HD-IHEP 99-xx
HD-ASIC 48-1099

# An Integrated Analog Network for Image Processing

INSTITUT FÜR HOCHENERGIEPHYSIK
UNIVERSITÄT HEIDELBERG

Dissertation
von
Johannes Schemmel

INAUGURAL-DISSERTATION
zur
Erlangung der Doktorwürde

der
Naturwissenschaftlich-Mathematischen
Gesamtfakultät

der
Ruprecht-Karls-Universität
Heidelberg

vorgelegt von
Dipl.-Phys. Johannes Schemmel
aus Mannheim

Tag der mündlichen Prüfung: 20.10.1999

# An Integrated Analog Network for Image Processing

Gutachter: Prof. Dr. Karlheinz Meier
Prof. Dr. Bernd Jähne

DISSERTATION

submitted to the
Combined Faculties for the Natural Sciences and for Mathematics

of the Rupertus Carola University of
Heidelberg, Germany

for the degree of
Doctor of Natural Sciences

# An Integrated Analog Network for Image Processing


presented by

Diplom-Physicist: Johannes Schemmel
born in: Mannheim


Heidelberg, October 20, 1999

Referees: Prof. Dr. Karlheinz Meier
Prof. Dr. Bernd Jähne

**Ein integriertes analoges Netzwerk für die Bildverarbeitung** - Ein wichtiger Schritt in der elektronischen Bildverarbeitung ist die Kantendetektion. Im Rahmen dieser Doktorarbeit wurde dafür ein Mikrochip entworfen und getestet. Er wurde in einer CMOS-Technologie (complementary metal oxide semiconductor) gefertigt und verarbeited Bilder mit einer Größe von $66 \times 66$ Bildpunkten in wenigen Mikrosekunden. Die Berechnungen werden dabei parallel in einem analogen Netzwerk aus speziellen Bauelementen ausgeführt, die die linearen Eigenschaften von Widerständen mit dem nichtlinearen Verhalten einer Sicherung kombinieren. Diese *resistive fuses* genannten Schaltungselemente wurden in einer Weise implementiert, die sich sowohl durch geringen Platzbedarf als auch durch Robustheit gegenüber den bei der Halbleiterherstellung unvermeidlichen Schwankungen der Bauteilparameter auszeichnet.

**An integrated analog network for image processing** - An important step of electronic image processing is edge detection. Within the scope of this Ph.D. thesis a microchip was developed and tested for this purpose. Realized in a complementary metal oxide semiconductor (CMOS) technology it is capable of handling images with a size of $66 \times 66$ pixels in a few microseconds. The processing is done in parallel by an analog network of special components which are combining the linear behavior of a resistor with the nonlinear one of a fuse. Those circuits, called *resistive fuses*, have been implemented in a way that is outstanding in terms of area usage and robustness against the unavoidable parameter fluctuations of semiconductor manufacturing processes.

# Contents

# Introduction

Being visually handicapped or blind is a severe limitation for the life of the affected people. Technology can help to make many things easier. In recent years even the replacement of the lost sight seems no longer impossible but is still many years away from practical usage. A technical assistance that could be available much sooner is the transfer of visual information to another sensory channel of human beings. Hearing and touch are the favorite alternative pathways to the brain.

This work was motivated by the need for a fast and portable image processing system. Such a system can be used as a front-end either for acoustic or tactile vision substitution systems that work not only indoors but in natural environments, too. This thesis is part of a project developing such a system based on the sense of touch [MEI96]. The image sensor is a black-and-white CMOS[1]-camera with logarithmic response and integrated fixed pattern noise correction small enough to be head-mounted, for example at some blind eyeglasses. It has some prerequisites for image processing built in, being able to output arbitrary regions of the picture at selectable resolutions [LOO98, LOO99]. The logarithmic response curve of the camera together with a dynamic range of about 6 decades help to identify image structures under very different illumination conditions[2].

In a traditional approach an analog-to-digital converter (ADC) would produce a digital data stream from the analog camera signal. This would be stored in a memory and processed by a linear filter operator like the Sobel or Kirsch operators [KLE95-1] before binarisation [KLE95-2]. Power and space consumption should be minimized for portability, but not at the cost of speed. For this reason a new approach has been chosen for the mentioned tasks. The camera output is directly fed into an analog processing array for edge detection and noise suppression: the *EDDA* chip (EDge Detection Array). This VLSI[3] microchip converts the analog gray scale picture into a binary edge image with a method called *resistive fuse technique*. It is based on a novel switched capacitor (SC) circuit developed in this thesis.

With the camera and the EDDA chip, a front-end for an active vision system exists. After acquiring and processing the image in a coarse resolution, interesting parts can be subsequently analyzed at higher detail levels. This thesis focuses on the EDDA chip. The algorithm it uses, its design and the results from the final silicon are presented in the following chapters.

In chapter 1 the resistive fuse algorithm and its theoretical background are explained in detail. The switched capacitor realization of the algorithm is presented. A numerical simulation has been developed to evaluate the algorithm and check the results from the VLSI-implementation.

[1] Complementary Metal Oxide Semiconductor. Todays most commonly used semiconductor manufacturing process.

[2] If the same structure is viewed under two different illumination levels with a linear camera, the ratios between the graylevels are preserved whereas a logarithmic image sensor keeps the differences constant. In hardware implementations subtractions are usually less costly than divisions making a logarithmic sensor an advantage.

[3] Very Large Scale Integration

Based on the SC circuit a mixed-signal CMOS microchip has been designed. Its implementation is described in chapter 2. It covers the analog network as well as the necessary circuitry needed for I/O, clock generation and control. Analog circuits are more sensitive to device variations than digital ones. Therefore the layout plays a crucial part in optimizing the performance like reducing the fixed-pattern noise for example. The layout techniques and their usage are also shown in chapter 2.

Despite being a prototype the EDDA chip contains the necessary circuitry for integrating it into the final portable vision system. Consequently the test setup described in chapter 3 uses only technologies that fit in such a system. An FPGA[4] does the interfacing to EDDA, connected via the PCI[5] bus to a microprocessor. The analog measurements are done by an integrated ADC. Chapter 3 covers also aspects of the software used for testing. Like the hardware it is designed to be reused in the final system to control the image processing components.

Real world images and artificial test pictures have been used to measure the performance of EDDA. Chapter 4 shows the results. Image noise suppression and the chip's own fixed pattern noise are important aspects, as well as speed and power consumption.

[4]Field Programmable Gate Array
[5]Personal Computer Interconnect

# Chapter 1

# Resistive Fuse Networks

*This chapter introduces resistive fuse networks. It is explained how they are formed by adding nonlinear behavior to linear resistive networks. Their ability for edge detection and noise suppression is investigated. A novel switched capacitor circuit is shown as an implementation of a resistive fuse network suitable for VLSI integration. At the end of this chapter a numerical simulation is presented used to visualize the behavior of the network and to verify the SC circuit.*

## 1.1 Edge Detection

In the field of early vision[1] edge detection plays a prominent role. This step is one of the first performed in the retina of a vertebrate eye that correlates the outputs of a multitude of sensory cells. It is done in many ways by the different neurons, each of them being specialized for certain patterns. As in the biological model there is no simple definition for the structure called an *edge* in electronic image processing systems. In general a discontinuity in the gray scale image is called an edge if it can be distinguished somehow from the background noise and reaches a certain size and level. In real world sceneries, with plants or a far away background for example, most of the noise is generated by structures smaller than the resolution of the camera but not small enough to produce a similar mean value in neighboring pixels. Only by looking at a uniformly painted wall, which is not a likely situation, the noise of the camera will dominate. A priori a statistical distribution function for the kind of noise the edge detection system has to cope with is not known because it depends on the scenery. In the case of irregular textures or unwanted small details it may not even exist. To solve this problem adaptive filtering could be used.
On a large scale this can be done by the image processing system at a whole. It can change the camera resolution or tune the parameters of individual parts like the edge detection filter. This kind of feedback loop will work only if the individual parts of the image processing system self adapt to the input image at least good enough to be able to generate the error information for the feedback.

[1] The term *early vision* refers to the first steps performed by an image processing system which operate on the two-dimensional image data to extract local features like orientation or texture.

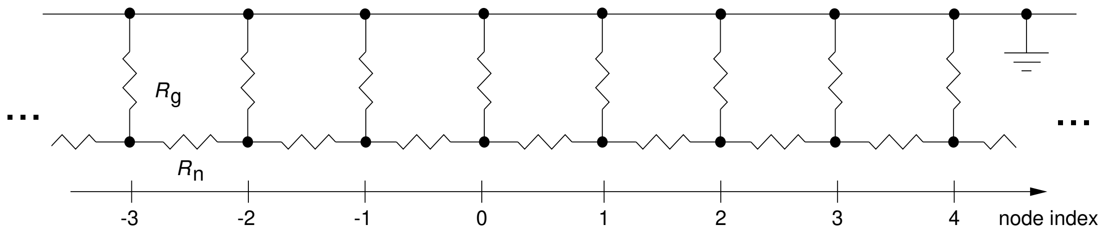



**Figure 1.1:** One-dimensional resistive network. $R_{\mathrm{g}}$ is the resistance to ground and $R_{\mathrm{n}}$ the resistance between the network nodes.

A good starting point, for example, would be looking at the whole image in low resolution while configuring the edge detection to suppress small structures. The coarse lines detected in this step could be used as an orientation after switching to higher detail levels. Adaptive filtering on a small scale means that the amount of smoothing along and orthogonal to an edge is different.
The prerequisites for a useful edge detection algorithm are:

- easy to parameterize (especially the edge threshold)
- adaptive noise suppression
- use of spatial correlations
- reduction of image information
- conservation of 'important' features

Any algorithm of this kind needs to look at the surrounding of each pixel boundary to decide if there is an edge or not. This leads to a high level of short range information exchange making it well suited for fully parallel implementations. If an analog solution is desired the restrictions imposed by VLSI design rules [GEI90-1] make additional constraints necessary:

- direct information exchange only between the nearest (at most the next-but-one) neighbors of a pixel to keep routing costs low
- low precision requirements (about 1 %)
- low per pixel complexity to reduce area usage

An edge detection method fulfilling all these conditions is the resistive fuse approach that will be explained in the next section.

## 1.2 The Resistive Fuse Concept

### 1.2.1 Linear Resistive Networks

Figure 1.1 shows a one-dimensional resistive network. Neighboring nodes are linked via the resistors $R_n$ while each $R_g$ connects the node to ground. For the continuous case, i.e. taking $R_{\mathrm{n}}$ as resistance and $1/R_{\mathrm{g}}$ as conductivity per unit length the voltage at node $x$ equals

$$V(x) = V(0)e^{-\frac{|x|}{l}} \tag{1.1}$$

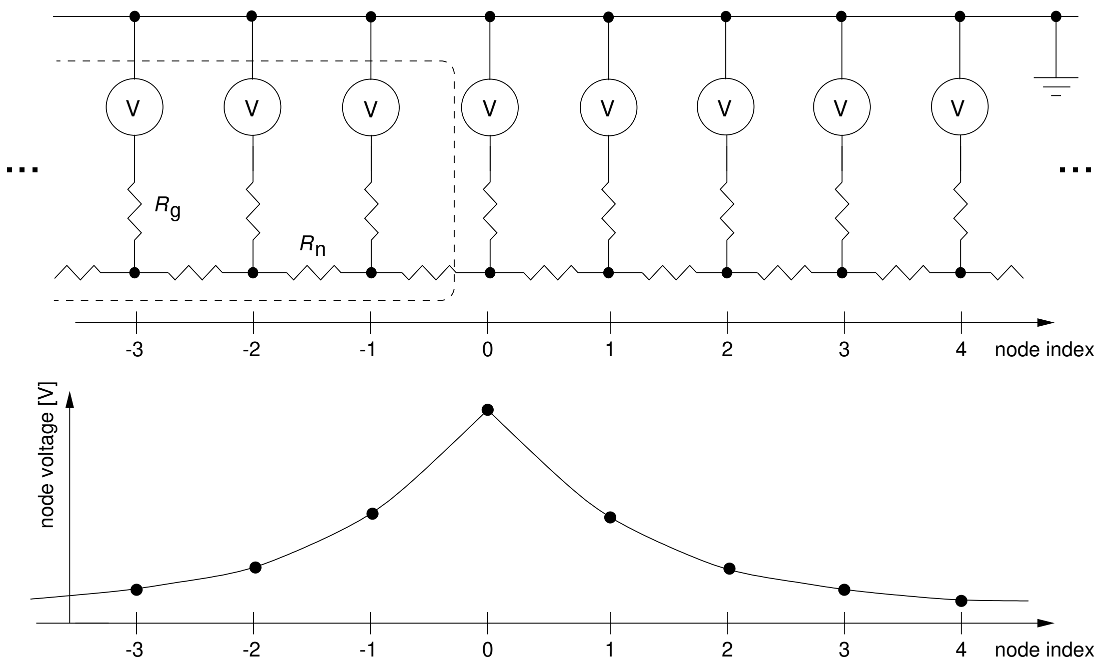



**Figure 1.2:** Top: Resistive network with voltage sources. Bottom: $V(0) \neq 0$. The exponential decay of the node voltages can be seen.

with the characteristic length

$$l = \sqrt{\frac{R_g}{R_n}} \tag{1.2}$$

and the initial voltage $V(0)$ at $x = 0$. In the discrete case the unit of $l$ changes from a length $x$ to a node index $n$. The equation for $V(n)$ is [MEA89]:

$$V(n) = V(0)\gamma^{|n|} \tag{1.3}$$

$$\gamma = 1 + \frac{1}{2l^2} - \frac{1}{l}\sqrt{1 + \frac{1}{4l^2}} \tag{1.4}$$

A comparison between both cases yields [LAN98]:

$$\gamma - e^{-\frac{1}{l}} = \frac{1}{24l^3} + \mathcal{O}\left(\frac{1}{l^4}\right) \tag{1.5}$$

Typical values of $l$ as used in this thesis lie in a range from 2 to 6. This results in a deviation of about 0.3% from the exponential decay for the worst discrete case ($n = 1$ and $l = 2$). In figure 1.2 the network is combined with voltage sources to input a pattern at the nodes. The sources do not alter neither the exponential behavior of the network nor its characteristic length because the resistance of an ideal voltage source is zero (it does not change $R_g$). The equivalent conductance of a semi-infinite network, as depicted by the dashed line in figure 1.2 for example, is:

$$G_{in} = \frac{1-\gamma}{R_n} = \sqrt{\frac{1}{R_n R_g}\left(1 + \frac{R_n}{4R_g}\right)} - \frac{1}{2R_g} \tag{1.6}$$

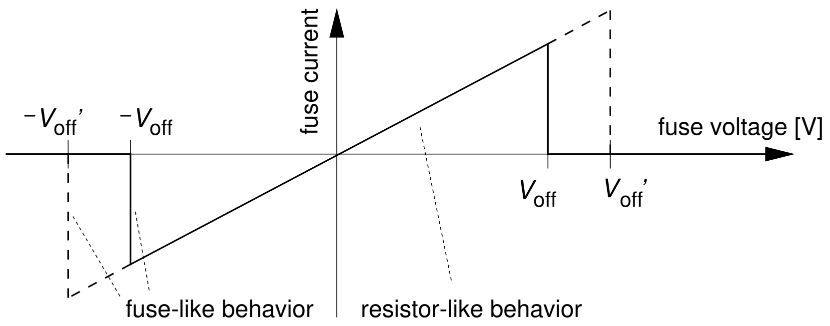



**Figure 1.3:** Characteristic curve of a resistive fuse for two different threshold voltages $V_{\text{off}}$ and $V_{\text{off}}'$. If the voltage across the fuse exceeds $V_{\text{off}}$ or $V_{\text{off}}'$, respectively, the current drops to zero.

Using this effective conductivity for each of the two semi-infinite networks merging at a node, the node voltage $V$ can be expressed by the source voltage $v$:

$$V = v\frac{1}{2G_{\text{in}}R_{\text{g}} + 1} = v\frac{1}{\sqrt{4l^2 + 1}} \tag{1.7}$$

The right part was derived by substituting equation 1.2 in 1.6. The bottom half of figure 1.2 shows the resulting node voltages if a source signal is applied at node zero. If an arbitrary input pattern is used, the principle of superposition[2] allows to calculate the node voltage from the single input responses.

Resistive networks are well suited as smoothing filters. They have the exponential decay necessary to keep the region of influence sufficiently under control. Although there are reasons to prefer a Gaussian shape for a smoothing filter[3] it may not be worth the additional implementation complexity[4] [KOB91].

### 1.2.2 Nonlinear Resistors

In contrast to smoothing noisy areas, an enhancement is needed at those parts of the image representing edges. Usually this involves some kind of spatial derivation of the gray value sequence. If only two adjacent pixels are considered, this is reduced to taking the difference between them. For a stable response, a threshold must be applied to the derivative suppressing everything below a certain level. In the resistive fuse network this is the elementary step of edge detection that is done at every node, too. The idea behind it is as follows: based on a resistive network, the connecting resistors $R_{\text{n}}$ are not longer ordinary resistor, but act like fuses in series with the original resistors. Figure 1.3 shows the characteristic curve of this so called *resistive fuse*. As long as the voltage across it is smaller than $V_{\text{off}}$, it behaves like a linear resistor. If $V_{\text{off}}$ is exceeded, the resistance becomes infinite, i.e. the fuse is 'blown'. Therefore the maximum current can be adjusted by the parameter $V_{\text{off}}$. Contrary to an ordinary fuse, this process is completely reversible.

[2] This is true as long as the resistors are linear.

[3] One reason for this is the invariance of the gauss function under the Fourier transformation. That means its behavior in the spatial and frequency domain is similar [JAE93-1].

[4] The additional complexity most likely increases the area needed for a network cell in a VLSI implementation. This reduces the number of network cells per area.

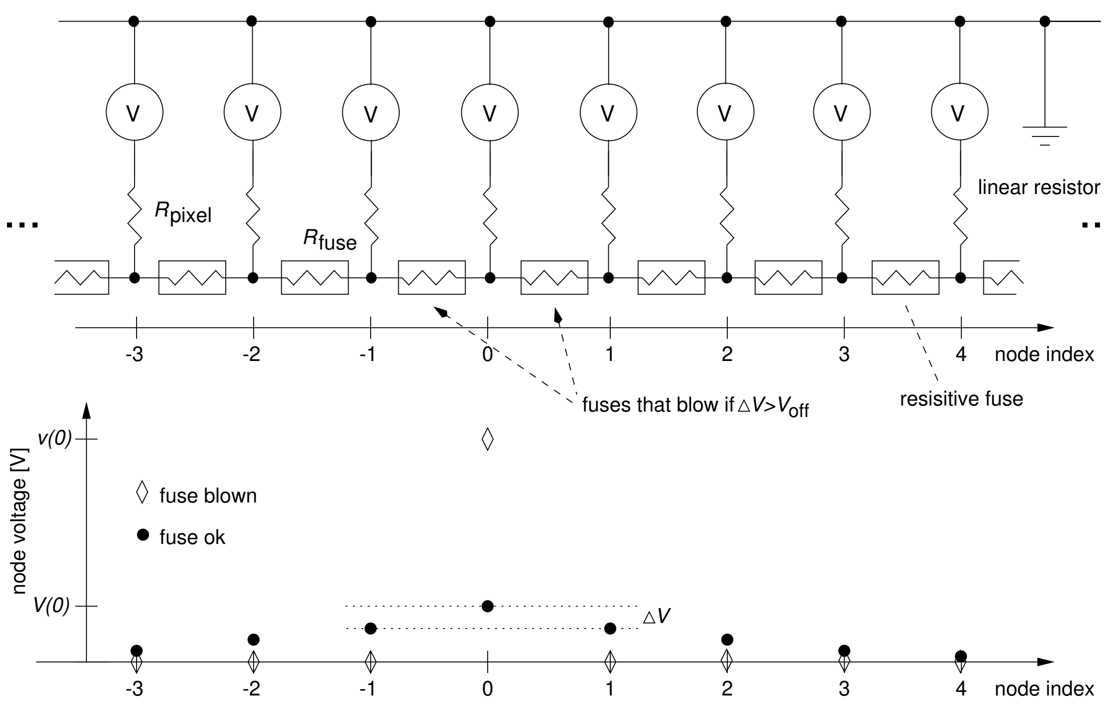



**Figure 1.4:** Response of a resistive fuse network to a single source $v(0)$ with $l = 2$. If $\Delta V$ is larger than $V_{\text{off}}$ the fuses connected to node 0 will blow.

$$R_{\text{fuse}} = \begin{cases} R_{\text{n}} & : \quad V \leq V_{\text{off}} \\ \infty & : \quad V > V_{\text{off}} \end{cases} \tag{1.8}$$

Figure 1.4 depicts the response of a network with resistive fuses to a single source voltage $v(0)$. As long as the maximum difference between the individual nodes $\Delta V$ is smaller than $V_{\text{off}}$ it behaves like an ordinary resistive network and the node values are governed by equations 1.3, 1.4 and 1.7. If $V_{\text{off}}$ is decreased below $\Delta V$ the two fuses next to the source at node zero will reach their threshold values and blow. This separates the network in three parts: two semi-infinite networks with no input signals anymore, thus all their nodes drifting to ground, and one isolated source driving node zero to the full input value of $v(0)$. Now the ratio between node zero and node one or minus one, respectively, has increased from $V(0)(1-\gamma)$ to $v(0)$. If $V$ is expressed in terms of $v$ this hysteresis between $\Delta V$ of the continuous and the interrupted network becomes:

$$h_{\text{single}} = \frac{\Delta V_{\text{fuse blown}}}{\Delta V_{\text{fuse ok}}} = \frac{\sqrt{4l^2+1}}{1-\gamma} \tag{1.9}$$

In the above example with $l = 2$ this amounts to $h_{\text{single}} = 10.6$. To switch the fuses back on, $V_{\text{off}}$ must be increased more than tenfold.

