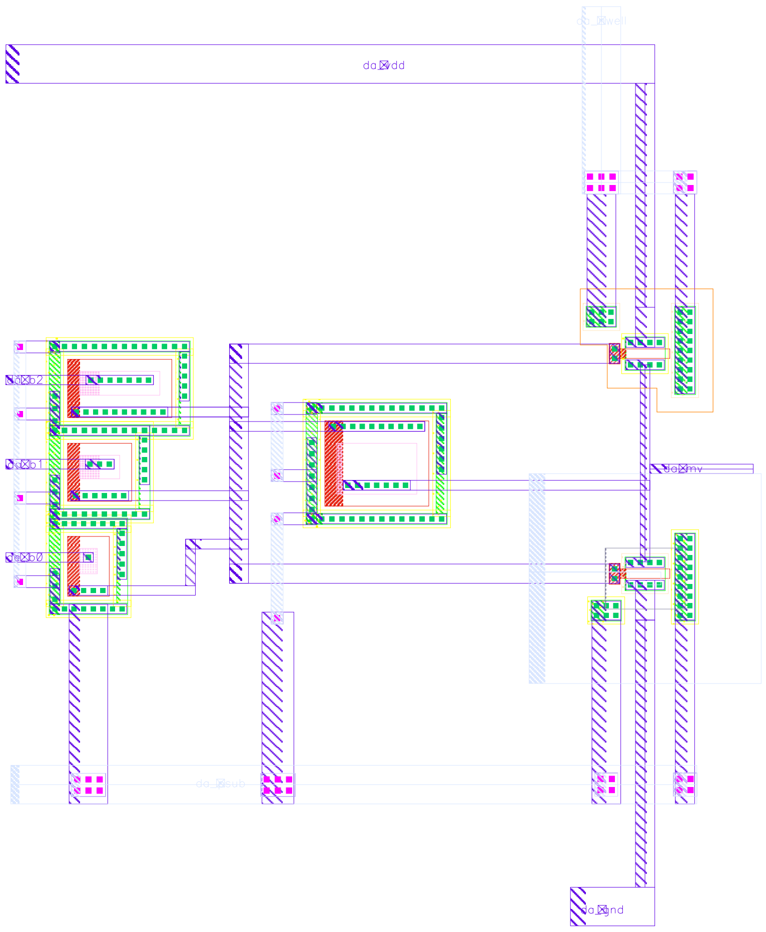


Figure C.2: *Layout of the binary to multiple-valued converter. The three input coupling capacitors can be seen in the left side of the figure. The uppermost input coupling capacitor is for the MSB of the binary input signal, while the lowermost input coupling capacitor is for the LSB. The feedback capacitor is located in the middle.*