Figure 1.5 shows the same network again, but instead of a single source all sources with index $n < 1$ are at voltage $V_{\text{in}}$, those with $n \geq 1$ are zero. This corresponds to an edge in one dimension. As long as $V_{\text{off}}$ is large enough to keep all fuses in their linear region, the ratio of $\Delta V$ to $V_{\text{in}}$ is exactly the same as $V$ to $v$ in equation 1.7. If $V_{\text{off}}$ is selected to be smaller than $\Delta V$, the center

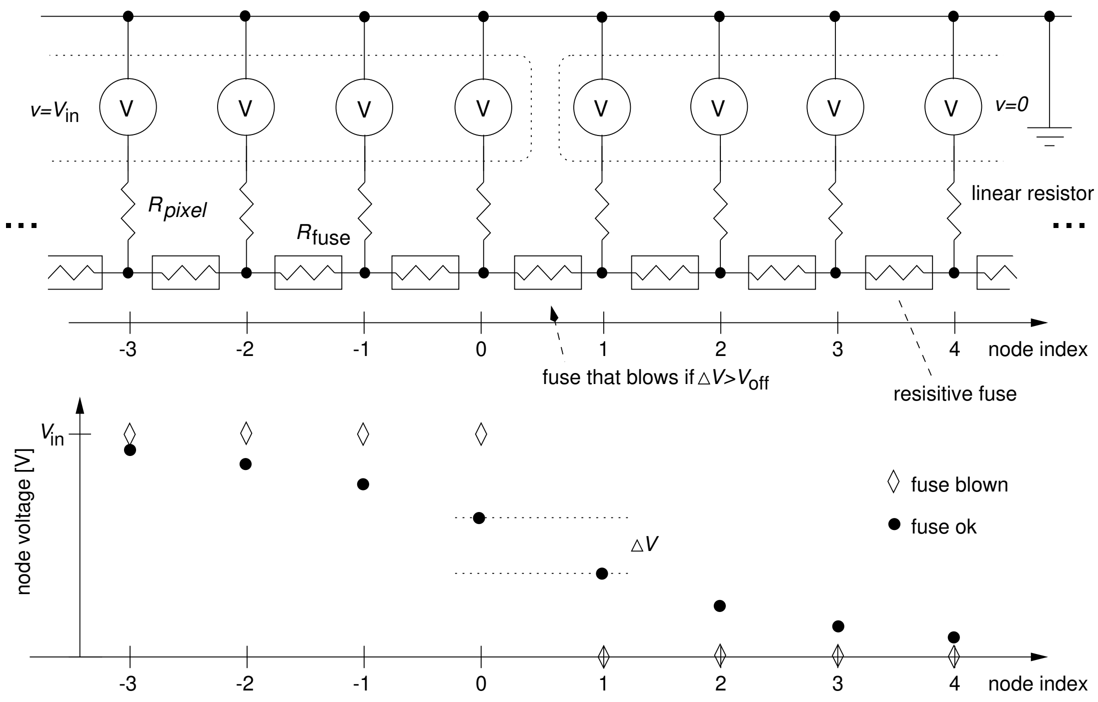



**Figure 1.5:** Response of a resistive fuse network ($l = 2$) to an edge input. The voltage sources enclosed by the dotted lines are at the same value. The source voltage jumps from $V_{\text{in}}$ to zero between node 0 and node 1. If $\Delta V$ exceeds $V_{\text{off}}$ the fuse between these nodes will blow.

fuse will blow and the network divides into two halves. The node voltage difference between them is $V_{\text{in}}$. The hysteresis in this case amounts to

$$h_{\text{edge}} = \sqrt{4l^2 + 1} \tag{1.10}$$

With $l = 2$ this results in $h_{\text{edge}} = 4.1$. Comparison of $h_{\text{single}}$ and $h_{\text{edge}}$ leads to:

$$\frac{h_{\text{single}}}{h_{\text{edge}}} = \frac{1}{1-\gamma} \tag{1.11}$$

Supposed that these different input patterns get combined, decreasing $V_{\text{off}}$ will make the edge fuse blow at a $\frac{1}{1-\gamma}$ higher voltage than the fuses next to the single source. By this way the resistive fuse network is able to distinguish objects by their size using only local comparisons. The voltage differences across the fuses are smaller for isolated input signals than for more extended ones. If a blown fuse is used as an edge indicator, by reducing $V_{\text{off}}$ the edges from larger objects will be detected first. The level of a single source must be $\frac{1}{1-\gamma}$ higher to produce an edge output at the same $V_{\text{off}}$ level than a spatial correlated group. In table 1.1 numerical values are listed for the discussed equations in dependency of the characteristic length $l$. With increasing $l$ the suppression of small structures $\frac{1}{1-\gamma}$ gets stronger. Also the ratio between node and source voltages $\left(1/\sqrt{4l^2+1}\right)$ decreases. This sets a limit for $l$ in any implementation, because voltage resolution and noise level must also be improved.

| $l$ | $\frac{1}{\sqrt{4l^2+1}}$ | $\gamma$ | $\frac{1}{1-\gamma}$ |
|---|---|---|---|
| 1 | 0.45 | 0.38 | 1.6 |
| 2 | 0.24 | 0.61 | 2.6 |
| 4 | 0.12 | 0.78 | 4.5 |
| 10 | 0.05 | 0.91 | 10.5 |

**Table 1.1:** Numerical values for different characteristic lengths.

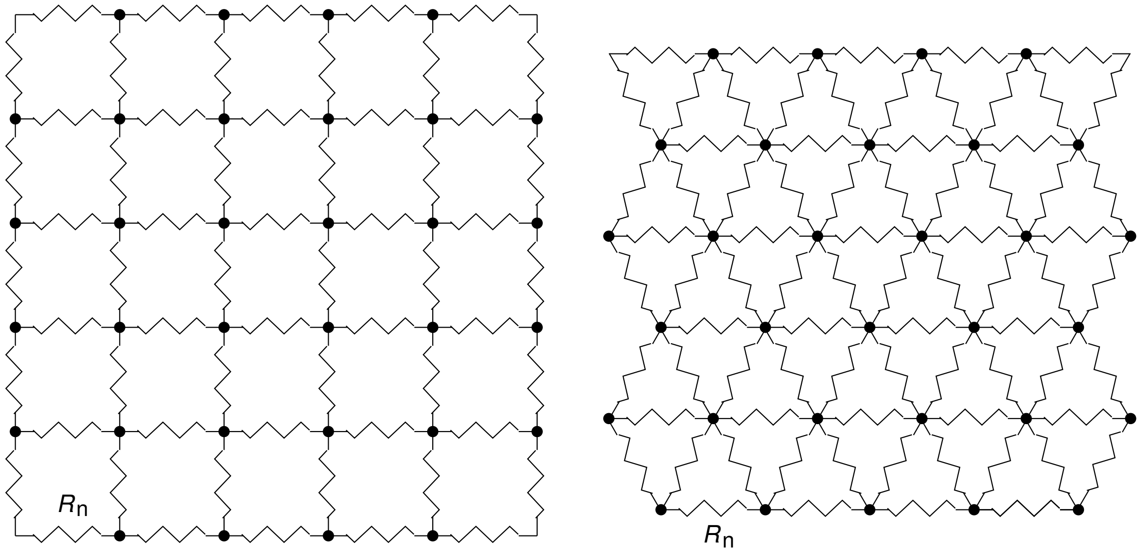



**Figure 1.6:** The two most commonly used topologies for a discrete resistive network in two dimensions. Left: square, right: hexagonal. $R_\mathrm{n}$ denotes the resistance between neighboring nodes.

### 1.2.3 Two-Dimensional Resistive Networks

In two dimensions an exact solution is known for the continuous case, too [MEA89]. The discrete network is replaced by a resistive sheet with a resistance of $\rho$ ohms per square and a conductivity to ground $\sigma$ per unit area. The equation for the radial decay of a source signal at the origin is the *modified Bessel function*, $K_0$:

$$V(r) = V_0 K_0(\alpha r) \tag{1.12}$$

with

$$\alpha = \frac{1}{l} = \sqrt{\rho\sigma} \tag{1.13}$$

Because of the divergence of the Bessel function at radii $\ll 1$, $V_0$ is not the value at $\alpha r = 0$ but at $\alpha r \approx 0.458$ where $K_0 = 1$. Unlike the one-dimensional case there is more than one possibility for a discrete network in two dimensions. Figure 1.6 shows the two most commonly used resistor topologies, hexagonal and square. Usually the hexagonal is considered as the better one, because it is closer to the rotational symmetry of the continuous resistive sheet. On the other hand it is harder to realize in VLSI techniques and combined with resistive fuses, more fuses per image pixel are

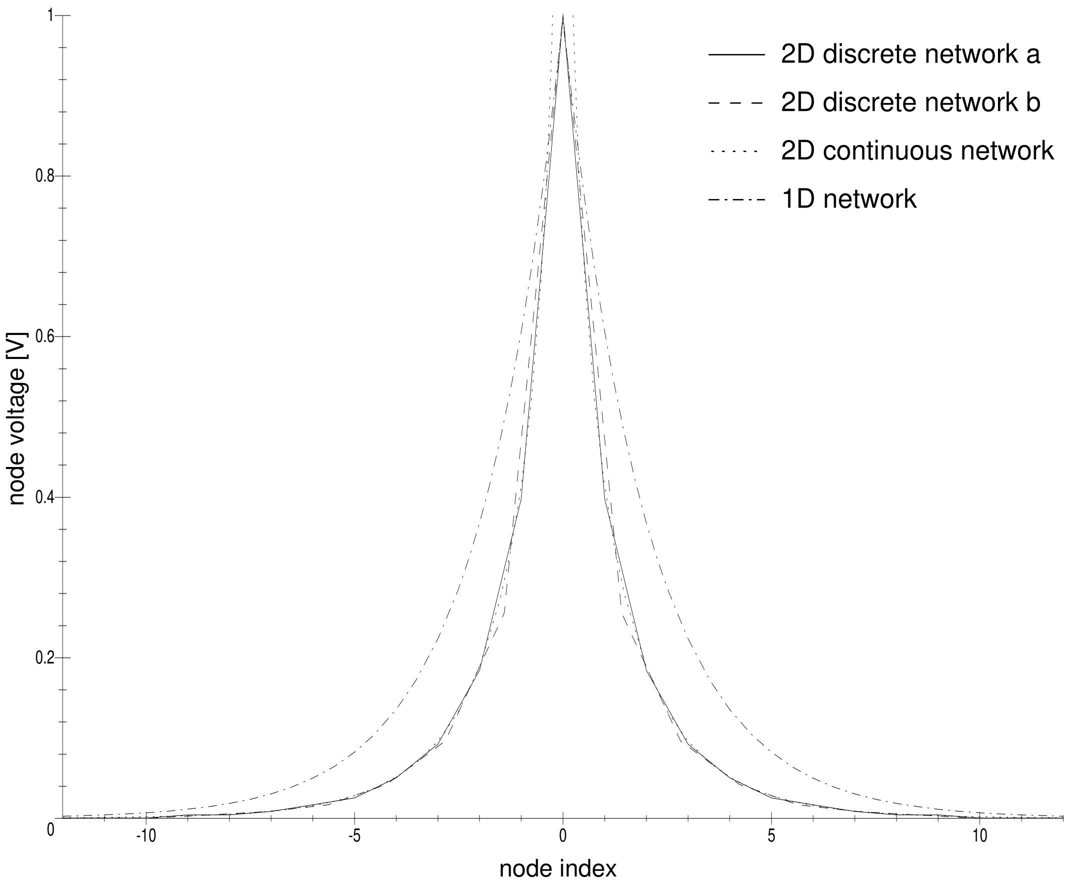



**Figure 1.7:** Comparison of discrete and continuous case in two dimensions.

needed. This makes the array roughly $3/2$ larger than the square one at a given resolution. Also most commercially available image sensors are using rectangular geometry, making it harder to connect them to a hexagonal instead of a square network. Because area saving is a major constraint for integrated circuits the square network was chosen for this thesis. The resistors between each node and ground, $R_{\mathrm{g}}$, are omitted for clarity in figure 1.6. Compared to the one-dimensional discrete network the number of network resistors $R_{\mathrm{n}}$ per node has doubled. Considering a square with $n$ nodes, $nR_{\mathrm{n}}$ is the resistance along one path, $n$ paths lie in parallel. This results in an effective resistance of $R_{\mathrm{n}}$, independent of the size $n$ of the square. Thus the resistance per square is $R_{\mathrm{n}}$ and the formula for the characteristic length stays the same as for the one-dimensional discrete network:

$$l = \sqrt{\frac{R_{\mathrm{g}}}{R_{\mathrm{n}}}}. \tag{1.14}$$

In figure 1.7 the discrete square network is compared to the continuous one for $l = 2$ and a single source at the origin. The discrete network data was generated by a numerical simulation developed within the scope of this thesis. It uses the same discrete time approach as the switched capacitor circuit described in section 1.3.1. Therefore it is explained afterwards in section 1.4.

The normalized node voltage is plotted versus the node number along a straight cut through the two-dimensional array. The discrete network has a rotational symmetry only for angles that are multiples of 90°. For this reason two different cuts are plotted. The one marked with 'a' is along the connecting resistors, 'b' has an angle of 45° to 'a'. For comparison with the preceding section the one-dimensional case is also included. The deviation from the continuous case is higher for direction 'b' than for 'a'. Also the signal declines faster in two dimensions than in one because it spreads across an area instead of a line, i.e. the signal loss to ground increases with $r^2$ instead of $r$. In the simulation of the two-dimensional network the source voltage is 10.8 times higher than the resulting node voltage. This is more than twice the source/node ratio of the one-dimensional case.

### 1.2.4 Edge Detection with Two-Dimensional Resistive Fuse Networks

To detect edges in two dimensions, the network resistors $R_{\text{n}}$ are replaced with resistive fuses, $R_{\text{fuse}}$ and an individual source is added at each node to input two-dimensional pictures into the array ($R_{\text{g}}$ gets renamed to $R_{\text{pixel}}$ to reflect this change). Figure 1.8 depicts this network structure. For the resistive fuse an implementation with standard electronic building blocks is shown. Two operational amplifiers with gain one calculate the difference between the left and right node voltages. Each comparator (operational amplifier with gain $\infty$) will close the switch to $R_{\text{fuse}}$ if this difference is smaller than $V_{\text{off}}$. Together the switches form a logical *and*: current will flow only if $l - r < V_{\text{off}}$ and $r - l < V_{\text{off}}$, thus realizing the characteristic curve of the resistive fuse (see figure 1.3). The *edge* output reflects the state of the fuse: if any of the switches is open it switches to a logical one, signaling a blown fuse.

Figure 1.9 shows more details about the behavior of the two-dimensional network. At the bottom the image data are depicted, consisting of a half-plane to the left, a straight line in the middle and a single point near the right edge. The horizontal cut through this point is drawn above, with the node voltage on the ordinate. The dark pixels are mapped to a source voltage of 0, the bright ones to 1. If the fuses are blown all node levels will be equal to the according source voltages else the voltage reduction caused by the network smoothing will be different depending on the size of the structure.

In Figure 1.9 all the fuses are either blown or intact. For image processing applications $V_{\text{off}}$ is selected in a way that a blown fuse represents an edge in the source image. In the mentioned figure for example the single point could be suppressed by adjusting $V_{\text{off}}$ to 0.1. In figure 1.10 the image consists of a dark bar against a bright background. At the upper half Gaussian distributed noise with a sigma of 25% in respect to the bar-to-background ratio is added. The two small images at the top show the node voltages. The left one represents the case that all fuses are blown. The right one depicts a $V_{\text{off}}$ adjustment which smoothes the noise but keeps the edges of the bar. In the diagram below cuts are plotted for both cases. For reference it contains also a cut through the noise-free node image. Figure 1.11 shows the histograms of both situations. The smoothing process reduces the width of the peaks and separates them.

To close this section the effect of a resistive fuse network on a real world scenery is displayed. A photograph of a street is used as input pattern. It is depicted at the top of figure 1.12. The image in the middle shows the node voltages for a $V_{\text{off}}$ level of about $1/60$ of the total dynamic range of the source data. At the bottom the blown fuses are printed as short black lines along the pixel boundaries. They apparently represent the edges of the picture. In the node image the smoothing process in the areas between the blown fuses is visible. It gives the image a 'painted' appearance.

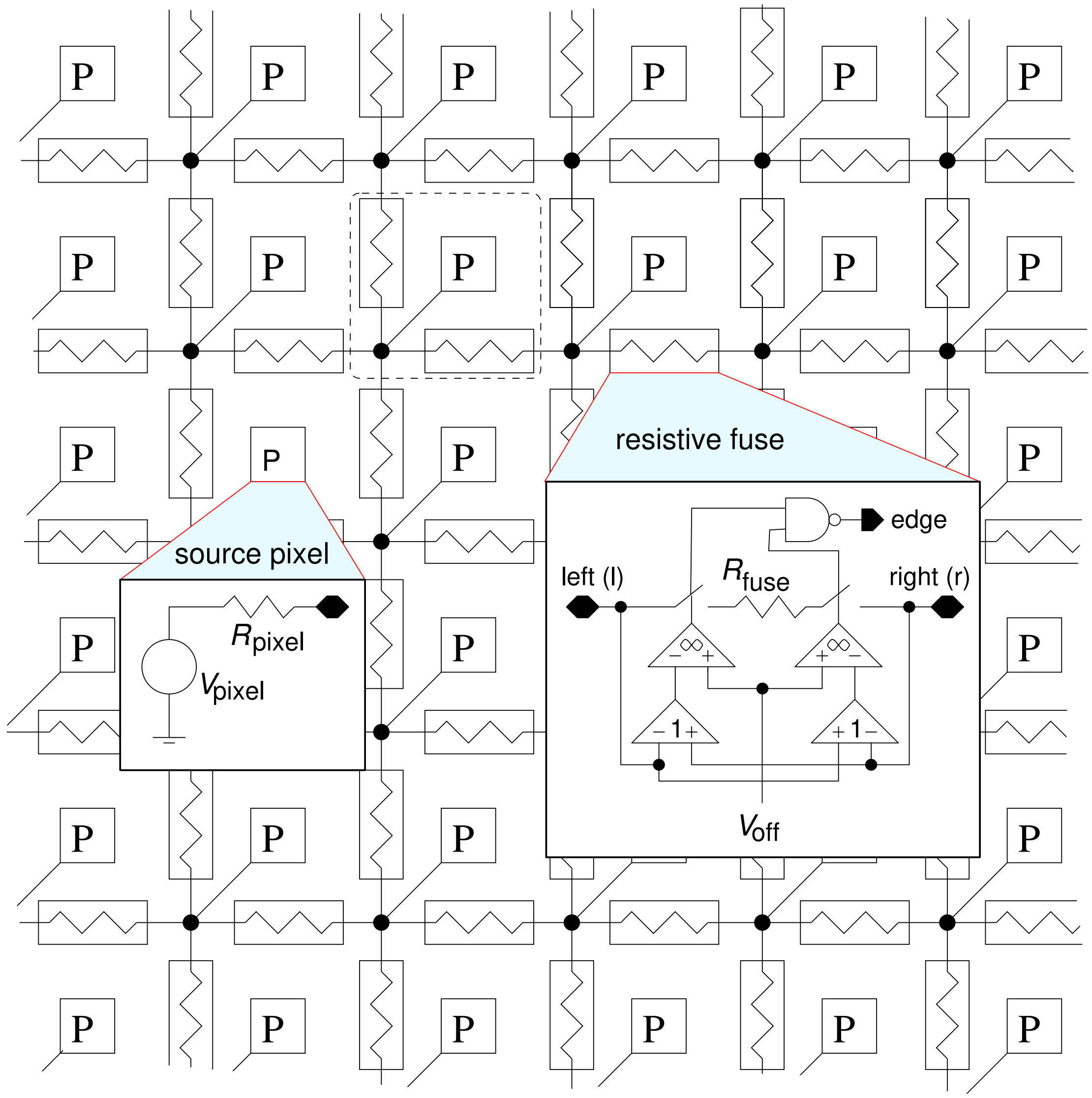



**Figure 1.8:** Cutout of a square resistive fuse network. The solid boxes show the circuit diagrams for a fuse and a pixel. The dashed box depicts one elementary cell consisting of two fuses and one pixel.

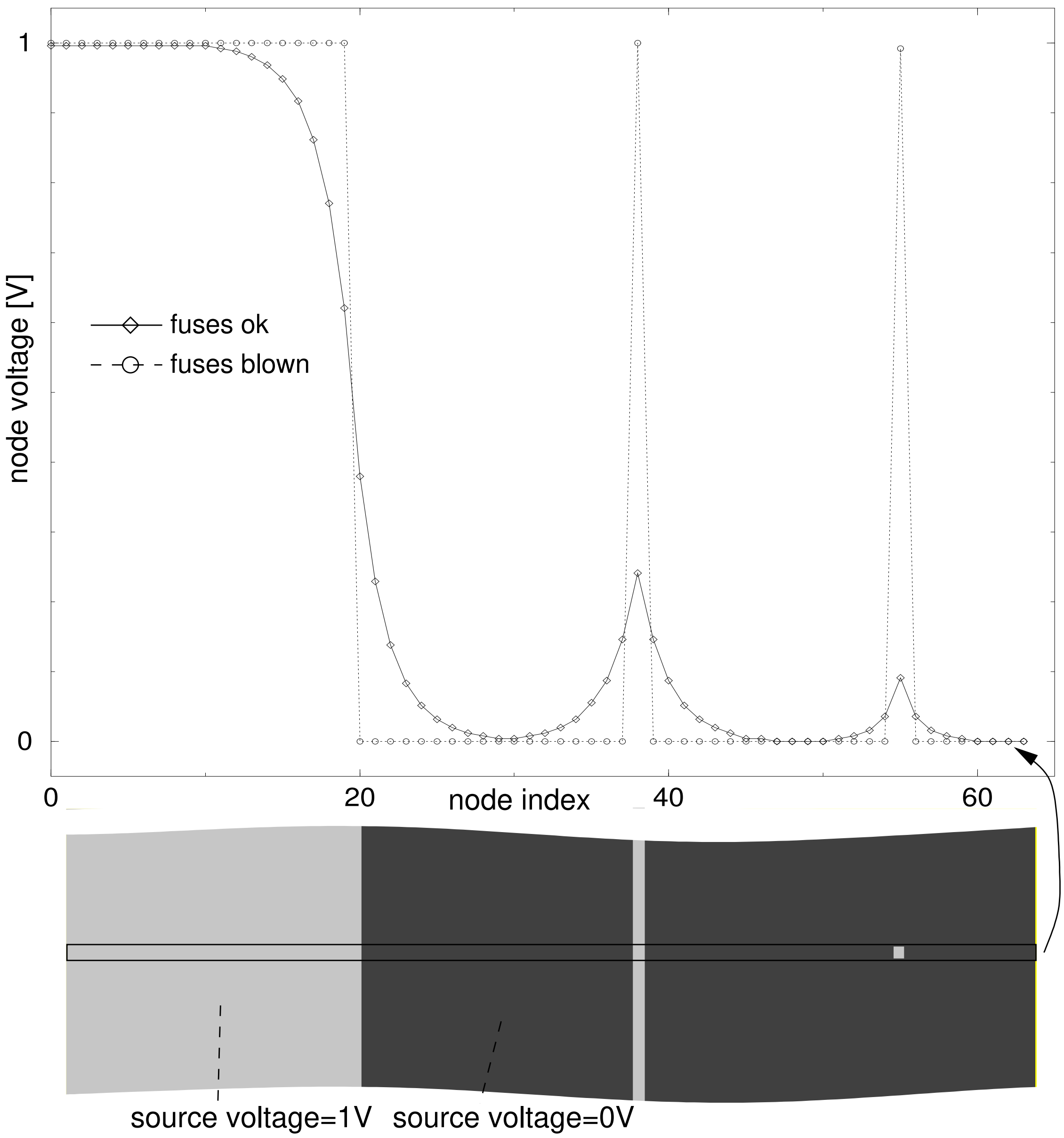



**Figure 1.9:** Smoothing of differently sized structures in a two-dimensional network. At the bottom a cutout of the source image is shown. The node voltages in the graph show the network's response to this image (along the cut depicted by the black box) for the two cases that all fuses are either blown (dashed line) or intact (solid line). It can be seen that the maximum node voltage difference for the case that the fuses are intact is correlated with the spatial size of the according image structure.

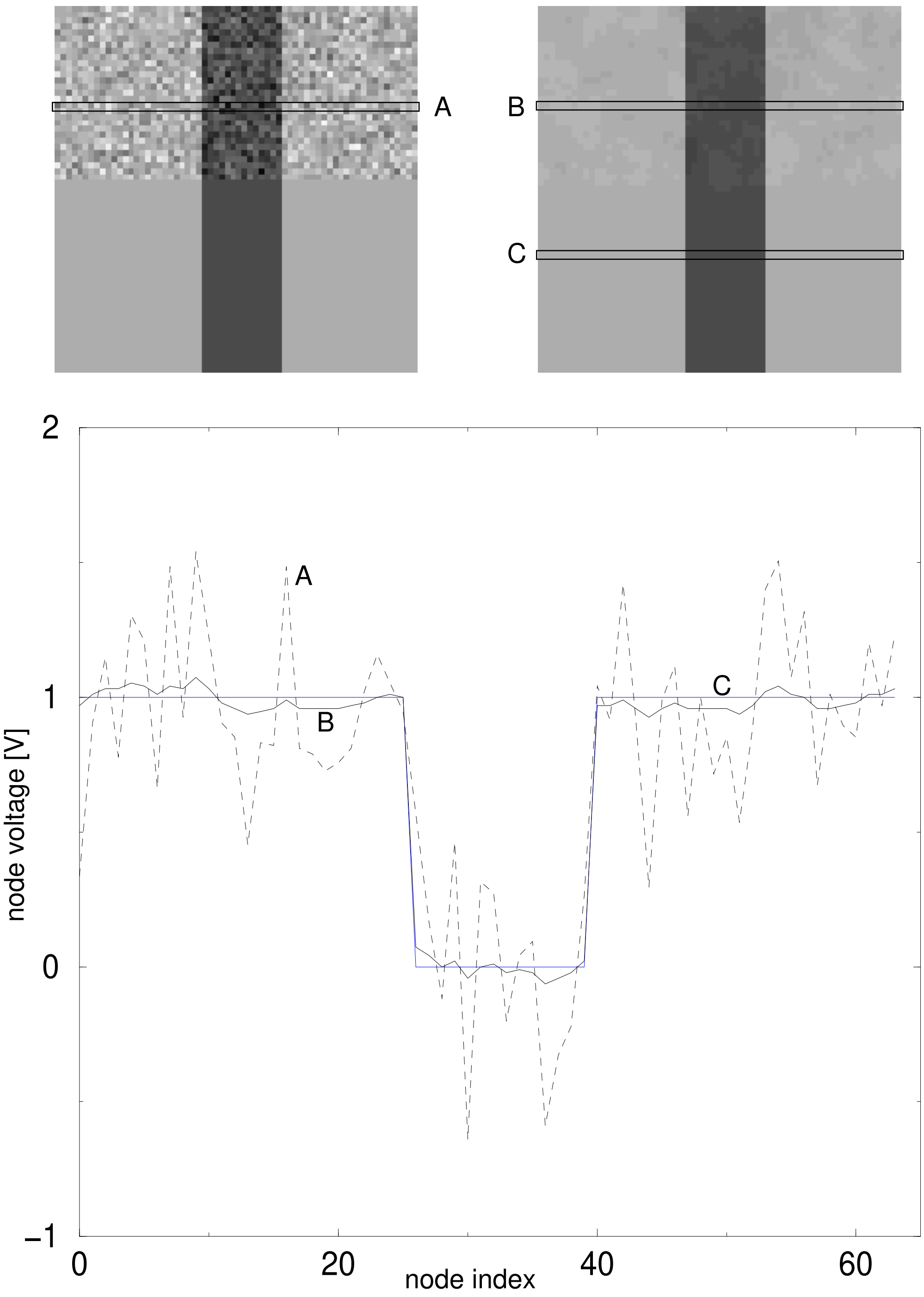



**Figure 1.10:** Noise suppression capabilities of the resistive fuse network. Two different node voltage images are shown at the top. In the bottom half of each image all fuses are blown, i.e. the node voltages reflect the source ones. The black bar is at 0 V and the surrounding at 1 V. In the top half noise has been added to the source image. In the image on the right the noise is reduced by the smoothing action of the resistive fuse network while the edges of the bar are preserved. The graph shows the node voltages for both cases together with the source image along the cuts depicted by the black boxes.

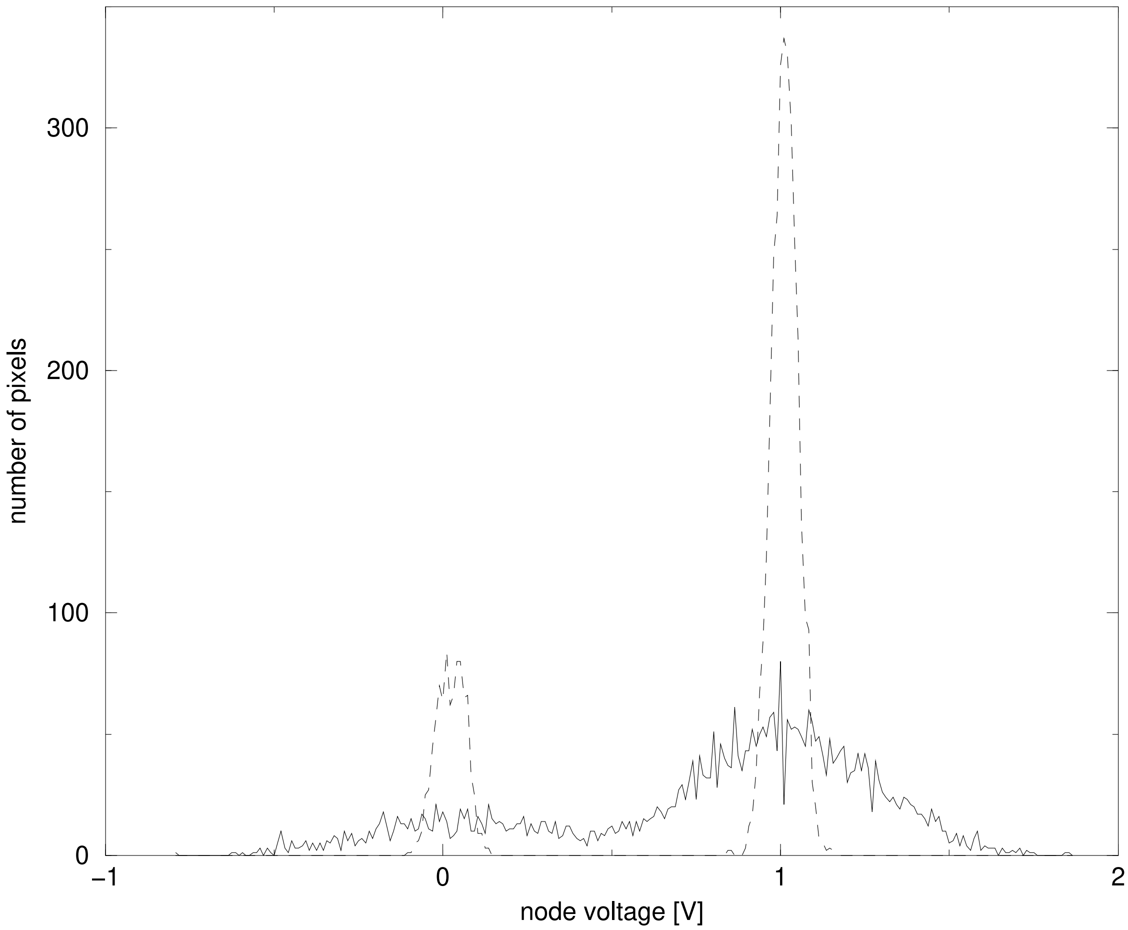



**Figure 1.11:** Histograms of the node voltages of the previous figure. Solid line: left image, all fuses are blown; dashed line: right image, fuses are partially blown (the histograms contain only the values from the top half of each image).

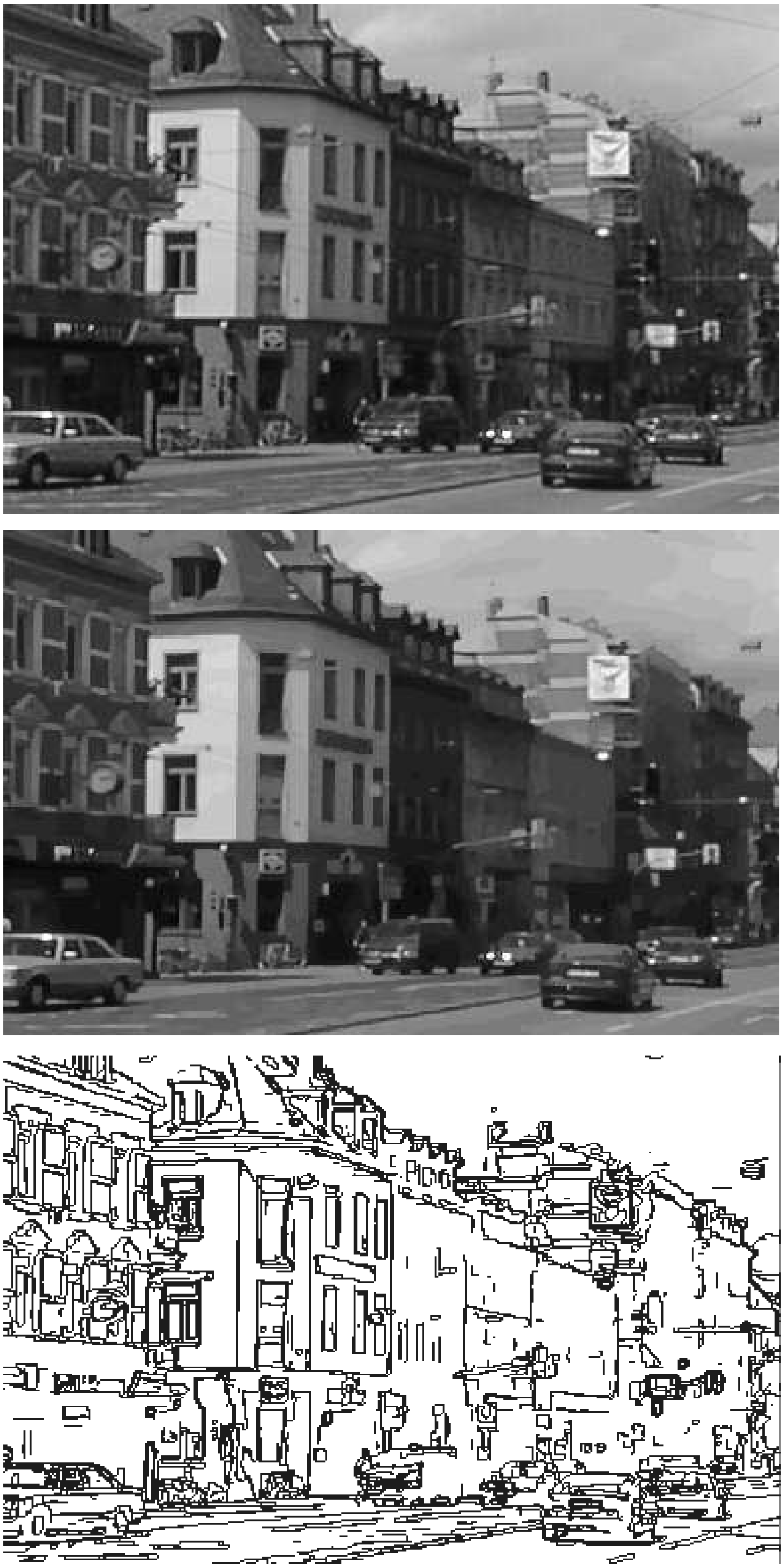


**Figure 1.12:** Real wold photograph processed by a resistive-fuse network. From top to bottom: original image, node image and blown fuses. The resolution is 384 $\times$ 288 pixel.

## 1.3 The Switched Capacitor Resistive Fuse Circuit

The major goal of this thesis was the analog realization of a resistive fuse network in a standard CMOS process in a way that allows the exploitation of the constant shrinking feature sizes caused by the rapid advances in semiconductor technology. If the array structure from figure 1.8 was to be implemented in a fully parallel analog way, a transistor level circuit must be developed for the fuse and the pixel voltage source. In addition to the design rules of the semiconductor manufacturing process three major constraints must be met: low area usage, low power consumption and low fixed pattern noise[5]. Clearly the direct realization of the circuit in figure 1.8 does not comply well with the area constraint. Eight operational amplifiers are needed for the two fuses that belong to every input pixel. In addition there are no linear resistors with a high resistance per area in standard CMOS processes. In the AMS[6] 0.6 $\mu$m process used for this thesis [AMS98-1, AMS98-2] the poly layer has a typical resistance of 33 $\Omega/\Box$[7]. If the current through the pixel resistor should be limited to 10 $\mu$A at a source voltage of 1 V, it needs 3030 squares which consume at least an area of $70 \times 70$ $\mu\mathrm{m}^2$ [8]. To keep both area and power consumption low active resistors are needed, either by using the ohmic region of the transistor characteristic curve or by switching capacitors [GEI90-2].

In [YU92, HAR89] continuous time implementations of resistive fuse circuits are presented. They use between 7 and 33 MOS transistors to build a single fuse. While the fuses with less transistors are good in terms of area, they are more sensitive to common mode variations and the linearity of their resistance is not very good. To keep the power consumption low those circuits usually work in weak inversion. This makes them more sensitive to threshold voltage variations caused by device mismatch than circuits biased in strong inversion[9]. Another disadvantage of continuous time implementations is their dependence on transistor properties, making it difficult to transfer those circuits to smaller feature-size processes. In the deep submicron range transistor properties change strongly with oxide thickness and doping concentrations, especially for minimum sized devices where short channel effects become dominant.

Another design goal in this thesis was the capability to read out edge information digitally from the resistive fuse array. As can be seen in figure 1.12 the information that a fuse is blown, i.e. if the voltage across it is larger than $V_{\mathrm{off}}$ or not, can be used as an indicator for an edge between the two nodes connected by that fuse. This is depicted symbolically by the NAND gate in figure 1.8. The continuous time circuits do not produce this information directly. If $V_{\mathrm{off}}$ is reached, the current through the fuse drops but it is still necessary to measure analog quantities to decide if it is blown or not.

[5] Fixed pattern noise is the spatial fluctuation in a periodic circuit like a camera or a resistive array consisting of many identical pixels. Due to the limited accuracy of the semiconductor manufacturing process device-to-device variations exist. The final pixel-to-pixel deviations are mostly the sum of a multitude of small uncorrelated device-level fluctuations. This makes it possible to treat them like normal distributed noise. Instead of varying with time, they vary with the location on the wafer. After manufacturing this pattern is stable for every chip, hence the term fixed pattern noise.

[6] Austria Mikro Systeme International AG

[7] The resistivity of a layer is given in $\Omega/\Box$ by the chip manufacturer since the resistance of a square-shaped resistor is independent of its size.

[8] This is twice the area that the complete circuit, as it is implemented in the EDDA chip, occupies.

[9] In weak inversion, i.e. the subthreshold region where $0 < V_{\mathrm{GS}} < V_{\mathrm{T}}$, the drain current is proportional to $e^{(V_{\mathrm{GS}}-V_{\mathrm{T}})/nV_{\mathrm{t}}}$ instead of $(V_{\mathrm{GS}} - V_{\mathrm{T}})^2$ in strong inversion [GEI90-3].