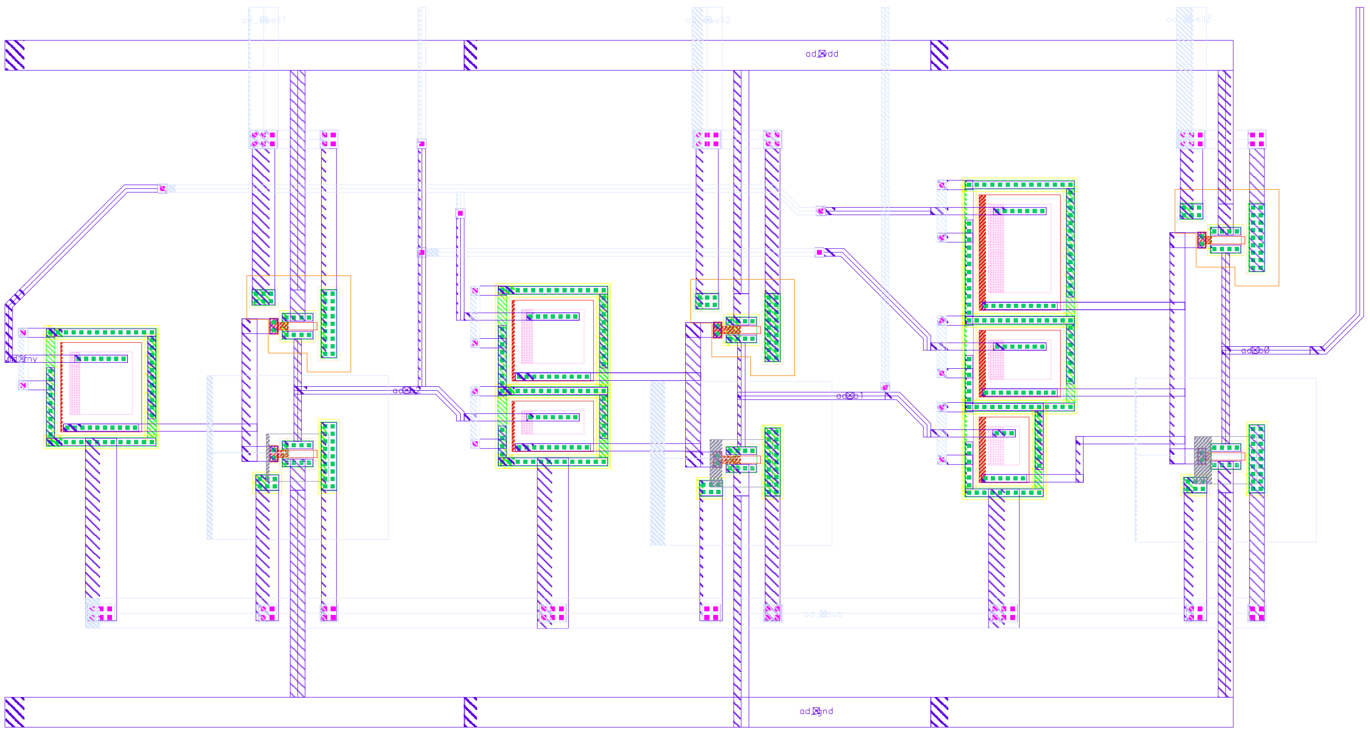


Figure C.3: *Layout of the multiple-valued to binary converter. The first conversion stage–the stage responsible for converting the MSB part of the binary signal–is located on the left side, second stage in the middle and the LSB stage on the right side of the figure.*

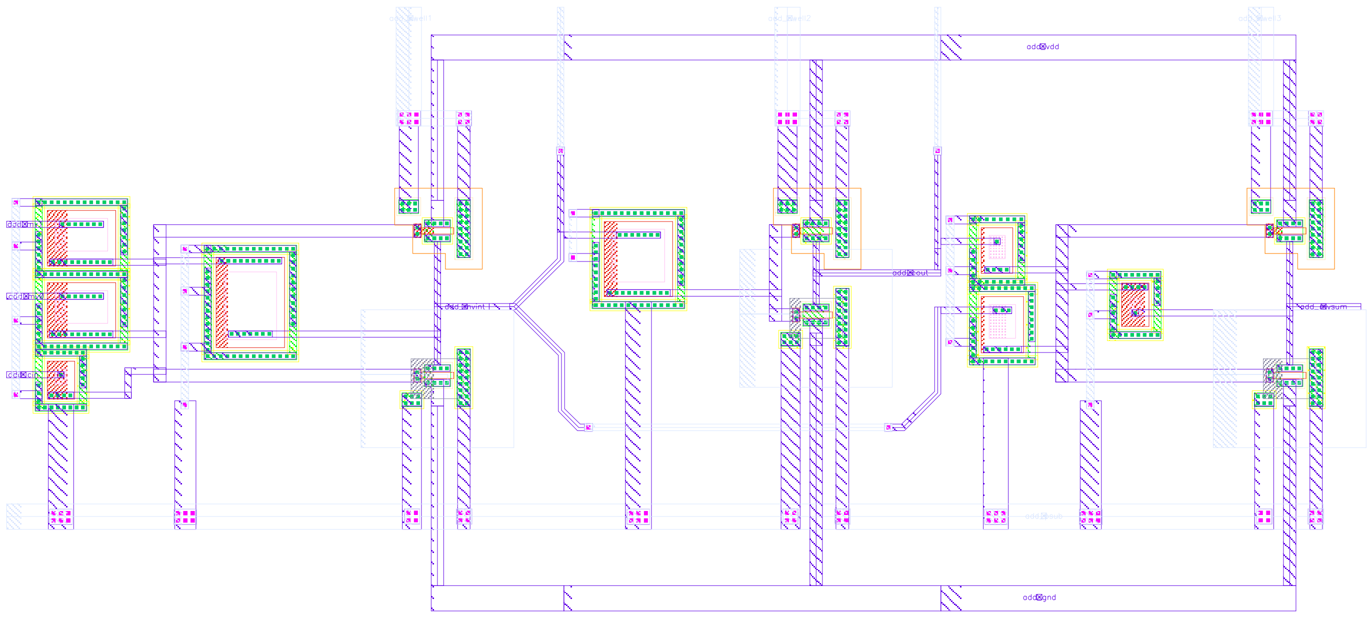
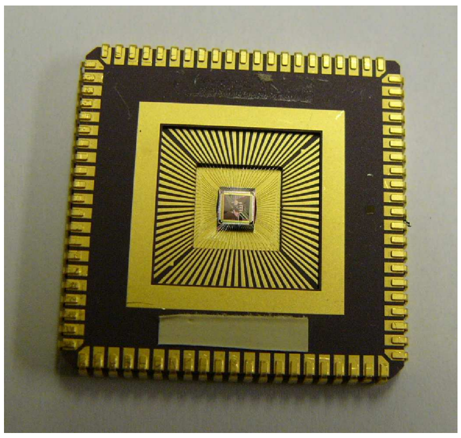


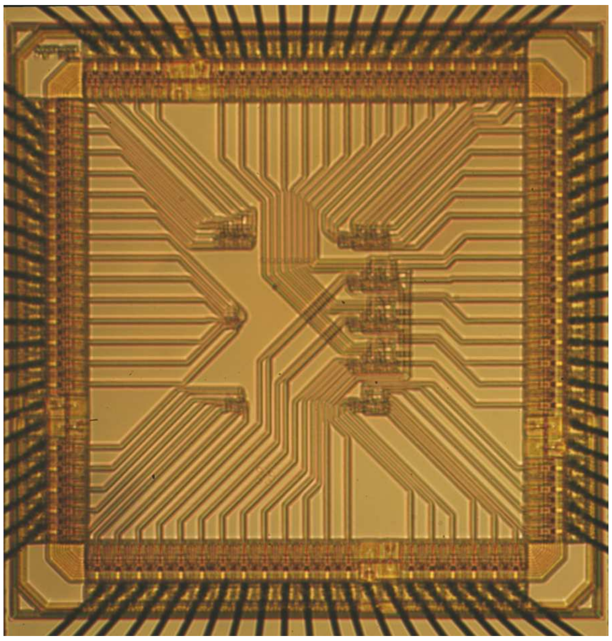
Figure C.4: *Layout of the multiple-valued full-adder. The input stage which adds the multiple-valued signal together–as well as the binary carry-in signal–is located on the left side. The two uppermost capacitor are for the multiple-valued input signals, while the lowermost is for the binary carry-in signal. In the middle we have the stage which generated the carry-out signal. And on the right side of the figure we can see the stage which generates the multiple-valued output sum signal.*

C.2 Fabricated Chip

The fabricated chip is shown in Figure C.5 (a). The chip was fabricated using the AMS $0.6\mu m$ CUX CMOS process. The process features three metal layers and two polysilicon layers. However, to fabricate the chips in this thesis, any standard CMOS double polysilicon layer process will suffice. A photography of the die through a microscope is shown in Figure C.5 (b).



(a)



(b)

Figure C.5: ***(a)*** *Photography of the chip. The die can be seen in the middle of the chip.* ***(b)*** *Photography of the die through a microscope. The bondwires are visible at the picture edge, with the padring encircling interconnects to the circuits in the middle of the die.*

Appendix D

Matlab Scripts

D.1 Simulations

Scripts related to various simulation tasks are presented in this section. These scripts extract data from a text file produced by the spectreS simulation program. The simulation data is then processed by the scripts to make it presentable for the thesis. Please refer to appendix A for a description of the simulation setups used to produce the text files containing the simulation data.

D.1.1 Binary Inverter Presentation

```
% Extract simulation data for the binary inverter and
% make the results presentable.


% binary inverter response to linear input
clear;
figure(1); clf; ylabel('V_{out} (V)');
xlabel('V_{in} (V)'); grid on; hold on;

% remove the top lines
! tail +4 bin_lin.txt > bin_lin.dat

% read in the simultion data with reference to time
[time, MV, Bn] = textread('bin_lin.dat',' %f %f %f');
```

```
% plot the binary inverter response to linear input
figure(1);
plot(MV, Bn);
print -deps2 ../../figs/bin_linear.eps;

% ------------------------------------------------------------

% binary inverter response to mv input
clear;
figure(2); clf; ylabel('V_{in}, V_{out} (V)');
xlabel('time (s)'); grid on; hold on;


% remove the top lines
! tail +4 bin_mv.txt > bin_mv.dat

% read in the simultion data with reference to time
[time, MV, Bn] = textread('bin_mv.dat',' %f %f %f');

% find the starting point of the simulation
start_idx=find(MV==0.2);
axis([time(start_idx(1)) time(length(time)) 0 2]);

% plot the binary inverter response to mv input
figure(2);
plot(time(start_idx(1):length(time)), ...
    MV(start_idx(1):length(time)),'--k');
plot(time(start_idx(1):length(time)), ...
    Bn(start_idx(1):length(time)));
legend('V_{in}','V_{out}', 0);
print -deps2 ../../figs/bin_mv.eps;
```

D.1.2 Multiple-Valued Inverter Presentation

```
% Transforms the simulations for the multiple-valued inverter
% into a presentable form.

% mv inverter response to linear voltage input
% mv inverter amplification (gm)
```

```
clear;

figure(1); clf; ylabel('V_{out} (V)');
xlabel('V_{in} (V)'); grid on; hold on;

figure(2); clf; ylabel('dV_{out}/dV_{in} (V)');
xlabel('V_{in} (V)'); grid on; hold on;

% remove the top lines
! tail +4 mv_inv.txt > mv_inv.dat

% read in simulation data with referance to time
[time, Bn, MV, dMV] = textread('mv_inv.dat',' %f %f %f %f');

% plot mv inverter linear voltage input response
figure(1);
plot(Bn, MV);
print -deps2 ../../figs/mv_linear.eps;

% find the starting and stopping point of the transition region
start_idx_array=find(Bn==0);
start_idx=start_idx_array(length(start_idx_array));
stop_idx_arrray=find(Bn==2);
stop_idx=stop_idx_arrray(1);

% plot mv inverter amplification
figure(2);
plot(Bn(start_idx+1:stop_idx), dMV(start_idx+1:stop_idx)./100);
print -deps2 ../../figs/mv_amp.eps;
```