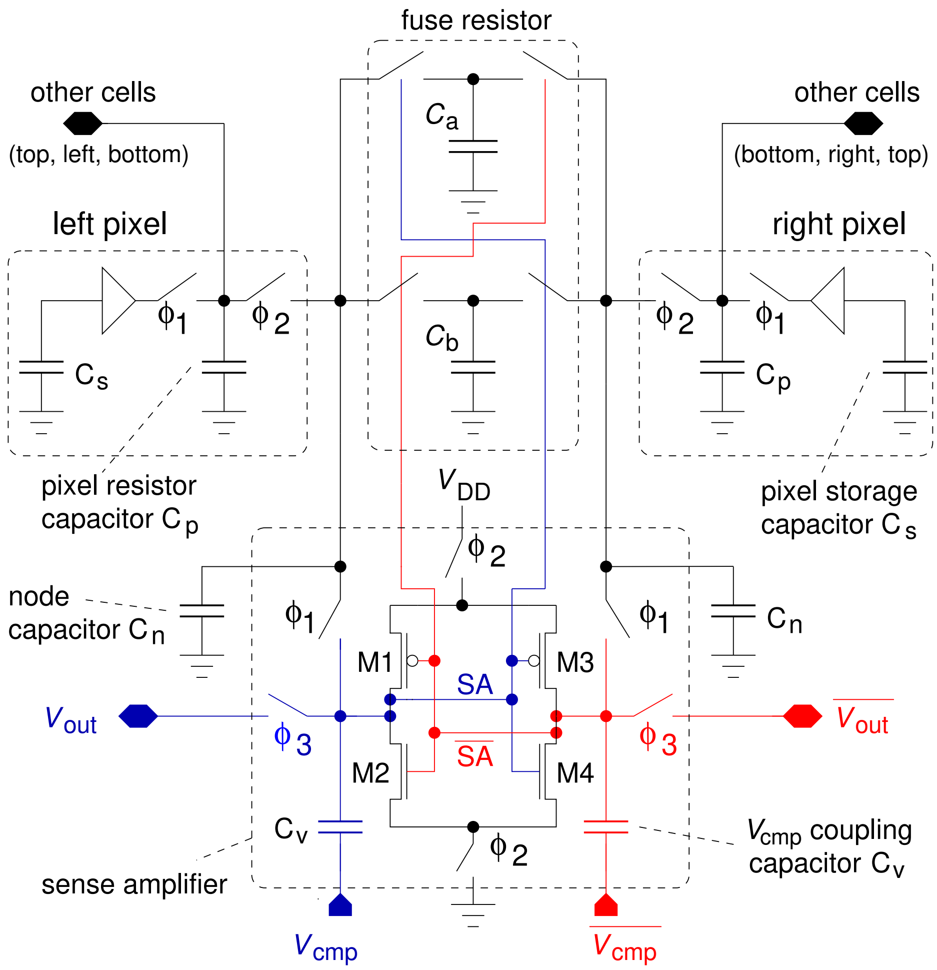



**Figure 1.13:** Circuit diagram of the switched capacitor fuse.

Therefore the approach realized in this thesis is not a continuous time one, it is based on switched capacitor techniques (SC) instead. Besides having the ability to implement accurate resistors with high resistances and low area usage, SC-circuits offer the ability to store values for short periods of time. This feature is used to reduce the number of operational amplifiers per fuse from four to one by performing some tasks sequentially in time.

### 1.3.1 Operation Principle

Figure 1.13 shows the circuit diagram of a resistive fuse as it has been developed in the scope of this thesis. Two pixels and one fuse circuit are drawn. The details of the analog storage and the readout part are omitted for simplicity. They will be covered in section 2.2. The pixel storage capacitor $C_s$ and the following buffer represent the pixel voltage source. The nodes labeled with *other cells* are connected to the three remaining neighbors of each pixel, as depicted below the arrows. The resistive fuse is shown in the middle, consisting of a resistor and a sense amplifier.

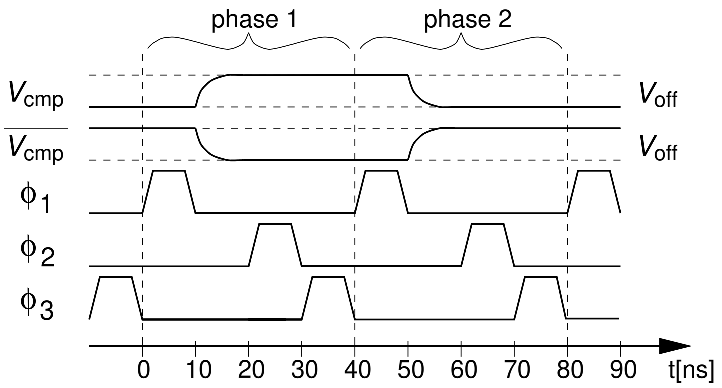



**Figure 1.14:** Timing of $\phi$1-$\phi$3 and $V_{\text{cmp}}$. One network cycle is shown. It is divided into two phases with different $V_{\text{cmp}}$ polarity.

The SC-resistor of the fuse is formed by $C_a$ and $C_b$ together with the surrounding switches. The sense amplifier is a special kind of clocked comparator [GEI90-4, PRI95], using three non-overlapping clock signals $\phi_1$ to $\phi_3$. Its core is build by the transistors M1 to M4. They are connected as a static latch (two cross coupled inverters). Two switches isolate it from the power supply lines $V_{\text{DD}}$ and ground if $\phi_2$ is inactive (low). With $\phi_1$ charge can be transferred from the inputs onto the internal nodes SA and $\overline{\text{SA}}$. This charge is stored on the gate-capacitances of M1-M4 and the $V_{\text{cmp}}$ coupling capacitors $C_{\text{v}}$, making the node voltages reflect the input voltages. By applying power (with $\phi_2$) the voltage difference between SA and $\overline{\text{SA}}$ is amplified up to $V_{\text{DD}}$ and ground. The third clock phase $\phi_3$ connects the nodes of the sense amplifier to a pair of global signals, called $V_{\text{out}}$ and $\overline{V_{\text{out}}}$. They are serving two different purposes. First they are both set to the precharge voltage to clear the sense amplifier. The second usage is to readout the state of the sense amplifier after it has made a decision. This is covered later in section 2.2.4.

Figure 1.14 shows the edge detection timing. The network cycle is split into two phases, each consisting of a single cycle of the non-overlapping clock signals $\phi_1$, $\phi_2$ and $\phi_3$. The difference between phase 1 and phase 2 is the polarity of the differential $V_{\text{cmp}}$ signals. $V_{\text{cmp}} - \overline{V_{\text{cmp}}}$ changes from $V_{\text{off}}$ to $-V_{\text{off}}$. The node voltages are compared against this difference.

At the beginning of each phase, after $\phi_3$, both nodes of the sense amplifier are at $V_{\text{precharge}}$ level. It is chosen to be approximately one transistor threshold voltage higher than the lowest node voltage. Since the fuse resistor switches are NMOS pass transistors, this ensures that no current flows after precharging the sense amplifiers. The momentary network node voltages are stored on the node capacitors $C_{\text{n}}$.

Each phase starts with $\phi_1$. The charge on the left and right node capacitors $C_n$ ($Q_l$, $Q_r$) is distributed between the gate capacitances of the sense amplifier ($C_g$) and the $V_{cmp}$ capacitor $C_v$. The differential input voltage of the sense amplifier $\Delta V = V_{\text{sa}} - \overline{V_{\text{sa}}}$ becomes ($C_{\text{sa}} = C_{\text{g}} + C_{\text{v}}$, $Q_{\text{sa}} = C_{\text{sa}} \cdot V_{\text{precharge}}$):

$$\Delta V = \frac{Q_{\text{l}} + Q_{\text{sa}}}{C_{\text{n}} + C_{\text{sa}}} - \frac{Q_{\text{r}} + Q_{\text{sa}}}{C_{\text{n}} + C_{\text{sa}}} = \frac{Q_{\text{l}} - Q_{\text{r}}}{C_{\text{n}} + C_{\text{sa}}} \tag{1.15}$$

This is independent of both the precharge voltage and any common node voltage level.

After $\phi_1$ $V_{cmp}$ changes from ground to $V_{off}$ or vice versa, depending on the phase (see figure 1.14), $\overline{V_{cmp}}$ is the complement, changing from $V_{off}$ to ground in phase 1. For the capacitive divider formed by $C_g$ and $C_v$ the charge sum at the common node cannot change if it is isolated. Therefore the charges at both capacitors must be equal:

$$V_g C_g = (V_{off} - V_g) C_v \tag{1.16}$$

$$\lambda_{off} = \frac{V_g}{V_{off}} = \frac{C_v}{C_g + C_v} \tag{1.17}$$

Using this result the new differential input voltage for the sense amplifier $\Delta V'$ can be calculated:

$$\Delta V' = \Delta V + 2\lambda_{off} \cdot \begin{cases} V_{off} & \text{in phase 1} \\ -V_{off} & \text{in phase 2} \end{cases} \tag{1.18}$$

The factor of two stems from the fact that the total change in $V_{cmp} - \overline{V_{cmp}}$ is two times $V_{off}$ (both $V_{cmp}$ and $\overline{V_{cmp}}$ change by $V_{off}$). In phase 1 $\Delta V$ gets enlarged, in phase 2 it is reduced.

With $\phi_2$ the sense amplifier is activated and if $\Delta V' > 0$, $V_{sa}$ will go to $V_{DD}$ and $\overline{V_{sa}}$ to ground. This connects $C_a$ to the left and $C_b$ to the right node. If $\Delta V' < 0$ the output will be inverted, connecting $C_a$ to the right and $C_b$ to the left. Table 1.2 shows all four possible combinations of $\Delta V'$ and phase. The forth one does not occur if $V_{off} > 0$. If $|\Delta V|$ is larger than the second addend of equation 1.18, the capacitors $C_a$ and $C_b$ will get connected to the same side in both phases, disabling any charge transport across the fuse. In the other case $V_{cmp}$ and $\overline{V_{cmp}}$ dominate the sense amplifier, making it switch to one side in phase 1 and to the other in phase 2. This leads to $C_a$ and $C_b$ acting like a SC-resistor connecting the two nodes. Figure 1.15 illustrates this behavior for the case that the left node voltage is larger than the right one, i.e. $\Delta V > 0$. At the top part the fuse resistor is shown for a blown fuse, at the bottom the fuse is intact. Phase 1 is depicted left, phase 2 right. The sign of $\Delta V'$ determines whether $V_{sa}$ will go to $V_{DD}$ or ground. If the sign is the same in phase 1 and phase 2, the sense amplifier switches both times to the same side. $C_a$ gets charged to the voltage of the left node, $C_b$ to the one of the right. If the voltage difference between the nodes is small, the sign of $\Delta V'$ depends only on the phase, i.e. $2\lambda_{off} V_{off} > |\Delta V|$ in both phases. In this case the capacitors are alternately connected either to the left or the right node, forming a SC-resistor.

The pixel capacitor and four fuse capacitors are connected in parallel after the sense amplifiers have made their decision. The new node voltage ($V_{node}$) is stored on $C_n$ for the next phase. The actual phase ends with $\phi_3$. The sense amplifier gets precharged by connecting both nodes to the $V_{out}/\overline{V_{out}}$ lines, which are held at $V_{precharge}$ during the whole cycle. This disconnects $C_a$ and $C_b$ from both nodes.

### 1.3.2 Fuse and Source Resistors

The effective resistance of the fuse resistor can be calculated as follows:

$$I_{eff} = \frac{Q}{t} = \Delta V_{node}(C_a + C_b) f_{cycle} \tag{1.19}$$

$$R_{fuse} = \frac{\Delta V_{node}}{I_{eff}} = \frac{1}{(C_a + C_b) f_{cycle}} = \frac{1}{2 C_a f_{cycle}} \tag{1.20}$$

| $\Delta$V′ | | C$_a$ | | fuse | nodes |
|---|---|---|---|---|---|
| phase 1 | phase 2 | phase 1 | phase 2 | | |
| $< 0$ | $< 0$ | right | right | blown | $r > l$ |
| $> 0$ | $> 0$ | left | left | blown | $l > r$ |
| $> 0$ | $< 0$ | left | right | ok | $\lvert l - r\rvert < 2\lambda_{\text{off}} V_{\text{off}}$ |
| $< 0$ | $> 0$ | right | left | does not happen | |

**Table 1.2:** Different $\Delta V'$ combinations and resulting states of the fuse.

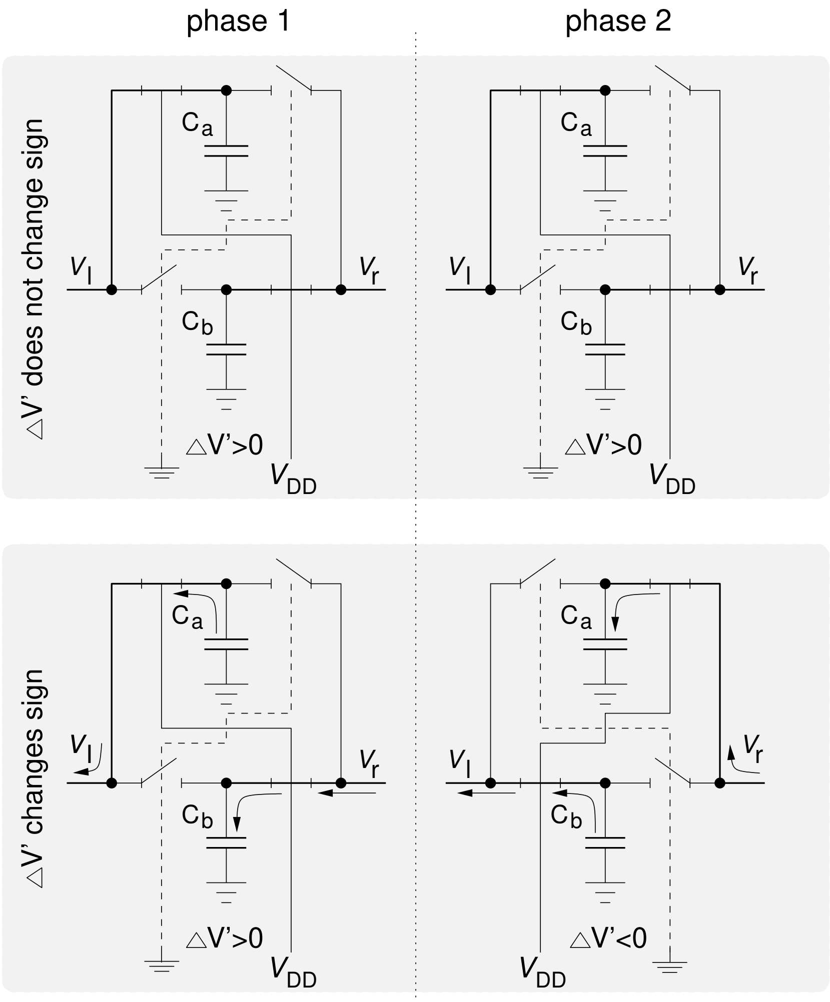



**Figure 1.15:** Illustration of the two different states of the fuse. Top: fuse blown, no current flows from one side to the other, Bottom: fuse ok, charge is transported across the capacitors.

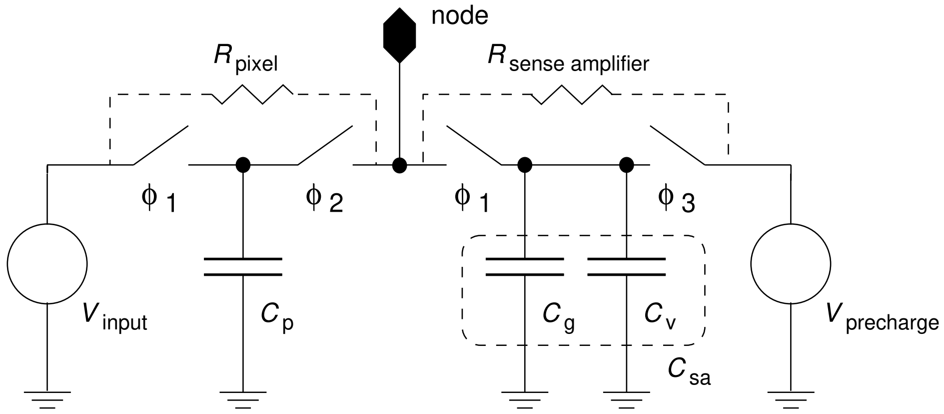



**Figure 1.16:** Two voltages sources act in parallel due to the sense amplifier capacitances.

where $\Delta V_{\text{node}}$ is the node voltage difference and $f_{\text{cycle}}$ is the network cycle frequency (one cycle consists out of phase 1 and phase 2). Since $C_{\text{a}}$ and $C_{\text{b}}$ are equal, $C_{\text{a}}+C_{\text{b}}$ was substituted by $2C_{\text{a}}$. The pixel resistor is clocked by $\phi_1$ and $\phi_2$, thus its clock frequency is $2f_{\text{cycle}}$:

$$R_{\text{pixel}} = \frac{1}{2f_{\text{cycle}}C_{\text{p}}} \tag{1.21}$$

The characteristic length of the SC-capacitor square resistive fuse array is therefore:

$$l = \sqrt{\frac{R_{\text{pixel}}}{R_{\text{fuse}}}} = \sqrt{\frac{C_{\text{a}}}{C_{\text{p}}}} \tag{1.22}$$

This equation holds as long as $C_{\text{p}} \gg C_{\text{sa}}$. If this is not the case the effective pixel capacitance becomes $C_{\text{p}}' = C_{\text{p}} + C_{\text{sa}}$ and the characteristic length decreases. Figure 1.16 shows how the precharged sense amplifier capacitances $C_{\text{v}}$ and $C_{\text{g}}$ act like a second voltage source in parallel to the pixel voltage source. By using Kirchhoff's laws the node voltage can be calculated:

$$\frac{V_{\text{node}} - V_{\text{input}}}{R_{\text{pixel}}} = \frac{V_{\text{precharge}} - V_{\text{input}}}{R_{\text{pixel}} + R_{\text{sa}}} \tag{1.23}$$

$$V_{\text{node}} = \lambda_{\text{sa}} V_{\text{precharge}} + V_{\text{input}}(1 - \lambda_{\text{sa}}) \tag{1.24}$$

with

$$\lambda_{\text{sa}} = \frac{R_{\text{pixel}}}{R_{\text{pixel}} + R_{\text{sa}}} \tag{1.25}$$

Substituting $1/fC$ for $R$ yields:

$$\lambda_{\text{sa}} = \frac{C_{\text{g}} + C_{\text{v}}}{C_{\text{p}} + C_{\text{g}} + C_{\text{v}}} \tag{1.26}$$

Two conclusions can be drawn out of this: first, by changing $V_{\text{precharge}}$ the node voltage gets shifted by $\lambda_{\text{sa}} V_{\text{precharge}}$ and second, the input voltage range of the sense amplifier is reduced by a factor of $\lambda_{\text{sa}}$. This affects any area-conserving implementation of this circuit in silicon because $C_{\text{g}}$ and $C_{\text{v}}$ cannot be kept $\ll C_{\text{p}}$.