D.1.3 Binary to Multiple-Valued Converter Presentation

```
% All the simulations performed on the binary to multiple-valued
% converter are processed into a presentable form.

% bmvc response to binary input
clear;
figure(3); clf;
```

```
% remove the top lines
! tail +4 bmvc.txt > bmvc.dat

% read in simulation data with reference to time
[time, B0, B1, B2, MV] = ...
    textread('bmvc.dat','              %f %f %f %f %f');

% find starting point of simulation
start_idx=find(B0==2);

% plot bmvc response to binary input
figure(3);
subplot(4,1,1);
axis([time(start_idx(1)) time(length(time)) 0 2]);
grid on; hold on; ylabel('V_{B0} (V)');
set(gca,'XTickLabel',[]);
plot(time(start_idx(1):length(time)), ...
    B0(start_idx(1):length(time)));

subplot(4,1,2);
axis([time(start_idx(1)) time(length(time)) 0 2]);
grid on; hold on; ylabel('V_{B1} (V)');
set(gca,'XTickLabel',[]);
plot(time(start_idx(1):length(time)), ...
    B1(start_idx(1):length(time)));

subplot(4,1,3);
axis([time(start_idx(1)) time(length(time)) 0 2]);
grid on; hold on; ylabel('V_{B2} (V)');
set(gca,'XTickLabel',[]);
plot(time(start_idx(1):length(time)), ...
    B2(start_idx(1):length(time)));

subplot(4,1,4);
axis([time(start_idx(1)) time(length(time)) 0 2]);
grid on; hold on; ylabel('V_{MV} (V)');
xlabel('time (s)');
plot(time(start_idx(1):length(time)), ...
    MV(start_idx(1):length(time)));
```

```
print -deps2 ../../figs/bmvc_mv.eps;
```

D.1.4 Multiple-Valued to Binary Converter Presentation

```
% Processing of all the simulations done on the multiple-valued to
% binary converter to transform them into a presentable form.

% MVBC response to mv input
clear;
figure(1); clf;

% remove the top lines
! tail +4 mvbc.txt > mvbc.dat

% read in the simulation data with reference to time
[time, MV, B0, B1, B2] = ...
    textread('mvbc.dat','%f %f %f %f %f');

% find the starting point of the simulation
start_idx=find(MV==0.2);

% plot the mvbc response to mv input
figure(1);
subplot(4,1,1);
axis([time(start_idx(1)) time(length(time)) 0 2]);
grid on; hold on; ylabel('V_{MV} (V)');
set(gca,'XTickLabel',[]);
plot(time(start_idx(1):length(time)), ...
    MV(start_idx(1):length(time)));

subplot(4,1,2);
axis([time(start_idx(1)) time(length(time)) 0 2]);
grid on; hold on; ylabel('V_{B0} (V)');
set(gca,'XTickLabel',[]);
plot(time(start_idx(1):length(time)), ...
    B0(start_idx(1):length(time)));

subplot(4,1,3);
axis([time(start_idx(1)) time(length(time)) 0 2]);
```

```
grid on; hold on; ylabel('V_{B1} (V)');
set(gca,'XTickLabel',[]);
plot(time(start_idx(1):length(time)), ...
    B1(start_idx(1):length(time)));

subplot(4,1,4);
axis([time(start_idx(1)) time(length(time)) 0 2]);
grid on; hold on; ylabel('V_{B2} (V)');
xlabel('time (s)');
plot(time(start_idx(1):length(time)), ...
    B2(start_idx(1):length(time)));

print -deps2 ../../figs/mvbc_mv.eps;
```

D.1.5 Multiple-Valued Adder Presentation

```
% Proccess simulations done on the multiple-valued full-adder
% in order to make them presentable for the thesis.

% mv adder response to binary input
clear;

% remove the top lines
! tail +4 adder.txt > adder.dat

% read in the simulation data with reference to time
[time, B0_A, B1_A, B2_A, B0_B, B1_B, B2_B, ...
        add_ci, add_B, add_A, ...
        add_Z, add_co, add_S, ...
        B2_S, B1_S, B0_S] = ...
    textread('adder.dat','%f %f %f %f %f %f %f %f %f' ...
             '%f %f %f %f %f %f %f');

% find starting point of simulation
start_idx=find(B0_A==0);

% plots of add_A, add_B, add_ci,
% add_Z, add_S, add_co
% and B0_A, B1_A, B2_A, B0_B, B1_B, B2_B
% and B2_S, B1_S, B0_S
%figure(1); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('V_{MV0} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
   % add_mv0(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_mv0.eps;

%figure(2); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('V_{MV1} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
   % add_mv1(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_mv1.eps;

%figure(3); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('V_{Ci} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    add_cin(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_cin.eps;

%figure(4); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('V_{int} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    add_mvint(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_mvint.eps;

%figure(5); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('V_{sum} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    add_mvsum(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_mvsum.eps;

%figure(6); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('V_{Co} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    add_cout(start_idx(1):length(time)));
```

```
%print -deps2 ../../figs/adder_cout.eps;

%figure(7); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('B0_{MV0} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    B0_MV0(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_B0_MV0.eps;

%figure(8); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('B1_{MV0} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    B1_MV0(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_B1_MV0.eps;

%figure(9); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('B2_{MV0} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    B2_MV0(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_B2_MV0.eps;

%figure(10); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('B0_{MV1} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    B0_MV1(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_B0_MV1.eps;

%figure(11); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('B1_{MV1} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    B1_MV1(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_B1_MV1.eps;

%figure(12); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('B2_{MV1} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    B2_MV1(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_B2_MV1.eps;

%figure(13); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('B0_{sum} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    B0_sum(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_B0_sum.eps;

%figure(14); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('B1_{sum} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    B1_sum(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_B1_sum.eps;

%figure(15); clf;
%axis([time(start_idx(1)) time(length(time)) 0 2]);
%ylabel('B2_{sum} (V)'); xlabel('time (s)'); grid on; hold on;
%plot(time(start_idx(1):length(time)), ...
%    B2_sum(start_idx(1):length(time)));
%print -deps2 ../../figs/adder_B2_sum.eps;


% plot a region (B=3) for all signals in 2 subplots
% find B=3 region, B0=0, B1=0, B2=2

% find indecies for the regions
region_idx1=find((B0_B == 0) & (B1_B==0) & ...
    (B2_B==2) & (add_ci==2));
region_idx2=find((B0_B == 0) & (B1_B==0) & ...
    (B2_B==2) & (add_ci==0));

% timeshift the second region to make the plot look nicer
timeshift=time(region_idx2(1)) - ...
    time(region_idx1((length(region_idx1))));
time(region_idx2) = time(region_idx2) - timeshift;

% concat the regions
region_idx=[region_idx1; region_idx2];
```

```
start_idx=region_idx(1);
end_idx=region_idx(length(region_idx));

% plot binary signals
figure(16); clf; grid on; hold on;

subplot(6,1,1);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{A,B0} (V)');
set(gca,'XTickLabel',[]);
plot(time(region_idx), B0_A(region_idx));

subplot(6,1,2);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{A,B1} (V)');
set(gca,'XTickLabel',[]);
plot(time(region_idx), B1_A(region_idx));

subplot(6,1,3);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{A,B2} (V)');
set(gca,'XTickLabel',[]);
plot(time(region_idx), B2_A(region_idx));

subplot(6,1,4);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{S,B0} (V)');
set(gca,'XTickLabel',[]);
plot(time(region_idx), B0_S(region_idx));

subplot(6,1,5);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{S,B1} (V)');
set(gca,'XTickLabel',[]);
plot(time(region_idx), B1_S(region_idx));

subplot(6,1,6);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{S,B2} (V)');
xlabel('time (s)');
plot(time(region_idx), B2_S(region_idx));
```

```
print -deps2 ../../figs/adder_region.eps;


% plot multiple-valued signals
figure(17); clf; grid on; hold on;

subplot(5,1,1);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{A} (V)');
set(gca,'XTickLabel',[]);
plot(time(region_idx), add_A(region_idx));

subplot(5,1,2);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{Ci} (V)');
set(gca,'XTickLabel',[]);
plot(time(region_idx), add_ci(region_idx));

subplot(5,1,3);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{Z} (V)');
set(gca,'XTickLabel',[]);
plot(time(region_idx), add_Z(region_idx));

subplot(5,1,4);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{S} (V)');
set(gca,'XTickLabel',[]);
plot(time(region_idx), add_S(region_idx));

subplot(5,1,5);
axis([time(start_idx) time(end_idx) 0 2]);
grid on; hold on; ylabel('V_{Co} (V)');
xlabel('time (s)');
plot(time(region_idx), add_co(region_idx));

print -deps2 ../../figs/adder_region2.eps;
```

D.2 Measurements

Several scripts pertaining to the measurement of the BMVC are presented here. They control the measurement equipment, timing of the UV-programming and set the inputs of and receive outputs from the measurement equipment during normal biased operation. There are also scripts that extract the data and make it presentable for the thesis. Please refer to appendix B for a detailed description of the measurement setup.