### 1.3.3 Temporal Behavior

The reason why $C_a$ and $C_b$ are separate is not a fundamental one. They could be replaced by a single capacitor $C_{fuse}$ with a capacitance of $2C_a$ to build the fuse without changing the characteristic length. This would save two switches. However the usage of two capacitors has one big advantage: it makes the sense amplifier circuit symmetric with respect to the nodes. In $\phi_2$ this becomes important because the charge from the pixel capacitor $C_p$ and the four neighboring fuse capacitors is shared at the node. If a single capacitor is used per fuse, $V_{node}$ would not only depend on the different $Q_{fuse\,i}$ but also on the number of sense amplifiers which have switched their capacitor to the side facing the node under consideration (side SA in the following equations):

$$V_{node} = \frac{Q_n + Q_p + \sum_{i=1}^{4} O_{sa\,i} Q_{fuse\,i}}{C_n + C_p + C_{fuse} \sum_{i=1}^{4} O_{sa\,i}} \tag{1.27}$$

The sense amplifier output vector $O_{sa\,i}$ is defined by:

$$O_{sa\,i} = \begin{cases} 1 & \text{if } V_{sa\,i} = V_{DD} \text{ and } \overline{V_{sa\,i}} = \text{ground} \\ 0 & \text{if } V_{sa\,i} = \text{ground and } \overline{V_{sa\,i}} = V_{DD} \end{cases} \tag{1.28}$$

For two capacitors $C_a$ and $C_b$ the $V_{node}$ equation is:

$$V_{node} = \frac{Q_n + Q_p + \sum_{i=1}^{4} (O_{sa\,i} Q_{a\,i} + (1 - O_{sa\,i}) Q_{b\,i})}{C_n + C_p + 4C_a} \tag{1.29}$$

In comparison to eq.1.27 the denominator is now independent of $O_{sa\,i}$, i.e. the node capacitance is constant. Especially if a fuse is blown this avoids permanent differences in the total node capacitance between adjacent nodes. Every time the input voltage for a pixel changes the fuse resistor capacitance $2C_a$ must be charged to the new node voltage. This takes several cycles since $l > 1$ implies $C_p < C_a$ (see eq.1.22). Using two capacitors for the fuse resistor makes this number independent of the fuse state. Thereby the switched capacitor array gets a time-constant $\tau_{SC}$ that can be derived from the following equation for the node voltage in the continuous case:

$$V_{node} = V_{pixel} \left(1 - e^{-\frac{t}{R_{pixel}(C_n + 4C_a)}}\right) \tag{1.30}$$

Substituting $R_{pixel}$ by eq.1.21 yields:

$$\begin{aligned} V_{node} &= V_{pixel} \left(1 - e^{-t\frac{2f_{cycle}C_p}{C_n + 4C_a}}\right) \\ &= V_{pixel} \left(1 - e^{-n_{cycle}\frac{2C_p}{C_n + 4C_a}}\right) \qquad (1.31) \\ &= V_{pixel} \left(1 - e^{-\frac{n_{cycle}}{\tau_{SC}}}\right) \qquad (1.32) \end{aligned}$$

with $n_{cycles}$ being the number of cycles elapsed since $t = 0$.

$$\tau_{SC} = \frac{C_n + 4C_a}{2C_p} \simeq \frac{2C_a}{C_p} = 2l^2 \tag{1.33}$$

The right part uses the fact that $C_n \ll 4C_a$. Figure 1.17 shows the temporal behavior of the SC resistive fuse network. The input patterns are the same as those in figure 1.9. Instead of

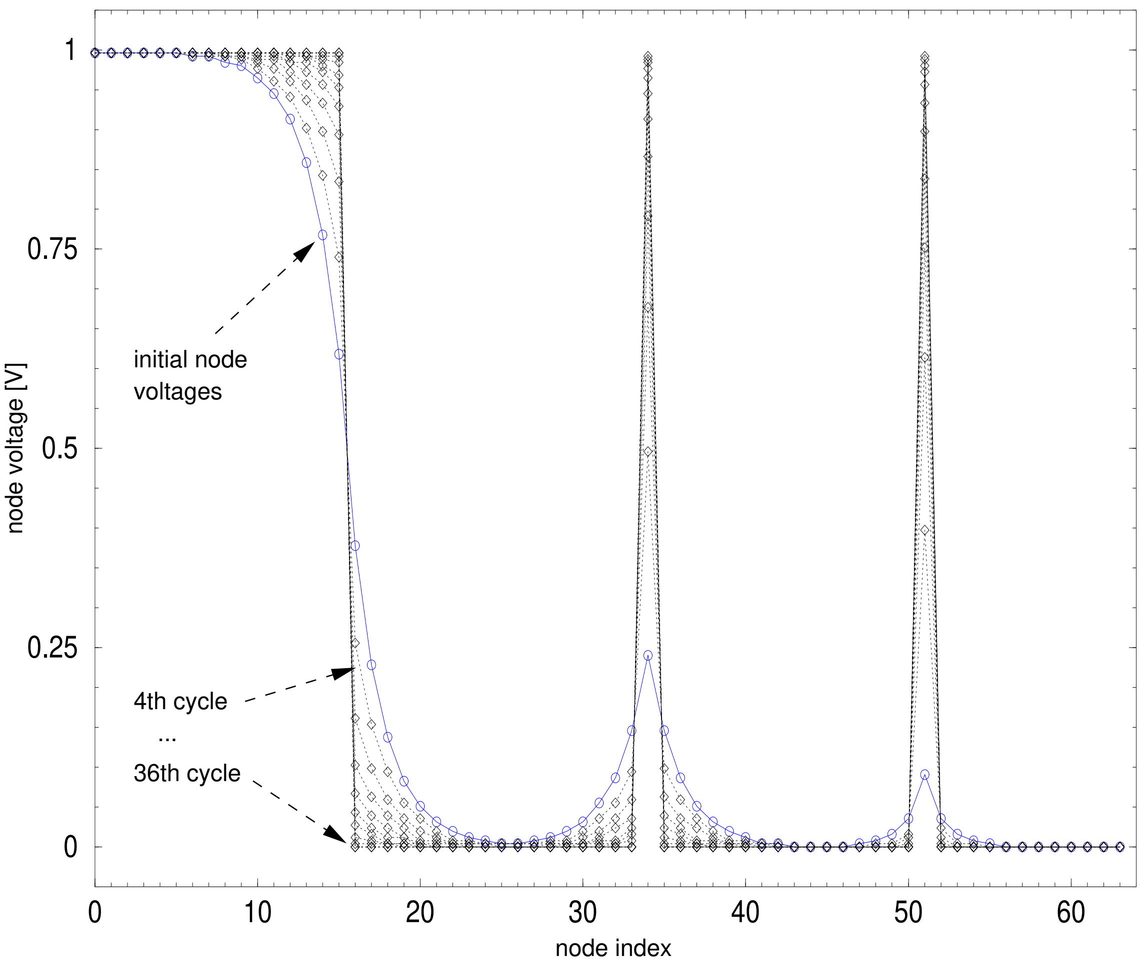



**Figure 1.17:** The input data of figure 1.9 again, this time processed by the SC-fuse network. At the beginning all fuses were blown. After $V_{\mathrm{off}}$ was decreased to 0.01 V, 36 network cycles have been executed. For every forth cycle the node voltages are drawn above.

changing the input voltages, $V_{\mathrm{off}}$ was decreased by a step large enough to make all fuses switch from resistive to off. $C_{\mathrm{n}}$ is $2/5$ of $C_{\mathrm{a}}$, which is a realistic value for a VLSI-implementation. The characteristic length $l$ is still 2 and according to eq.1.33 $\tau_{\mathrm{SC}} = 8$. The solid line depicts the initial state of the network with all the fuses in their resistive domain. Snapshots of the node voltages are plotted every forth network cycle after the fuses have been switched off (dashed lines). It can be seen that at the 8th cycle (second dashed line) the node voltages have reached $1 - e^{-1} \simeq 63\%$ of their final voltage levels in accordance with eq.1.32.

## 1.4 Simulation of the SC Resistive Fuse Network

A simulation algorithm has been developed to verify the behavior of the switched capacitor fuse. It can be also used to produce a numerical solution for the two-dimensional square resistive fuse network and can be adapted to other network topologies as well. Its base concept is the use of the charge conservation technique for the capacitors connected by closed switches. If $k$ capacitors are linked in parallel the resulting charge $Q'_i$ for each capacitor $C_i$ is:

$$Q'_i = V'_k C_i = \frac{\sum_{i=1}^{k} Q_i}{\sum_{i=1}^{k} C_i} C_i \tag{1.34}$$

with $Q_i$ representing the charge on $C_i$ before closing the switches. For the different phases of the network cycle the capacitors $C_{1\ldots k}$ must be appropriately chosen. In $\phi_2$ the sense amplifier output determines if the fuse capacitor $C_\mathrm{a}$ or $C_\mathrm{b}$ gets connected to a certain node. In fact the sense amplifier action is simulated by selecting either $C_\mathrm{a}$ or $C_\mathrm{b}$ depending on the voltage difference between SA and $\overline{\mathrm{SA}}$. If a pixel or precharge voltage source are to be connected to a capacitor, it will be charged to the according voltage.

### Simulation Algorithm

The simulation algorithm can be implemented in any procedural programming language like 'C' or 'Pascal'. In this paragraph it is formulated in a pseudo code using only assignments, denoted by ':=' and general program flow statements like 'for…to' and 'if…then' that can be found in many programming languages. Comments are preceded by '//'. Array elements are identified by indices printed in italics.

The following two-dimensional arrays are used to store the actual charge on the different capacitors:

$$Q_{\mathrm{p}xy}, Q_{\mathrm{n}xy}, Q_{\mathrm{sa4}xy}, Q_{\mathrm{av}xy}, Q_{\mathrm{bv}xy}, Q_{\mathrm{ah}xy}, Q_{\mathrm{bh}xy}\colon x \in [1 \ldots X_\mathrm{array}],\; y \in [1 \ldots Y_\mathrm{array}]$$

$X_\mathrm{array}$ and $Y_\mathrm{array}$ are the number of pixels in the x- and y-direction. The indices $x$ and $y$ denote one elementary cell consisting out of a source pixel ($Q_{\mathrm{p}xy}$), one node capacitor ($Q_{\mathrm{n}xy}$) and two fuses connecting the node to the $x+1$ ($Q_{\mathrm{ah}xy}$, $Q_{\mathrm{bh}xy}$) and $y+1$ ($Q_{\mathrm{av}xy}$, $Q_{\mathrm{bv}xy}$) cells. The sense amplifier capacitances $Q_{\mathrm{sa}i}$ of the four neighboring fuses are combined to one capacitance $Q_{\mathrm{sa4}xy}$ per node. This can be done because every sense amplifier is precharged to the same voltage before it is connected to the node in $\phi_1$. They are connected in parallel regarding the node.

Other Variables:

$s \in [-1, 1]$ sign of $V_\mathrm{cmp}$, phase 1: 1, phase 2:-1
$v$ actual node voltage
$q$ temporary charge value
$O_{ixy} : i \in$ [t,r,b,l], $x \in [1 \ldots X_\mathrm{array}]$, $y \in [1 \ldots Y_\mathrm{array}]$
output of comparators (t=top, r=right, b=bottom, l=left)

Constants:

| | |
|---|---|
| $C_\mathrm{a}$ | capacitance of one fuse capacitor |
| $C_\mathrm{p}$ | pixel capacitor |
| $C_\mathrm{n}$ | node capacitor |
| $C_\mathrm{sa}$ | total capacitance of a sense amplifier node SA or $\overline{\mathrm{SA}}$ |
| $C_\mathrm{sa4}$ | $:= 4C_\mathrm{sa}$ |
| $C_\mathrm{nodesum}$ | $:= C_\mathrm{sa4} + C_\mathrm{n}$ |
| $C_\mathrm{netsum}$ | $:= 4C_\mathrm{a} + C_\mathrm{p} + C_\mathrm{n}$ |
| $Q_\mathrm{off}$ | $:= 2\lambda_\mathrm{off} C_\mathrm{sa4} V_\mathrm{off}$ change in $Q_\mathrm{sa4}$ caused by $V_\mathrm{off}$ (see eq.1.17) |

$V_{\mathrm{in}xy}:\ x \in [1 \ldots X_\mathrm{array}],\ y \in [1 \ldots Y_\mathrm{array}]$ source voltages

The algorithm consists of the simulation loop that executes one network cycle and a subroutine called 'Close()' to calculate equation 1.29 and update the fuse and node capacitors.

Simulation loop:

**for** $s := 1$ **to** -1 **step** -2 **begin** *//loop over both phases*
    *//$\phi_1$ (includes $\phi_3$)*
    **for** $y := 1$ **to** $X_\mathrm{array}$ **begin**
        **for** $x := 1$ **to** $X_\mathrm{array}$ **begin**
            $Q_{\mathrm{sa4}xy} := C_\mathrm{sa4} V_\mathrm{precharge}$ *//precharge sense amplifiers ($\phi_3$)*
            $O_{ixy} := 0 \quad \forall i$ *//clear all sense amplifier outputs ($\phi_3$)*
            $Q_{\mathrm{p}xy} := C_\mathrm{p} V_{\mathrm{in}xy}$ *//close pixel switch*
            $v := (Q_{\mathrm{sa4}xy} + Q_{\mathrm{n}xy})/C_\mathrm{nodesum}$ *//new node voltage*
            $Q_{\mathrm{sa4}xy} := vC_\mathrm{sa4}$
            $Q_{\mathrm{n}xy} := vC_\mathrm{n}$
        **end** *x*
    **end** *y*
    *//$\phi_2$*
    **for** $y := 1$ **to** $X_\mathrm{array} - 1$ **begin**
        **for** $x := 1$ **to** $X_\mathrm{array} - 1$ **begin**
            $q := Q_{\mathrm{sa4}xy} + sQ_\mathrm{off}$
            *//determine comparator outputs*
            **if** $q < Q_{\mathrm{sa4}x,y+1}$ **then** $O_{\mathrm{t}xy} := 1$
                        **else** $O_{\mathrm{b}x,y+1} := 1$
            **if** $q < Q_{\mathrm{sa4}x+1,y}$ **then** $O_{\mathrm{r}xy} := 1$
                        **else** $O_{\mathrm{l}x+1,y} := 1$
            **call** Close()
        **end** *x*
        **call** Close()
    **end** *y*
    **for** $x := 1$ **to** $X_\mathrm{array}$ **call** Close()
**end** *s*

Close() subroutine:
*//calculate charge sum q*
$q := Q_{\mathrm{n}xy} + Q_{\mathrm{p}xy}$
$q := q + O_{\mathrm{t}xy} Q_{\mathrm{bv}xy} + (1 - O_{\mathrm{t}xy}) Q_{\mathrm{av}xy}$
$q := q + O_{\mathrm{r}xy} Q_{\mathrm{bh}xy} + (1 - O_{\mathrm{r}xy}) Q_{\mathrm{ah}xy}$
*//avoid left and bottom edge of network*
**if** $y > 1$ **then** $q := q + O_{\mathrm{b}xy} Q_{\mathrm{bv}x,y-1} + (1 - O_{\mathrm{b}x,y}) Q_{\mathrm{av}x,y-1}$
**if** $x > 1$ **then** $q := q + O_{\mathrm{l}xy} Q_{\mathrm{bh}x-1,y} + (1 - O_{\mathrm{l}x,y}) Q_{\mathrm{ah}x-1,y}$
*//compute new node voltage*
**if** $y = 1$ **and** $x = 1$ **then** $v := q/(C_{\mathrm{netsum}} - 2C_{\mathrm{a}})$
**else if** $y = 1$ **or** $x = 1$ **then** $v := q/(C_{\mathrm{netsum}} - C_{\mathrm{a}})$
  **else** $v := q/C_{\mathrm{netsum}}$
*//update fuse capacitors*
$q := vC_{\mathrm{a}}$
**if** $O_{\mathrm{t}xy}$ **then** $Q_{\mathrm{bv}xy} := q$ **else** $Q_{\mathrm{av}xy} := q$
**if** $O_{\mathrm{r}xy}$ **then** $Q_{\mathrm{bh}xy} := q$ **else** $Q_{\mathrm{ah}xy} := q$
**if** $y > 1$ **then**
  **if** $O_{\mathrm{b}xy}$ **then** $Q_{\mathrm{bv}x,y-1} := q$ **else** $Q_{\mathrm{av}x,y-1} := q$
**if** $x > 1$ **then**
  **if** $O_{\mathrm{l}xy}$ **then** $Q_{\mathrm{bh}x-1,y} := q$ **else** $Q_{\mathrm{ah}x-1,y} := q$
*//update node capacitor*
$Q_{\mathrm{n}xy} := vC_{\mathrm{n}}$

In the simulation loop's $\phi_1$ part the sense amplifier input voltages are calculated for each node. In $\phi_2$ they are compared to their top and right neighbors taking into account the $V_{\mathrm{off}}$ voltage and the phase. If the sense amplifier switches to the SA side the according $O_{\mathrm{i}xy}$ will be set to one. Otherwise $O_{\mathrm{i}x+1,y}$ or $O_{\mathrm{i}x,y+1}$ are used to store that the sense amplifier's $\overline{\mathrm{SA}}$ side is at $V_{\mathrm{DD}}$. By this way $O_{\mathrm{i}xy}$ contains the valid sense ampifier data for all neighbors of every pixel after the $x$- and $y$-loops have finished. No special care must be taken for the network edges where the pixels do not have all their four fuses. Only the Close() subroutine must take this into account. For the bottom row it must not add the charge from the bottom neighbors, for example. The denominator of eq.1.29 is also different for the edge pixel, making it necessary to reduce the capacitance by one or two $C_{\mathrm{a}}$ at $x = 1$ and/or $y = 1$.
The edge information can be extracted from the simulation by comparing the $O_{\mathrm{t}xy}$ and $O_{\mathrm{r}xy}$ values in both phases with table 1.2.

### Simulation of the SC-fuse

With this algorithm the data for the two-dimensional square network was calculated in the previous sections. Table 1.3 shows the values used for the different capacitances. Besides $C_{\mathrm{sa}}$ they are of the same order of magnitude as in the VLSI implementation. With $C_{\mathrm{sa}}$ *ideal* the effects of the sense amplifier capacitance, as calculated in eq.1.24, are supressed. If $\lambda_{\mathrm{off}}$ (see eq.1.17) is kept constant, any decrease in $C_{\mathrm{sa}}$ will only reduce $\lambda_{\mathrm{sa}}$. $C_{\mathrm{sa}}$ *real* is the value that $C_{\mathrm{sa}}$ will have if realized in a CMOS process, ie. $C_{\mathrm{sa}} \approx C_{\mathrm{p}}$. In section 4.3.3 the measurements of the EDDA chip are compared to the simulation.

| | $C_{\mathrm{a}}$ | $C_{\mathrm{p}}$ | $C_{\mathrm{n}}$ | $C_{\mathrm{sa}}$ *ideal* | $C_{\mathrm{sa}}$ *real* |
|---|---|---|---|---|---|
| capacitance [fF] | 250 | 62.5 | 100 | $10^{-5}$ | 50 |

**Table 1.3:** Capacitor values used to produce the results for the two-dimensional square network in this chapter.

### Numerical solutions for the continuous-time network

If $C_{\mathrm{sa}}$ is selected to be a few orders of magnitude smaller than $C_{\mathrm{p}}$, $C_{\mathrm{a}}$ and $C_{\mathrm{n}}$ the simulation can also be used to calculate a numerical solution of the continuous-time resistive fuse network. For a steady-state input the accuracy depends on the number of cycles performed by the simulation and the numerical precision of the binary representation of the variables and constants. If standard single precision floating point numbers[10] are used the simulation needs $16.6\tau_{\mathrm{SC}}$ cycles to exploit their precision[11]. This is due to the exponential convergence of the array node voltages (see section 1.3.3).

[10] They cover the range from $3.4 \cdot 10^{-38}$ to $3.4 \cdot 10^{38}$ using 24 bits for the mantissa and 8 bits for the exponent.

[11] $2^{-24} = e^{-\frac{n_{\mathrm{cycle}}}{\tau_{\mathrm{SC}}}} \Rightarrow n_{\mathrm{cycle}} = -\tau_{\mathrm{SC}} \log 2^{-24} \approx 16.6\tau_{\mathrm{SC}}$

# Chapter 2

# Implementation

*This chapter describes the VLSI implementation of the SC resistive fuse network. Starting with the single network cell - consisting of the pixel and fuse circuits - the different parts of the EDDA chip are explained in detail. Additional topics are the interplay of the individual circuits and the timing of the clock signals in the distinct operational modes. At the end of this chapter the important aspects of the chip's layout are shown. It closes with a die photograph.*

## 2.1 Chip Overview

The EDDA chip is based on the resistive fuse network as it has been presented in the previous chapter. The image is stored in the pixel circuits. The node voltages and the state of the fuses can be read out from the network. An integrated digital control logic generates the clock signals needed for the SC circuits. The EDDA chip was implemented in the AMS[1] 0.6 $\mu$m process [AMS98-1, AMS98-2]. It operates from a single supply voltage $V_{DD}$ of 5 volt. Figure 2.1 shows a block diagram of the whole chip. An array of 66 $\times$ 66 elementary cells, each consisting of one pixel and two fuses, form the analog core. They are based on the SC resistive fuse technique. Below the bottom row the clock drivers for the individual columns are located. On the right each array row is terminated by a readout sense amplifier, on the left by a storage amplifier. To select individual columns for I/O, a shift register runs in parallel to the clock driver row to enable or disable certain clock signals by column.

Readout data gets multiplexed on an eight bit bus running from top to bottom right from the readout amplifiers. The analog input data is stored in one sample-and-hold (S&H) stage per column. This is controlled by a shift register, depicted on the left side of the storage cells in figure 2.1. The storage amplifiers transfer the contents of the S&H capacitors into the array column-by-column.

[1]Austria Mikro Systeme International AG

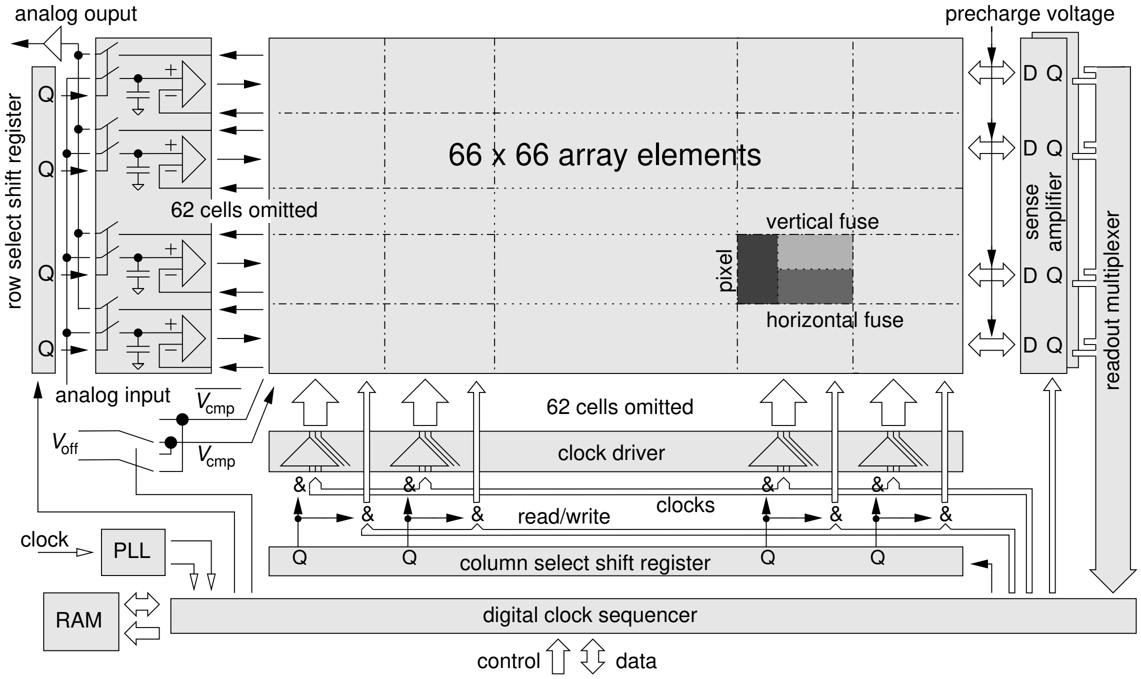



**Figure 2.1:** Block diagram of the EDDA chip.

The digital control section is drawn at the bottom of figure 2.1. It contains a sequencer to produce the different clock signals. Its program is stored in a static RAM, shown to the left of the control block. A bidirectional synchronous bus connects the digital part of EDDA to the outside world. The sequencer clock and the bus clock are derived from the external *clock* input by a phase-locked-loop (PLL). The clock multiplier for the internal sequencer clock can be 1, 2, 3 or 4.

At the lower left corner of the array the switches for the $V_{\mathrm{off}}$ voltage are located. They change the polarity of the $V_{\mathrm{cmp}}$ signals according to the phase of the clock cycle (see figure 1.14). The node voltages can be read out through an analog output, buffered by an integrated amplifier (upper left). The desired cell is selected by the row and column shift registers.

## 2.2 The Elementary Network Cell

The operation principle of the SC fuse and pixel circuits have been explained in chapter 1.3.1. This section goes into the details of their functionality and timing. The elementary cell of the network consists of one pixel and two fuses. The cell has an analog input to store the pixel source information as well as an analog output to read out the node voltage. Every signal that enters or leaves the array cell can be connected to a shared line, except the network connections to the neighbors. By arranging the cells in rows and columns a resistive fuse network is formed. In figure 2.2 these main parts of an array element are shown. The network node line is drawn as a thick cross centered between the blocks connecting to it. The fuses separate the node from the neighboring ones in the top and right direction. The node capacitor $C_{\mathrm{n}}$, like every capacitor in the array, is implemented by a transistor with a common drain and source connection. This avoids

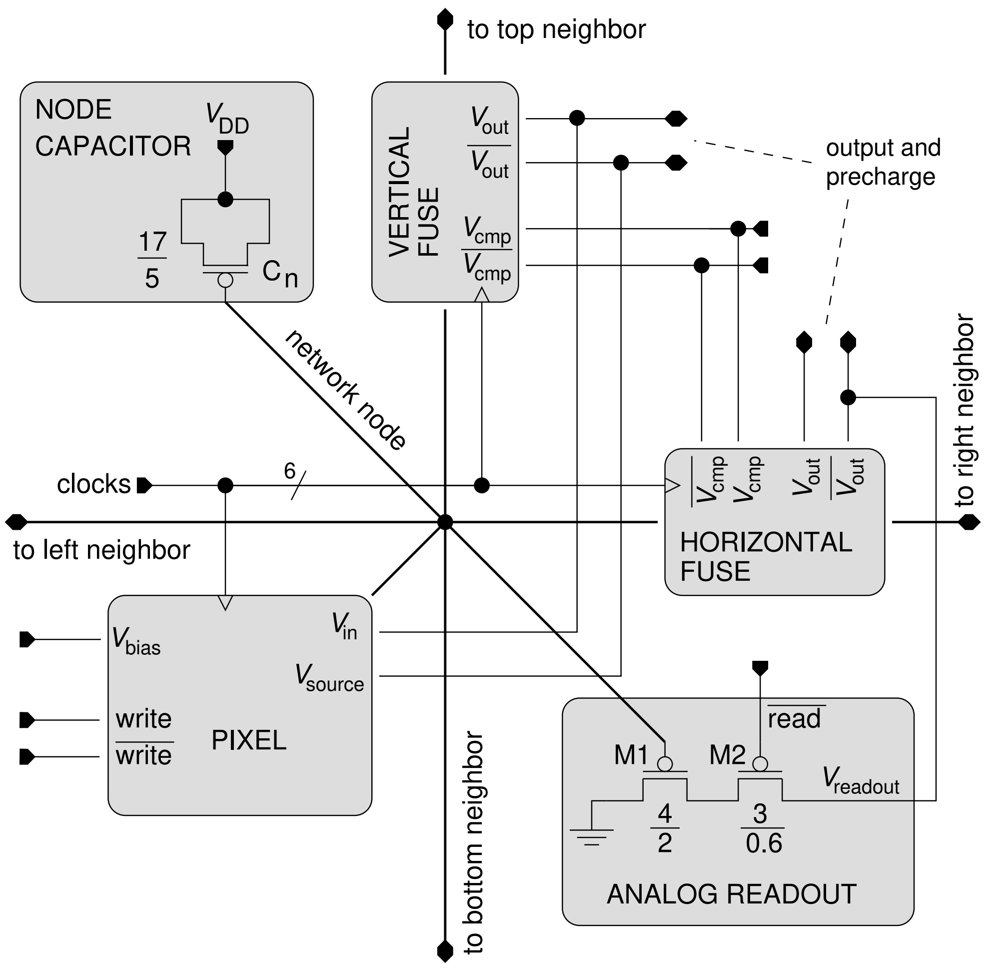



**Figure 2.2:** Schematic diagram of an elementary cell of the network. The pixel and fuse circuitry are shown as blocks only. Their schematics are depicted in the following figures. The numbers next to the capacitor $C_n$ and the transistors M1 and M2 are the drawn width and length (W/L) of the gate in $\mu$m.

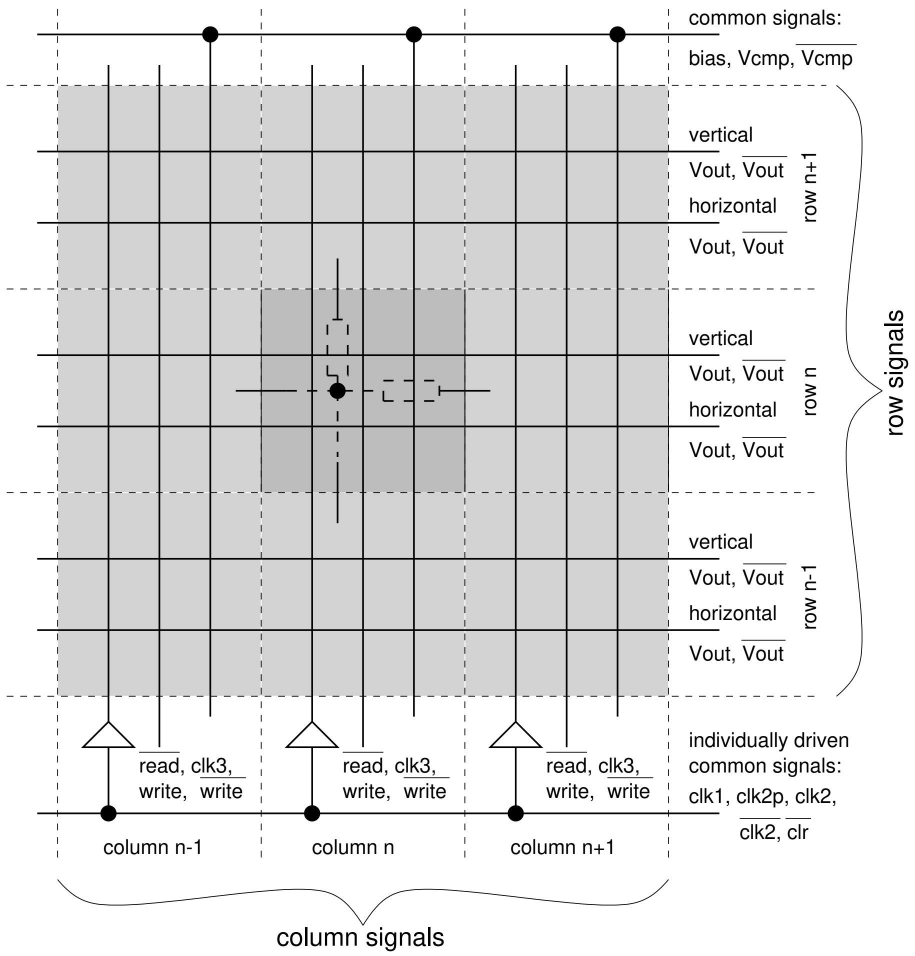



**Figure 2.3:** Cutout of the array to illustrate the different kinds of global signals. Clk1, clk2p, clk2, $\overline{\text{clk2}}$, clk3 and $\overline{\text{clr}}$ are abbreviated as *'clocks'* in the previous figure.

the need for an extra process step to generate dual-poly capacitors. The capacitance per area is even higher for a MOS-capacitor, because the gate-oxide is the thinnest oxide in the process, thus possessing the largest $C_{\text{Ox}}$[2]. Considering the C-V characteristics of such a capacitor the linear range with maximum capacitance lies in-between $V_{\text{T}}$ and $V_{\text{DD}}$ [GEI90-5]. By using either NMOS or PMOS field effect transistors (FET) voltages up to $V_{\text{DD}}$ or down to ground respectively can be used. The exact width to length ratio (W/L) is determined by layout considerations (see 2.6). The total capacity is calculated by adding up the channel, gate-bulk and gate-drain/source capacities. In case of $C_{\text{n}}$ the channel counts for 95% of the total capacity. PMOS capacitors are better suited for sensitive signals since they are isolated from the substrate noise by an n-well.

Figure 2.3 illustrates the connection scheme of the network. The bias, $V_{\text{cmp}}$ and $\overline{V_{\text{cmp}}}$ signals are common to all array elements. The clock signals clk1, clk2p, clk2, $\overline{\text{clk2}}$ and $\overline{\text{clr}}$ are also the same within the network, but individually buffered in each column. The horizontal and vertical

[2]In the AMS 0.6 $\mu$m process $C_{\text{Ox}}$ =2.76 fF/$\mu$m$^2$ compared to 0.86 fF/$\mu$m$^2$ for the poly1/poly2 capacitor. This is roughly 1/3 of the MOS-capacitor.

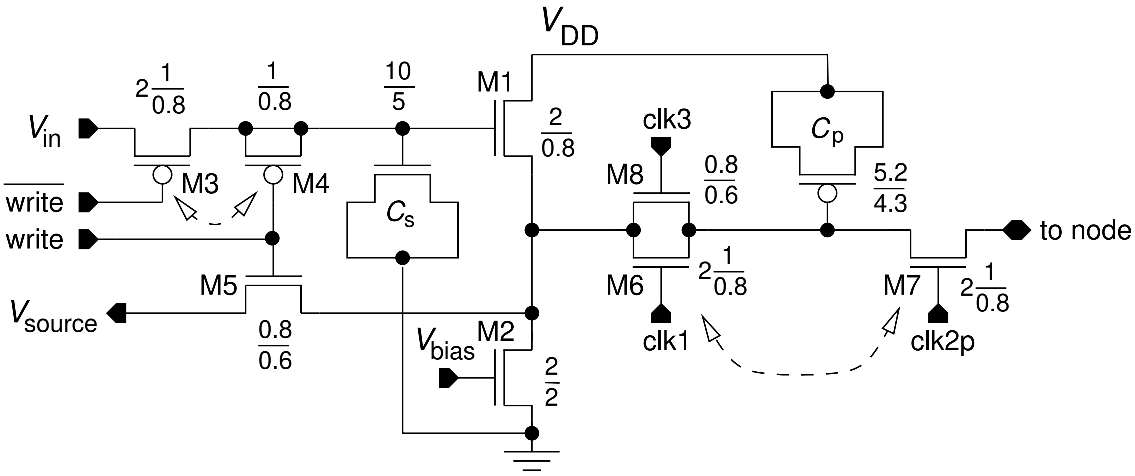



**Figure 2.4:** Schematic diagram of the pixel. Transistors connected by a dashed arrow should match. If a number is given in front of the W/L ratio, the transistor consists of multiple gates.

$V_{out}$ and $\overline{V_{out}}$ lines are shared between one row of cells. Clk3, $\overline{\text{read}}$, write and $\overline{\text{write}}$ connect one array column.

In the lower right of figure 2.2 the readout circuit is shown consisting of transistors M1 and M2. M1 is connected as a source follower [GEI90-6]. There is only one load resistor shared among all array pixels. It is located in the output amplifier and the desired pixel is connected to it by the combination of M2 - thereby selecting the column - and a transistor choosing the appropriate array row. The readout signal uses the horizontal $\overline{V_{out}}$ line. The readout of the node voltage is non-destructive and can be performed at any time when the $\overline{V_{out}}$ line is not driven by the fuse or the precharge circuits.

### 2.2.1 Pixel

The schematic of a pixel is depicted in figure 2.4. $C_s$ together with M1 and M2 form the voltage source. The source follower M1 provides the necessary current to charge the pixel capacitor $C_p$. The speed and the quiescent current of this stage are controlled by the gate voltage of M2, connected to the global bias line. The voltage drop of the M1/M2 stage sets the upper limit for the node voltage. At a bias voltage of 1 V it is about 3.2 V. The quiescent current of the pixel source follower is the only static current flowing in the network. Thus by setting the bias voltage to zero static power consumption is reduced to the leakage currents. $C_s$ must be an NMOS transistor because the input voltage of M1 can reach $V_{DD} - V_{T\,PMOS} \approx$4.2 V.

To store a voltage on $C_s$ the $\overline{\text{write}}$ signal must switch on M3. It is a PMOS transistor to keep its $|V_{GS}|$ voltage sufficiently high across the total input range of M1/M2. The linearity of the source follower is degraded by the body-effect and the slope of the I-V characteristic of M2. Also there is a mismatch between M1 and M2 of different pixels. The minimum size transistor in the AMS 0.6 $\mu$m process has a W/L of 0.8/0.6. The ratios chosen for M1 and M2 are larger than that to improve matching and linearity. This is far from enough to suppress the fixed pattern noise completely but area is a major constraint. To enhance the precision of the voltage source without increasing area usage the pixel source follower is part of a feedback loop. Operational amplifiers at the end of each row compare the output signal $V_{\text{source}}$ of M1 with the desired source voltage and change $V_{\text{in}}$ accordingly. M5 enables this feedback. Its exact W/L is not critical and therefore minimum size. Only a low channel resistance is needed because it has to drive the capacitance of the line back to the feedback amplifier. Due to the dynamic range of $V_{\text{source}}$ from 0.4 to 3.2 V this makes an NMOS device necessary. The $\overline{\text{write}}$ signal therefore needs an active-high counterpart: *write*. $V_{\text{in}}$ and $V_{\text{source}}$ use the vertical $V_{\text{out}}/\overline{V_{\text{out}}}$ pair. All pixels of one column of the array can be written simultaneously.

The critical moment in the operation of the feedback loop is when the write signal becomes inactive. The gate-source capacitance of M3 injects charge into the storage node. The amount of charge transferred depends on the gate voltage swing which is 5 V for the digital control signals like write and $\overline{\text{write}}$. Charge conservation techniques give the following voltage change for the storage node:

$$\Delta V_{\text{s}} = \frac{C_{\text{GS M3}}}{C_{\text{s}} + C_{\text{GS M3}}} V_{\text{inject M3}} \tag{2.1}$$

Using minimum size W/L values for M3 and assuming a constant $C_{\text{GS}} = C_{\text{GS max}}$[3]:

$$\Delta V_{\text{s}} \approx \frac{0.95 \text{ fF}}{146 \text{ fF} + 0.95 \text{ fF}} \, 5 \text{ V} = 32.4 \pm 12 \text{ mV} \tag{2.2}$$

The error is the uncertainty of the drawn width/length of the transistor as given by the manufacturer [AMS98-3]. This is a worst case error, usually only encountered by devices on different wafers. The effective error will be much smaller if the devices are in proximity to each other. This has been achieved by using the dummy transistor[4] M4 to compensate for the charge injection of M3. The inverted $\overline{\text{write}}$ signal, write, has been already necessary for M3. It is no longer the matching between transistors in different array elements but the matching of M3 to M4 that dominates the charge injection variation between different pixels. To improve this matching M3 is split into two transistors making the resulting layout of M3 and M4 more insensitive to process gradients (see 2.6).

M6, M7 and $C_{\text{p}}$ form the SC pixel resistor. Clk1 is active in $\phi_1$, clk2p in $\phi_2$ (the exact relationship is explained in 2.2.3). M6 and M7 are split into two gates for better matching. This shall avoid node voltage errors due to charge injection. $\Delta V_{\text{p clk1}}$ is compensated by $\Delta V_{\text{p clk2p}}$[5]. M8 is used to decrease the power consumption of the circuit. It is enabled by clk3 (active in $\phi_3$). At this moment clk2p is inactive and clk1 not yet high. Thus the time interval for charging $C_{\text{p}}$ is $\phi_3 + \phi_1$ instead of only $\phi_1$ and the output impedance of the source follower can be increased by a factor of 2 (assuming clock phases of equal length). As can be seen from the equation of a MOS source follower's small signal output resistance [GEI90-6],

$$r_{\text{out}} \approx \frac{1}{g_{\text{M1}}} = \sqrt{\frac{L_{\text{M1}}}{2 K_{\text{N}} W_{\text{M1}} I_{\text{D M1}}}}, \tag{2.3}$$

reducing $r_{\text{out}}$ by a factor of two leads to a current of $I_{\text{D M1}}/4$.

[3] The voltage independent terms of $C_{\text{GS}}$, like the gate-source overlap capacitance, are getting larger compared to the voltage dependent channel capacitance if minimum size geometries are used, making the assumption justified.

[4] A dummy transistor is a transistor whose drain and source are connected to the node that should be compensated and whose gate is driven by the inverted clock. Its charge injection cancels that of the switch transistor if it has half the width. The advantage compared to a transmission gate is the area saving since there is no need for complementary devices.

[5] In fact M7 also adds $\Delta V_{\text{n}}$ to the node capacitor $C_{\text{n}}$, but this gets cancelled when clk2p changes to low again.