D.2.1 UV-light Control

```
% Turns the UV-light on using a relay controlled by K213

! echo ‘date‘ : UV light on
K213_SetDigital(255, ’K213A’);
```

```
% Turns the UV-light off using a relay controlled by K213

! echo ‘date‘ : UV light off
K213_SetDigital(0, ’K213A’);
```

D.2.2 Programming Voltages

```
% Sets the programming voltages to use for later on

function prog(ProgVss, ProgVdd)

save prog.mat ProgVss ProgVdd
```

D.2.3 UV-light Programming

```
% Preforms the UV-light programming cycle and the
% convergence measurements.

function[a_uvl, v_uvl, time] = uv_light_prog(sec, Vdev, Adev)

uv_on;
```

```
tic;                                       % start timer
pause(sec); % program for sec seconds
v_uvl = K617_ReadQuick(Vdev);   % get convergence voltage
a_uvl = K236_ReadQuick(Adev);   % get convergence current
time = toc;                                % end timer
uv_off;
```

D.2.4 Initializing Measurement Equipment

```
% Initializes all the measurement equipment to a known state
% before performing the measurements.

%init UV light
uv_off;

% init K617 (3)

K617_Reset('K617A');
pause(1);
K617_Init('K617A');
pause(1);
K617_SetMode('V', 'K617A');

K617_Reset('K617B');
pause(1);
K617_Init('K617B');
pause(1);
K617_SetMode('V', 'K617B');

K617_Reset('K617C');
pause(1);
K617_Init('K617C');
pause(1);
K617_SetMode('V', 'K617C');


% init K213 (2)
K213_182Init('K213A');
pause(1);
K213_Init('K213A');
```

```
pause(1);
K213_SetVoltage(0, 1, ’K213A’);
K213_SetVoltage(0, 2, ’K213A’);
K213_SetVoltage(0, 3, ’K213A’);
K213_SetVoltage(0, 4, ’K213A’);

K213_182Init(’K213B’);
pause(1);
K213_Init(’K213B’);
pause(1);
K213_SetVoltage(0, 1, ’K213B’);
K213_SetVoltage(0, 2, ’K213B’);
K213_SetVoltage(0, 3, ’K213B’);
K213_SetVoltage(0, 4, ’K213B’);

% init K236, current and vdd
%K236_182Init(’K236A’);
K236_Init(’K236A’);
K236_SetMeasure(’A’, ’K236A’);
K236_SetVolt(0, ’K236A’);
pause(1);
K236_Operate(’K236A’);
```

D.2.5 BMVC Measurements

```
% Performs the measurements on the BMVC.

% DC response for 3-bit BMVC using binary input

! echo DAC measurement starting...
clear;
vdd = 2;
vss = 0;
figure(1); clf; ylabel(’V’); xlabel(’time’);
title(’convergence, with UV light’); grid on; hold on;

figure(2); clf; ylabel(’A’); xlabel(’time’);
title(’convergence, with UV light’); grid on; hold on;
```

```
figure(3); clf; ylabel('V'); xlabel('time');
title('convergence, no UV light'); grid on; hold on;

figure(4); clf; ylabel('A'); xlabel('time');
title('convergence, no UV light'); grid on; hold on;

figure(5); clf; ylabel('V'); xlabel('gate (V)');
title('linear measurement, voltage'); grid on; hold on;

figure(6); clf; ylabel('A'); xlabel('gate (V)');
title('linear measurement, current'); grid on; hold on;

figure(7); clf; ylabel('V'); xlabel('index');
title('MV measurement, voltage'); grid on; hold on;

figure(8); clf; ylabel('A'); xlabel('index');
title('MV measurement, voltage'); grid on; hold on;

% set timestamp for measurement files
! echo ‘date +%y%m%d%H‘ > datefile

% init all the equipment
! echo init all equipment...

init_equip;
Adev='K236A';
Vdev='K617B';

% program the chip with UV light
step_sec = 120; % in sec (120)
prog_time = 120; % in min (120)
load prog.mat % ProgVss
ProgVss
ProgVdd
V_UVL = [];
A_UVL = [];
V_noUVL = [];
A_noUVL = [];
Time = [];
```

```
! echo starting UV programming...

K213_SetVoltage(vdd/2, 1, 'K213A'); % B0
K213_SetVoltage(vdd/2, 2, 'K213A'); % B1
K213_SetVoltage(vdd/2, 3, 'K213A'); % B2
K213_SetVoltage(vdd, 1, 'K213B'); % NWell

% programming and measurement of convergence time
for i=0:step_sec:prog_time*60
  K213_SetVoltage(ProgVdd, 3, 'K213B'); % Vdd
  K236_SetVolt(ProgVss, 'K236A'); % Gnd
  [a_uvl, v_uvl, time] = uv_light_prog(step_sec, Vdev, Adev);
  pause(3);
  K213_SetVoltage(vdd, 3, 'K213B');     % Vdd
  K236_SetVolt(vss, 'K236A'); % Gnd
  pause(3);
  v_nouvl = K617_ReadQuick(Vdev); % get programmed voltage
  a_nouvl = K236_ReadQuick(Adev); % get programmed current
  v_nouvl
  %   add values to vector
  V_UVL = [V_UVL, v_uvl];
  A_UVL = [A_UVL, a_uvl];
  V_noUVL = [V_noUVL, v_nouvl];
  A_noUVL = [A_noUVL, a_nouvl];
  % previous time index, to get the time right
  t = i/step_sec;
  if t <= 0
    Time = [Time, time];
  else
    Time = [Time, time+Time(t)];
  end
  figure(1); plot(Time, V_UVL);
  figure(2); plot(Time, A_UVL);
  figure(3); plot(Time, V_noUVL);
  figure(4); plot(Time, A_noUVL);
end

% uniquely store data for later
save dac_uv.mat ProgVdd vdd ProgVss vss ...
     Time V_UVL V_noUVL A_UVL A_noUVL
! cp dac_uv.mat dac/dac_uv.`cat datefile`.mat
! echo starting measurement run...

% set all voltages
K213_SetVoltage(vdd, 1, ’K213B’); % NWell
K213_SetVoltage(vdd, 3, ’K213B’); % Vdd
K236_SetVolt(vss, ’K236A’); % Gnd

% measure and store in vector and file

% linear measurement
  Vdac = [];
  Adac = [];
  Din = [];
  for din=0:0.01:2
    K213_SetVoltage(din, 1, ’K213A’); % B0
    K213_SetVoltage(din, 2, ’K213A’); % B1
    K213_SetVoltage(din, 3, ’K213A’); % B2
    pause(3);
    [resV, ovflA] = K617_ReadQuick(Vdev);
    [resA, ovflA] = K236_ReadQuick(Adev);
    din
    resV
    Din = [Din, din];
    Vdac = [Vdac, resV];
    Adac = [Adac, resA];
    figure(5); plot(Din, Vdac);
    figure(6); plot(Din, Adac);
  end
save dac_lin.mat vdd vss Din Vdac Adac
! cp dac_lin.mat dac/dac_lin.‘cat datefile‘.mat


% set all voltages
K213_SetVoltage(vdd, 1, ’K213B’); % NWell
K213_SetVoltage(vdd, 3, ’K213B’); % Vdd
K236_SetVolt(vss, ’K236A’); % Gnd

% MV measurement
BitVal = [0 0 0
```

```
          0 0 1
          0 1 0
          0 1 1
          1 0 0
          1 0 1
          1 1 0
          1 1 1];

  Vdac = [];
  Adac = [];
  Din = [];
  for din=1:8
   for i=1:10
     K213_SetVoltage(BitVal(din, 3)*vdd, 1, 'K213A'); % B0
     K213_SetVoltage(BitVal(din, 2)*vdd, 2, 'K213A'); % B1
     K213_SetVoltage(BitVal(din, 1)*vdd, 3, 'K213A'); % B2
     pause(3);
     [resV, ovflA] = K617_ReadQuick(Vdev);
     [resA, ovflA] = K236_ReadQuick(Adev);
     din
     resV
     Din = [Din, din];
     Vdac = [Vdac, resV];
     Adac = [Adac, resA];
     figure(7); plot(Vdac);
     figure(8); plot(Adac);
   end
  end
save dac_mv.mat vdd vss Din Vdac Adac
! cp dac_mv.mat dac/dac_mv.`cat datefile`.mat

! echo DAC measurement ending...
```