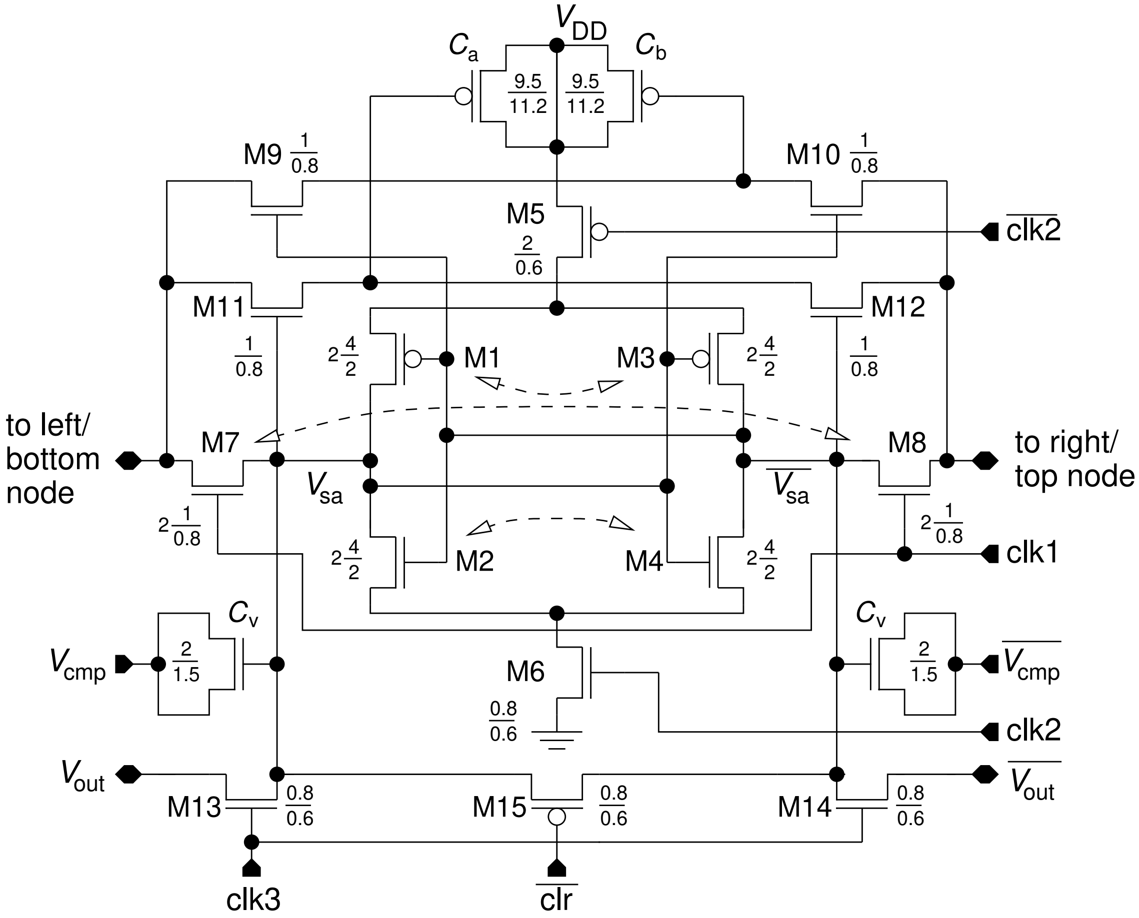



**Figure 2.5:** Schematic diagram of the fuse.

### 2.2.2 Fuse

The fuse circuit is shown in figure 2.5. M1 to M6 form a symmetric sense amplifier. It consists of two cross coupled inverters M1/M2 and M3/M4. They can be connected to $V_{\mathrm{DD}}$ and ground by M5 and M6. If clk2 and $\overline{\mathrm{clk2}}$ are inactive, the gates of M1 to M4 float. Charge can be transmitted onto them by either M7/M8 or M13/M14. If power is applied via M5/M6 the sense amplifier switches to one of its two stable states: $V_{\mathrm{SA}} = V_{\mathrm{DD}}$ and $\overline{V_{\mathrm{SA}}} =$ ground (state SA) or $V_{\mathrm{SA}} =$ ground and $\overline{V_{\mathrm{SA}}} = V_{\mathrm{DD}}$ (state $\overline{\mathrm{SA}}$). The state entered depends on the gate voltages of M1/M2 and M3/M4. The matching between M1/M3 and M2/M4 determines the offset of the sense amplifier. To keep it low, M1 to M4 are larger than the other transistors (W/L = 8/2) and have split gates for a better matching layout. The input switches M11 and M12 use the same technique to avoid differences in their charge injection resulting in a differential voltage error on the sense amplifier inputs and thereby a $V_{\mathrm{off}}$ offset. The $\Delta V_{\mathrm{n}}$ caused by them cancels at the end of clk1. The $V_{\mathrm{cmp}}/\overline{V_{\mathrm{cmp}}}$ coupling capacitors $C_{\mathrm{v}}$ are NMOS transistors. The fuse resistor consists of $C_{\mathrm{a}}$ and $C_{\mathrm{b}}$ together with the NMOS transistors M9 to M12.

The sense amplifier is cleared by precharging it with an external voltage. $V_{\mathrm{out}}$ and $\overline{V_{\mathrm{out}}}$ are connected to the precharge voltage source (see section 2.3.2). Clk3 enables charge transfer across

M13 and M14 until $V_{\mathrm{SA}} = \overline{V_{\mathrm{SA}}} = V_{\mathrm{precharge}}$. To keep the voltage error between $V_{\mathrm{SA}}$ and $\overline{V_{\mathrm{SA}}}$ small a PMOS transistor M15 has been added to short-circuit both sense amplifier halves. This mechanism also reduces the current across the $V_{\mathrm{out}}/\overline{V_{\mathrm{out}}}$ lines[6] and makes it independent from the former fuse state. The $\overline{\mathrm{clr}}$ clock signal enables M15. If the network is operated as described in chapter 1 $\overline{\mathrm{clr}}$ will be equal to the inverted clk3 signal ($\overline{\mathrm{clk3}}$). The usage of a PMOS transistor cancels the charge injection of the NMOS switches M13 and M14 reducing $V_{\mathrm{SA}} - \overline{V_{\mathrm{SA}}}$ to a minimum.

The pixel and fuse circuits are optimized for low area usage and speed. All network switches are NMOS transistors instead of transmission gates. This makes it necessary to use layout techniques suppressing charge injection differences between transistors that should match. The node voltage range is also limited to about $V_{\mathrm{DD}} - 1.5$ V to keep $V_{\mathrm{GS}}$ sufficiently high and thereby the channel resistance low. For the network capacitors this is an advantage because the they must be PMOS transistors to cope with a voltage range extending closer to ground than $V_{\mathrm{DD}}$. Thus they can be kept in a separate n-well with very low impedance $V_{\mathrm{DD}}$ connections isolating them from the switching noise injected into the substrate by the total of the clocked components of the network.

The lower limit of the dynamic range has its roots in the fuse circuit. The $C_{\mathrm{v}}$ capacitors are of the NMOS type. This is due to the fact that for narrow transistors the matching of NMOS-FETs is better than that of PMOS (in [AMS94] a width error larger by a factor of 2.6 for PMOS than for NMOS is reported. Similar values can be found in [DRO99]). The mismatch of $C_{\mathrm{v}}$ introduces a scaling factor to $V_{\mathrm{off}}$ that differs from fuse to fuse. Another advantage is that $V_{\mathrm{off}}$ can be referenced to ground. The C-V characteristic of $C_{\mathrm{v}}$ sets the lower limit of the sense amplifier voltage to $V_{\mathrm{off}} + V_{\mathrm{T}}$. Below that $\lambda_{\mathrm{off}}$ (see 1.17) depends on the sense amplifier voltage since $C_{\mathrm{v}}$ is no longer voltage independent. The charge sharing between the node capacitor and the sense amplifier reduces the effective voltage swing on the gates of M1 to M4. In addition it is shifted by the precharge voltage. According to section 1.3.1 the sense amplifier voltage $V_{\mathrm{SA}}$ is equal to ($C_{\mathrm{SA}}$ is the total sense amplifier capacitance):

$$\begin{aligned} V_{\mathrm{SA}} &= \frac{Q_{\mathrm{n}} + Q_{\mathrm{SA}}}{C_{\mathrm{n}} + C_{\mathrm{SA}}} \\ &= \frac{V_{\mathrm{n}} C_{\mathrm{n}} + V_{\mathrm{precharge}} C_{\mathrm{SA}}}{C_{\mathrm{n}} + C_{\mathrm{SA}}} \\ &= \lambda_{\mathrm{n}} V_{\mathrm{n}} + (1 - \lambda_{\mathrm{n}}) V_{\mathrm{precharge}} \end{aligned} \tag{2.4}$$

with $\lambda_{\mathrm{n}}$ being the ratio between $V_{\mathrm{SA}}$ and $V_{\mathrm{n}}$:

$$\frac{V_{\mathrm{SA}}}{V_{\mathrm{n}}} = \frac{C_{\mathrm{n}}}{C_{\mathrm{n}} + C_{\mathrm{SA}}} = \lambda_{\mathrm{n}} \approx 0.80 \tag{2.5}$$

Table 2.1 lists the numeric capacitance values of the different capacitors of the network circuit. A new one is introduced: $C_{\mathrm{w}}$ is the total capacitance of the $V_{\mathrm{SA}}$ wire to ground. The small values of $C_{\mathrm{v}}$ and $C_{\mathrm{g}}$ make it necessary to include the wiring capacitance in the total sense amplifier capacitance: $C_{\mathrm{SA}} = C_{\mathrm{v}} + C_{\mathrm{g}} + C_{\mathrm{w}}$. The numerical value of $\lambda_{\mathrm{n}}$ was calculated with these numbers. $V_{\mathrm{SA}}$ has also an upper limit with respect to $V_{\mathrm{n}}$:

$$V_{\mathrm{SA}} \leq V_{\mathrm{n\,min}} + V_{\mathrm{T}}, \quad V_{\mathrm{T}} \approx 0.78\mathrm{V} \tag{2.6}$$

[6]Since 66 fuses share the $V_{\mathrm{out}}/\overline{V_{\mathrm{out}}}$ lines the current can become quite large. To make things worse it is also dependent of the edge pattern. Every fuse in the row can be either at state SA or $\overline{\mathrm{SA}}$ when clk3 is enabled. By connecting them to a common wire a cross-coupling between different fuses may occur if the impedance of the precharge source is not low enough and M15 is omitted.

| | $C_a$ | $C_p$ | $C_n$ | $C_g$ | $C_v$ | $C_w$ |
|---|---|---|---|---|---|---|
| [fF] | 305 | 67 | 246 | 30 | 10 | 21 |

| $\lambda_{off}$ | $\lambda_{sa}$ | $\lambda_n$ | $l$ |
|---|---|---|---|
| 0.16 | 0.48 | 0.8 | 1.55 |

**Table 2.1:** Capacitor values and derived parameters of the VLSI implementation of the two-dimensional square network. The typical process parameter values of channel, gate-source/drain overlap and gate-bulk overlap capacitance have been used to calculate each table entry. $C_w$ is the wire (parasitic) capacitance of the fuse. $\lambda_{off}$ and $\lambda_{sa}$ have been defined in equation 1.17 or 1.24 respectively, $l$ is the characteristic length of the network.

Since $V_{GS\,M9} = V_{SA} - V_n$ the fuse resistor switches M9 to M12 will start to conduct if eq. 2.6 is violated[7]. This limit is also valid for $V_{precharge}$ because eq.2.6 must be observed not only in $\phi_1$ but also in $\phi_3$ where $V_{SA} = V_{precharge}$. $V_{n\,min}$ is the smaller one of the voltages from both nodes connected to the fuse. If the fuse is blown and $V_{n\,min}$ is the voltage of the node connected to the $\overline{V_{SA}}$ side of the sense amplifier there is no correlation between $V_n$ and $V_{n\,min}$. Therefore $V_{n\,min}$ must be equal to the minimum node voltage of any node, i.e. the lower node voltage limit of the network. Expressing $V_{SA}$ in eq.2.6 by eq.2.4 gives a second inequation for $V_{precharge}$:

$$\lambda_n V_{n\,min} + (1-\lambda_n)V_{precharge} \leq V_{n\,min} + V_T \quad (2.7)$$

$$V_{precharge} \leq V_{n\,min} + \frac{V_T}{1-\lambda_n} \quad (2.8)$$

Since $1-\lambda_n \leq 1$ and $V_T > 0$:

$$V_{precharge} \leq V_{n\,min} + V_T \leq V_{n\,min} + \frac{V_T}{1-\lambda_n} \quad (2.9)$$

The maximum $V_{precharge}$ is therefore:

$$V_{precharge\,max} = V_{n\,min} + V_T \quad (2.10)$$

Equation 1.24 expresses the relationship between $V_{n\,min}$ and $V_{p\,min}$:

$$V_{n\,min} = \lambda_{sa} V_{precharge\,max} + (1-\lambda_{sa})V_{p\,min} \quad (2.11)$$

By substituting eq.2.10 for $V_{n\,min}$ the maximum precharge voltage can be written in terms of the minimum pixel voltage $V_{p\,min}$ and constants:

$$V_{precharge\,max} - V_T = \lambda_{sa} V_{precharge\,max} + (1-\lambda_{sa})V_{p\,min} \quad (2.12)$$

$$V_{precharge\,max} = V_{p\,min} + \frac{V_T}{1-\lambda_{sa}} \approx V_{p\,min} + 1.5\mathrm{V} \quad (2.13)$$

[7]The sub-threshold current flowing before $V_{SA} - V_n$ reaches $V_T$ is too small to change the charge on $C_a$ or $C_b$ significantly on the time-scale of the network. With $V_{GS} = 500$ mV the $RC$ time constant of $C_a$ is more than 1 ms compared to operating frequencies greater than 10 MHz.

For the right part the numbers from table 2.1 have been used. The minimum sense amplifier voltage $V_{\text{SA min}}$ for a given $V_{\text{precharge}} = V_{\text{precharge max}}$ and expressed in terms of $V_{\text{p min}}$ is (following from eq.2.4 and eq.1.24):

$$\begin{aligned}
V_{\text{SA min}} &= \lambda_{\text{n}} V_{\text{n min}} + (1-\lambda_{\text{n}}) V_{\text{precharge max}} \\
&= \lambda_{\text{n}} (\lambda_{\text{sa}} V_{\text{precharge max}} + (1-\lambda_{\text{sa}}) V_{\text{p min}}) \\
&\quad + (1-\lambda_{\text{n}}) V_{\text{precharge max}} \\
&= \lambda_{\text{n}} \left( \lambda_{\text{sa}} \left( V_{\text{p min}} + \frac{V_{\text{T}}}{1-\lambda_{\text{sa}}} \right) + (1-\lambda_{\text{sa}}) V_{\text{p min}} \right) \\
&\quad + (1-\lambda_{\text{n}}) \left( V_{\text{p min}} + \frac{V_{\text{T}}}{1-\lambda_{\text{sa}}} \right) \\
&= V_{\text{p min}} + V_{\text{T}} \frac{1-\lambda_{\text{n}} + \lambda_{\text{n}}\lambda_{\text{sa}}}{1-\lambda_{\text{sa}}} \approx V_{\text{p min}} + 0.88\text{V} \qquad (2.14)
\end{aligned}$$

Using this result the maximum $V_{\text{off}}$ value can be calculated as it depends on the minimum pixel voltage ($V_{\text{off max}} + V_{\text{T}} = V_{\text{SA min}}$):

$$V_{\text{off max}} = V_{\text{p min}} + V_{\text{T}} \frac{\lambda_{\text{n}}\lambda_{\text{sa}} - \lambda_{\text{n}} + \lambda_{\text{sa}}}{1-\lambda_{\text{sa}}} \approx V_{\text{p min}} - 9.6\text{mV} \qquad (2.15)$$

If $V_{\text{off}}$ is kept smaller than $V_{\text{off max}}$ $\lambda_{\text{off}}$ remains constant. The formerly stated $V_{\text{p min}}$ of 0.8 V means an effective voltage swing of

$$\Delta V_{\text{SA}} = \lambda_{\text{off}} V_{\text{off}} = \lambda_{\text{off}} (V_{\text{p min}} - 9.6\text{mV}) = 0.13\text{V} \qquad (2.16)$$

at the gates of M1/M2 and M3/M4. If $V_{\text{precharge}}$ is selected smaller than $V_{\text{precharge max}}$ the $V_{\text{off}}$ range decreases. Using the numbers of table 2.1:

$$V_{\text{off max}} = 0.42 V_{\text{p min}} + 0.58 V_{\text{precharge}} - V_{\text{T}} \qquad (2.17)$$

To get an impression of the $V_{\text{off}}$ values necessary for edge detection $V_{\text{off max}}$ can be compared to $V_{\text{off}}$ as used in figure 1.12. In the edge image $V_{\text{off}}$ has been 1/60 of the total dynamic range, $\lambda_{\text{sa}} = 0$ and $\lambda_{\text{off}} = 1$. Starting off with an input voltage range of 0.8 to 3.2 V $V_{\text{off}}$ for the VLSI implementation can be calculated:

$$V_{\text{off}} = \frac{1}{60} \frac{(1-\lambda_{\text{sa}})(3.2-0.8)\text{V}}{\lambda_{\text{off}}} \approx 0.13\text{V} \qquad (2.18)$$

Comparing this to the $V_{\text{off max}}$ result from eq.2.15, 0.79 V, there is enough safety margin to account for the threshold voltage variations of the manufacturing process. Considering $V_{\text{T}}$ variations as reported in [AMS94] or [LOV98], the mismatch between adjacent array elements is expected to lie in the range of 15 to 20 mV. This changes $V_{\text{precharge max}}$ for a maximum of 40 mV. Worst case process parameter variations are about 100 mV. By adjusting $V_{\text{precharge max}}$ to take the worst case as well as the mismatch into account, $V_{\text{off max}}$ is still 0.7 V.

### 2.2.3 Timing

Figure 2.6 illustrates the timing of the network clock signals. In section 1.3.1 the clock signals have been named $\phi_1$, $\phi_2$ and $\phi_3$. In the VLSI circuit there is only a direct correspondence between

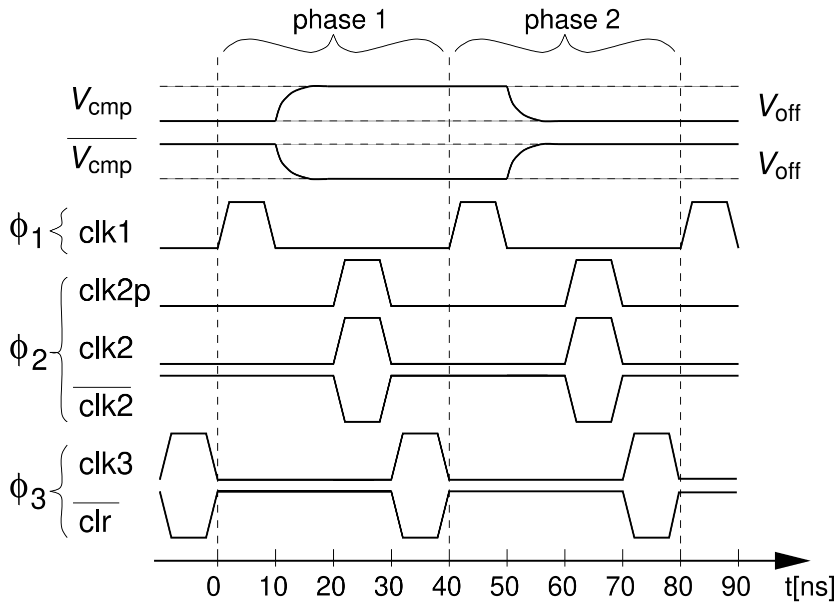



**Figure 2.6:** Time relationship of the six clock signals. A standard network cycle is shown.

clk1 and $\phi_1$. The functionality of $\phi_2$ and $\phi_3$ is taken over by multiple clock signals. Clk2 and clk2p are active-high (they control NMOS switches) while $\overline{\text{clk2}}$ is active-low (M5 is a PMOS-FET). Clk2p controls the pixel resistor, clk2 and $\overline{\text{clk2}}$ the fuse and thereby the network resistor. Clk3 enables the NMOS precharge switches, $\overline{\text{clr}}$ the PMOS transistor M15 that short-circuits the sense amplifier in $\phi_3$. In figure 2.6 a standard network cycle is depicted. 'Standard' in this context means the SC resistive fuse circuit is emulating the continuous time network as it has been explained in section 1.3.1: one cycle consists of two phases, with four subdivisions each. The different clock signals are activated in the order of $\phi_1$ to $\phi_3$, between $\phi_1$ and $\phi_2$ one time slot is used to invert the $V_{\text{cmp}}$ and $\overline{V_{\text{cmp}}}$ signals.

By controlling clk2p independently of clk2/$\overline{\text{clk2}}$ the characteristic length of the array can be changed. In eq.1.22 the characteristic length was calculated with the assumption that the cycle frequency $f_{\text{cycle}}$ is the same for the fuse and the pixel. Rewriting it without cancelling down $f_{\text{cycle}}$, i.e. $f_{\text{fuse}} \neq f_{\text{pixel}}$, gives:

$$l = \sqrt{\frac{R_{\text{pixel}}}{R_{\text{fuse}}}} = \sqrt{\frac{f_{\text{fuse}} C_{\text{a}}}{f_{\text{pixel}} C_{\text{p}}}} \tag{2.19}$$

It is possible to decrease $f_{\text{pixel}}$ by omitting clk2p during certain cycles. If it is left out $x$ times per second, $f_{\text{pixel}}$ is reduced to $f_{\text{pixel}} - x$, increasing $l$. By this way every $l' \geq l$ can be selected:

$$\frac{l'}{l} = \sqrt{\frac{f_{\text{pixel}}}{f_{\text{pixel}} - x}} = \sqrt{\frac{n}{m}} \qquad n, m \in \mathcal{N}, n \geq m \tag{2.20}$$

In $m$ from $n$ cycles clk2p is activated together with clk2. If $l$ was to be doubled for example, $n/m = 4$ and clk2p must be high every fourth cycle. Clk2p can be activated in both phases of the cycle. In principle it is possible to leave it out in one phase only to change $l$ but this would make the phases differing from each other. Therefore it seems better to implement the clk2p patterns at cycle boundaries, i.e. clk2p is either active in both phases or inactive, it does not change within the cycle.