D.2.6 BMVC Presentation

```
% Makes the BMVC measurement data presentable.

% plot measurements for the BMVC
```

```
clear;
% ready figure windows

% linear with ideal+actual
figure(1); clf; ylabel('V_{MV} (V)');
xlabel('V_{Bn} (V)'); grid on; hold on;

% linear with deviance of actual from ideal
figure(2); clf; ylabel('V_{MV},ideal - V_{MV},actual (V)');
xlabel('V_{Bn} (V)'); grid on; hold on;

% linear amp with ideal+actual
figure(3); clf; ylabel('dV_{MV}/dV_{Bn} (V)');
xlabel('V_{Bn} (V)'); grid on; hold on;

% mv with ideal+actual
figure(4); clf; ylabel('V_{MV} (V)');
xlabel('Binary input (B_{2}B_{1}B_{0})'); grid on; hold on;
set(gca,'XTickLabel',[' ']);

% mv with deviance of actual from ideal
figure(5); clf; ylabel('V_{MV},ideal - V_{MV},actual (V)');
xlabel('Binary input (B_{2}B_{1}B_{0})'); grid on; hold on;
set(gca,'XTickLabel',[' ']);

% mv amp with ideal+actual
% $$$ figure(6); clf; ylabel('V_{MV}/dV_{Bn} (V)');
% $$$ xlabel('Binary input (B_{2}B_{1}B_{0})'); grid on; hold on;
% $$$ set(gca,'XTickLabel',[' ']);


% -----------------------

% load file with linear measurements
load('dac_lin.mat');

% linear with ideal+actual
X=0:2/(length(V_mv)-1):2;
y(1)=V_mv(1);
y(2)=1;
y(3)=V_mv(length(V_mv));
```

```
x=[0 1 2];
z=polyfit(x, y, 1);
V_mv_ideal=polyval(z, X);
figure(1); plot(X, V_mv_ideal, ’b--’);

%plot linear actual
X=0:2/(length(V_mv)-1):2;
figure(1); plot(V_bin, V_mv,’k-’); legend(’ideal’,’actual’, 0);

print -deps2 ../../figs/dac_linear.eps;


% linear with deviance of actual from ideal
X=0:2/(length(V_mv)-1):2;
y(1)=V_mv(1);
y(2)=1;
y(3)=V_mv(length(V_mv));
x=[0 1 2];
z=polyfit(x, y, 1);
V_mv_ideal=polyval(z, X);
%V_mv_dev=abs(V_mv_ideal - V_mv);
V_mv_dev=V_mv_ideal - V_mv;
figure(2); plot(X, V_mv_dev,’k-’);

print -deps2 ../../figs/dac_linear_dev.eps;


% linear amp with ideal+actual
X=0:2/(length(V_mv)-1):2;
Vin_diff=diff(X);
y(1)=V_mv(1);
y(2)=1;
y(3)=V_mv(length(V_mv));
x=[0 1 2];
z=polyfit(x, y, 1);
V_mv_ideal=polyval(z, X);
V_mv_ideal_diff=diff(V_mv_ideal);
X=0+2/(length(V_mv)-1):2/(length(V_mv)-1):2;
figure(3); plot(X, V_mv_ideal_diff./Vin_diff,’b--’);

V_mv_diff=diff(V_mv);
```

```
%figure(3); plot(X, V_mv_diff./Vin_diff,'k-');
Vsmooth = csaps(X, V_mv_diff./Vin_diff, 0.5, X);
figure(3); plot(X, Vsmooth,'k-'); legend('ideal','actual', 0);

print -deps ../../figs/dac_linear_amp.eps;


% -----------------------

% load file with  measurements
load('dac_mv.mat');


% mv with ideal+actual
low_ind = find(V_bin==1);
high_ind = find(V_bin==8);
low_mean = mean(V_mv(low_ind));
high_mean = mean(V_mv(high_ind));
mv_steps = (high_mean - low_mean) / 7 ;
V_mv_ideal = [];
for i=1:8
 for j=1:10
  V_mv_ideal = [V_mv_ideal low_mean + (mv_steps*(i-1))];
 end
end
figure(4); plot(V_mv_ideal,'b--');

figure(4); plot(V_mv,'k-'); legend('ideal','actual', 0);

print -deps2 ../../figs/dac_mv.eps;


% mv with deviance of actual from ideal

low_ind = find(V_bin==1);
high_ind = find(V_bin==8);
low_mean = mean(V_mv(low_ind));
high_mean = mean(V_mv(high_ind));
mv_steps = (high_mean - low_mean) / 7 ;
V_mv_ideal = [];
for i=1:8
```

```
 for j=1:10
  V_mv_ideal = [V_mv_ideal low_mean + (mv_steps*(i-1))];
 end
end
%V_mv_dev=abs(V_mv_ideal - V_mv);
V_mv_dev=V_mv_ideal - V_mv;
figure(5); plot(V_mv_dev,'k-');

print -deps2 ../../figs/dac_mv_dev.eps;



% mv amp with ideal+actual
% linear amp with ideal+actual
% $$$ low_ind = find(V_bin==1);
% $$$ high_ind = find(V_bin==8);
% $$$ low_mean = mean(V_mv(low_ind));
% $$$ high_mean = mean(V_mv(high_ind));
% $$$ mv_steps = (high_mean - low_mean) / 7 ;
% $$$ V_mv_ideal = [];
% $$$ for i=1:8
% $$$  for j=1:10
% $$$   V_mv_ideal = [V_mv_ideal low_mean + (mv_steps*(i-1))];
% $$$  end
% $$$ end
% $$$ V_mv_ideal_amp = [];
% $$$ for i=1:length(V_mv)-10
% $$$  V_mv_ideal_amp(i+10) = V_mv_ideal(i) - V_mv_ideal(i+10);
% $$$ end
% $$$ figure(6); plot(V_mv_ideal_amp,'b--');
% $$$
% $$$ V_mv_amp = [];
% $$$ for i=1:length(V_mv)-10
% $$$  V_mv_amp(i+10) = V_mv(i) - V_mv(i+10);
% $$$ end
% $$$ figure(6); plot(V_mv_amp,'k-'); legend('ideal','actual', 0);
% $$$
% $$$ print -deps2 ../../figs/dac_mv_amp.eps;
```

Appendix E

Publications

Yngvar Berg, Snorre Aunet, Øivind Næss, Øyvind Hagen and Mats Høvin, **A Novel Floating-Gate Multiple-Valued CMOS Full-Adder**, IEEE International Symposium on Circuits and Systems, vol. 1, pp. 877–880, Arizona, 26th-29th of May 2002.