Decreasing $l$ by omitting clk2/$\overline{\text{clk2}}$ during $\phi_2$ is also possible. Since the node will be isolated if the fuse is not activated this does not change the ratio of $f_{\text{fuse}}$ to $f_{\text{pixel}}$. The pixel resistor charges the node capacitor $C_{\text{n}}$ for a longer time instead. In the limit it can reach the pixel voltage. With $i$ being the number of clk2p cycles elapsed without clk2/$\overline{\text{clk2}}$, the node voltage can be calculated as follows (see appendix D):

$$V_{\text{n}i} = V_{\text{p}}(1-\lambda_{\text{np}}^i) + \lambda_{\text{np}}^i V_{\text{n}0} \tag{2.21}$$

$\lambda_{\text{np}}$ is defined as:

$$\lambda_{\text{np}} = \frac{C_{\text{n}}}{C_{\text{n}}+C_{\text{p}}} \tag{2.22}$$

The larger $i$ is, the closer $V_{\text{n}i}$ comes to $V_{\text{p}}$. This is similar to having a larger $C_{\text{p}}$, thus decreasing $l$. For $i \to \infty$ the effective $C_{\text{p}}$ becomes as large as $C_{\text{n}}$. This sets a lower limit for $l'/l$:

$$\left(\frac{l'}{l}\right)_{\text{min}} = \sqrt{\frac{C_{\text{p}}}{C_{\text{n}}}} \approx 0.5 \tag{2.23}$$

$\phi_3$ uses two clock signals: clk3 and $\overline{\text{clr}}$. In the standard cycle timing they are the complement of each other. While they are active the $V_{\text{out}}$/$\overline{V_{\text{out}}}$ lines are held at $V_{\text{precharge}}$ outside of the array.

### 2.2.4 Edge Readout

One of the outstanding features of the SC implementation of the resistive fuse principle is the ability to read out the digital edge information directly without using additional power. As explained in the previous sections the combination of the sense amplifier states of two consecutive phases determines if the fuse is blown or not (see table 1.2). To read out the edges it is therefore necessary to transmit the state of the sense amplifier out of the array.
There are two different ways the sense amplifier states for both phases can be read out:

**Column-by-column** The data of both phases is read out column-by-column. This mode changes the standard timing only slightly and reads out one column per network cycle.

**Burst** The data of one phase is read out for the whole array before the next phase begins. With the burst mode the total of fuses can be read out in one network cycle.

#### Column-by-column readout

The timing of the column-by-column readout mode is based on the standard timing. In figure 2.3 it has been shown that clk3 is not a global but a column signal. Figure 2.7 depicts the readout timing for two adjacent columns, $n$ and $n+1$. The $V_{\text{out}}$ and $\overline{V_{\text{out}}}$ lines connect one row of cells. With the *precharge* signal they get connected to the precharge voltage source. Unlike in the standard timing *precharge* is not always active.

After $\phi_2$ the sense amplifier has switched to one state, either SA or $\overline{\text{SA}}$. Since M5 and M6 are disabled it is not actively driven anymore. The information is stored capacitive. Due to the increased $V_{\text{GS}}$ voltage of either M1/M4 (state SA) or M2/M3 (state $\overline{\text{SA}}$) the capacity is much larger than $C_{\text{g}}$ in table 2.1. The channel capacity of 45 fF adds to each side. Like in a DRAM[8] the

[8]Dynamic Random Access Memory

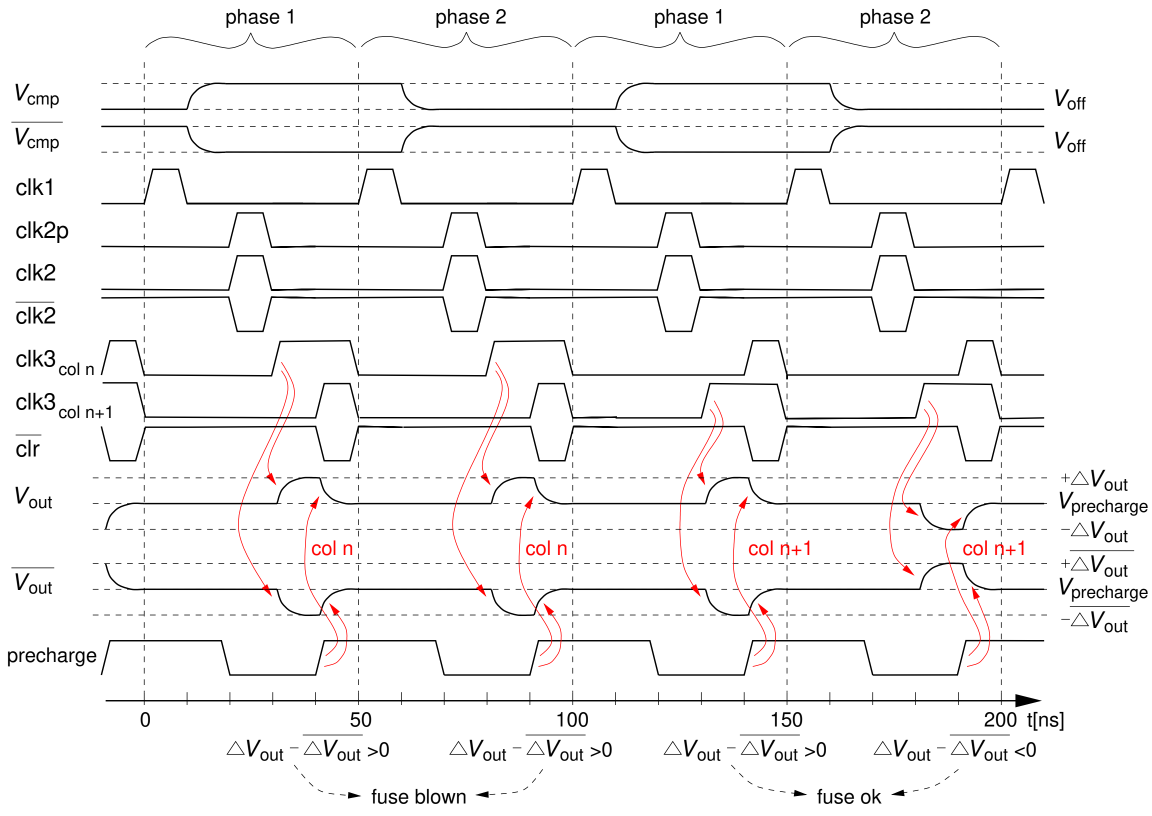



**Figure 2.7:** Column-by-column readout timing. Clk3 is depicted for column $n$ and $n+1$. $V_{\text{out}}$ and $\overline{V}_{\text{out}}$ show the output voltages of an example row.

charge stored on the sense amplifier is read out directly by connecting the sense amplifier to $V_{\text{out}}$ and $\overline{V}_{\text{out}}$ in clk3. This is done as usual by M13 and M14. The differences to the standard timing are:

- Only one column gets the clk3 signal.
- The precharge voltage source is not activated. The $V_{\text{out}}$ and $\overline{V}_{\text{out}}$ lines have been precharged in the previous phase and are floating now.
- The $\overline{\text{clr}}$ signal is not asserted.

By this way the charge on the sense amplifier and on the $V_{\text{out}}/\overline{V}_{\text{out}}$ lines is shared. As can be seen in figure 2.7 this changes $V_{\text{out}}$ to $V_{\text{precharge}} + |\Delta V_{\text{out}}|$. The sign of $\Delta V_{\text{out}}$ depends on the side the storage amplifier has switched to. $\overline{V}_{\text{out}}$ changes to $V_{\text{precharge}} + |\overline{\Delta V}_{\text{out}}|$. $\Delta V_{\text{out}}$ and $\overline{\Delta V}_{\text{out}}$ can be calculated as follows:

$$\Delta V_{\text{out}} = \frac{C_{\text{sa2}}}{C_{\text{sa2}} + C_{\text{out}}}(V_{\text{sa}} - V_{\text{precharge}}) \tag{2.24}$$

$C_{\text{sa2}}$ is the capacitance of the charged sense amplifier, i.e. the sense amplifier after $\phi_2$. $C_{\text{out}}$ is the total capacitance of the $V_{\text{out}}$ line as seen by the sense amplifier. It is about 3 pF with 66 array elements connected in a row[9]. Taking the difference between $\Delta V_{\text{out}}$ and $\overline{\Delta V_{\text{out}}}$ gives the differential output voltage $V_{\text{diffout}}$:

$$\begin{aligned} V_{\text{diffout}} = \Delta V_{\text{out}} - \overline{\Delta V_{\text{out}}} &= \frac{C_{\text{sa2}}}{C_{\text{sa2}} + C_{\text{out}}}(V_{\text{sa}} - V_{\text{precharge}} - (\overline{V_{\text{sa}}} - V_{\text{precharge}})) \\ &= \frac{C_{\text{sa2}}}{C_{\text{sa2}} + C_{\text{out}}}(V_{\text{sa}} - \overline{V_{\text{sa}}}) \end{aligned} \tag{2.25}$$

Since $|V_{\text{sa}} - \overline{V_{\text{sa}}}| = V_{\text{DD}}$ this reduces to:

$$V_{\text{diffout}} = \underbrace{\frac{C_{\text{sa2}}}{C_{\text{sa2}} + C_{\text{out}}}}_{\approx 170\text{ mV}} V_{\text{DD}} \cdot \begin{cases} 1 & \text{state SA} \\ -1 & \text{state } \overline{\text{SA}} \end{cases} \tag{2.26}$$

This can be reliably detected in the row readout circuitry outside the array (see 2.3.2). This calculation has assumed voltage independent capacitances. This is valid for $C_{\text{out}}$ since the voltage change $\Delta V_{\text{out}}$ is small. Different from that $C_{\text{sa2}}$ gets reduced to $C_{\text{sa}}$ if the sense amplifier is discharged. This happens at $V_{\text{precharge}} - V_{\text{T}} < V_{\text{sa}} < V_{\text{precharge}} + V_{\text{T}}$. If this is taken into account $|V_{\text{diffout}}|$ will be reduced to 148 mV. The expected error due to the uncertainty in the absolute value of $C_{\text{out}}$ is larger since $C_{\text{out}}$ varies with $V_{\text{precharge}}$[10] and process variations[11]. Considering the worst case the readout amplifier must have a sensitivity of about 100 mV or better.

After the readout amplifier has sampled $V_{\text{diffout}}$ the precharge voltage is switched on together with the clk3 and $\overline{\text{clr}}$ signals of all columns. This ends the actual phase of the cycle like in the standard timing and precharges the sense amplifiers. The precharge signal stays active until clk3 is low again to make sure that the $V_{\text{out}}/\overline{V_{\text{out}}}$ lines are precharged for the next read cycle as well. If it was to be deactivated together with clk3 the charge injection of M13 and M14 will change $V_{\text{out}}$. Since a slight difference between M13 and M14 is multiplied by the number of cells in a row this would introduce a voltage difference between $V_{\text{out}}$ and $\overline{V_{\text{out}}}$.

In the next phase the readout cycle repeats with different $V_{\text{cmp}}$ polarity. After phase 2 the information necessary to determine the fuse state has been gathered by the readout circuit for column $n$. The fuse states for column $n+1$ are read out in the next cycle. The only difference is the point in time clk3 is activated for the individual columns. The readout amplifier samples $V_{\text{diffout}}$ from that column whose output transistors are enabled before $\overline{\text{clr}}$ and *precharge* become active.

The readout circuitry is able to store the fuse state data of one column. It is possible to mix standard network cycles with column-by-column readout cycles. By this way the network can run continuously while the readout circuitry transfers the edge data out of the chip. If the column storage is free, the next cycle will be a readout one. If not, standard network cycles are performed.

### Burst readout

Burst mode readout uses $\phi_3$ to transfer the state of all sense amplifiers out of the array. This is illustrated in Figure 2.8. After $\phi_2$ of phase 1 clk3 of each column is activated once. The

[9] It is the sum of the capacitances of the metal wiring, the drain-bulk junctions of M13 and M14 ($\times$ 66), the gate-drain overlap of M13 and M14 ($\times$ 66), the readout-amplifier input and the precharge voltage switch.

[10] This is caused by the voltage dependent drain-bulk junction capacitance.

[11] Metal capacitances have worst case errors up to $\pm$ 50 %.

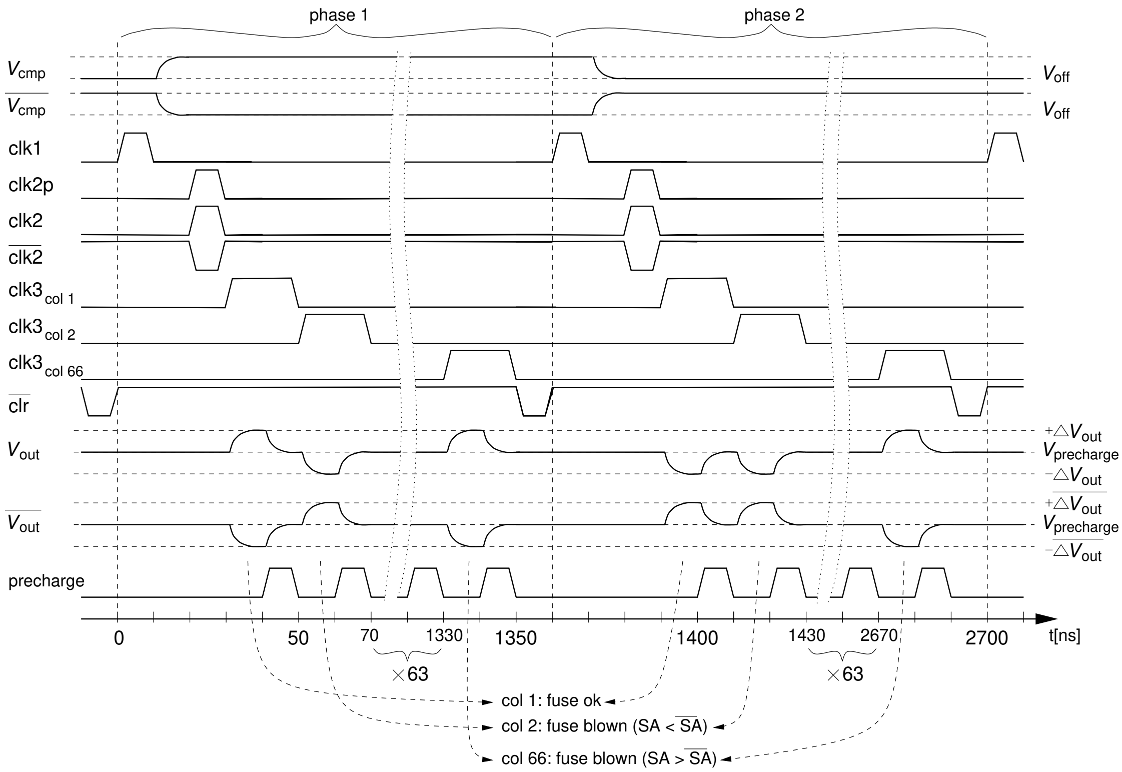



**Figure 2.8:** Burst readout timing. Clk3 is shown for column 1, 2 and 66. The ones in-between have been omitted for clarity. The complete array is read out in the depicted cycle.

readout takes place by charge sharing between $V_{out}/\overline{V_{out}}$ and the sense amplifier, as described in the previous section. The differential voltage on the $V_{out}/\overline{V_{out}}$ lines is cleared by the precharge signal. Afterwards the next column's clk3 signal gets high. If all 66 columns are read out, $\overline{\text{clr}}$ will be asserted and the next phase starts. The same pattern is repeated in phase 2 to acquire the sense amplifier states for the inverted $V_{cmp}$ voltages as well.

The burst mode has the advantage of taking a 'snapshot' of the sense amplifier states. The node voltages cannot change between the readout of the different columns because clk2 stays inactive. It is possible to monitor every change in the network by performing only burst readout cycles. The downside is the need to buffer the phase 1 edge data for the complete array before the second half can be read out in phase 2. 132 bits per column must be transferred out of the array. For 66 columns a storage capacity of 8712 bits is necessary. If the data transmission to this memory is not fast enough wait-states have to be inserted between adjacent clk3 signals.

## 2.3 Row and Column Circuits

After the previous section has covered the network cells in detail, this section focuses on the analog and digital support circuitry located at the edges of the array. They provide the different clock and control signals needed by the array elements as well as the analog I/O.

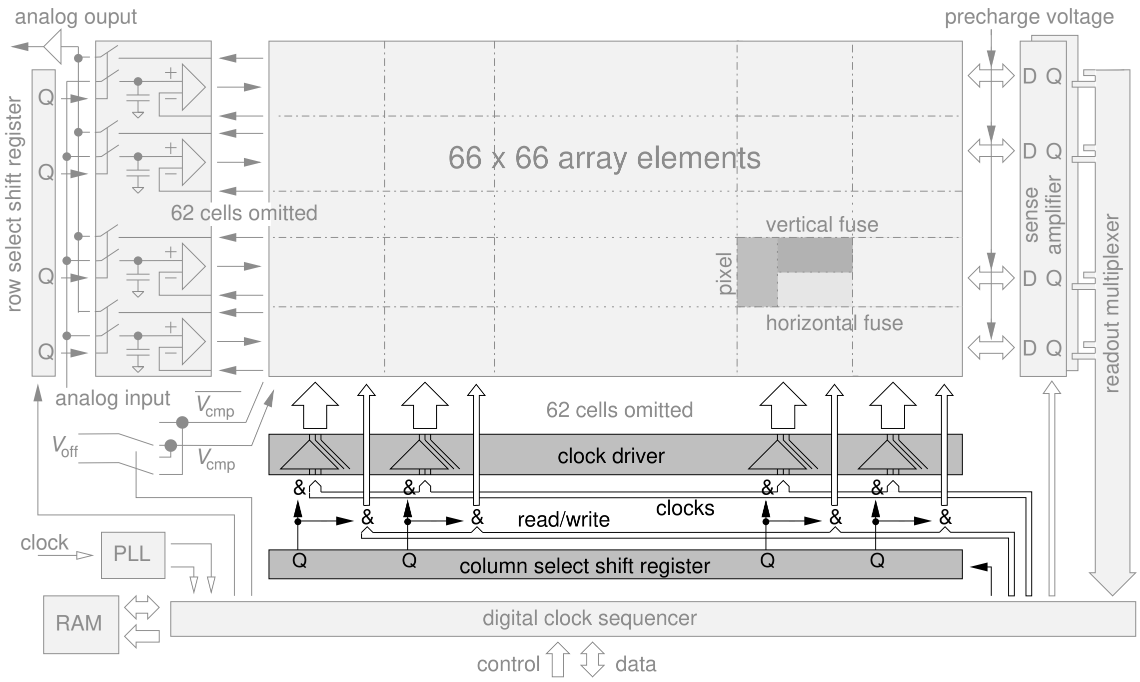



**Figure 2.9:** EDDA block diagram. The column circuits are highlighted.

### 2.3.1 Column Clock Driver Circuits

The column signals are shown in figure 2.3: $\overline{\text{read}}$, write, $\overline{\text{write}}$ and clk3 are used to select a single column of the array[12]. The clock signals clk1, clk2p, clk2, $\overline{\text{clk2}}$ and $\overline{\text{clr}}$ are logically the same for all the columns but electrically individual. In figure 2.9 the column circuits are highlighted. They consist of one row of identical cells. Each cell is connected to one array column. Global signals from the digital control part are running in parallel to the cell-row providing the master clock and control signals. Figure 2.10 depicts the schematic diagram of a column driver cell. At the top the clock drivers are shown. The global signals have the letter 'g' prefixed. The numbers inside the gates and inverters are the PMOS and NMOS widths. The gate length is always 0.6 $\mu$m.

The flipflops (FF) of all the column cells form an array wide shift register. They are standard d-type flipflops with a zero hold time and complementary outputs Q and $\overline{\text{Q}}$. The signal at the d-input (D) is stored with the raising edge of the clock signal (C) . Csout is connected to csin of its right neighbor. The first column gets it csin signal from the digital control logic who also generates the clock (csclk) and the reset ($\overline{\text{csres}}$) signals. By shifting a one-bit through the register every column can be selected. The logic for the clk3 signal takes two inputs: gclk3 and gclk3read. The gclk3read signal activates only the clk3 line for the selected column, i.e. the one with its FF's Q-output high. If gclk3 is high all the clk3 lines of the array will be activated. The global read and write enable signals, gread and gwrite, are combined with the Q-output by a logical NAND to generate the column $\overline{\text{read}}/\overline{\text{write}}$ signals (write = $\overline{\text{Q}}$ NOR $\overline{\text{gwrite}}$).

Since there is no need for the fast selection of a certain column (besides the next one) the usage of a shift register instead of an address decoder is of advantage. It makes it possible to select

[12]Read: node voltage readout, write/$\overline{\text{write}}$: source voltage storage, clk3: fuse state readout

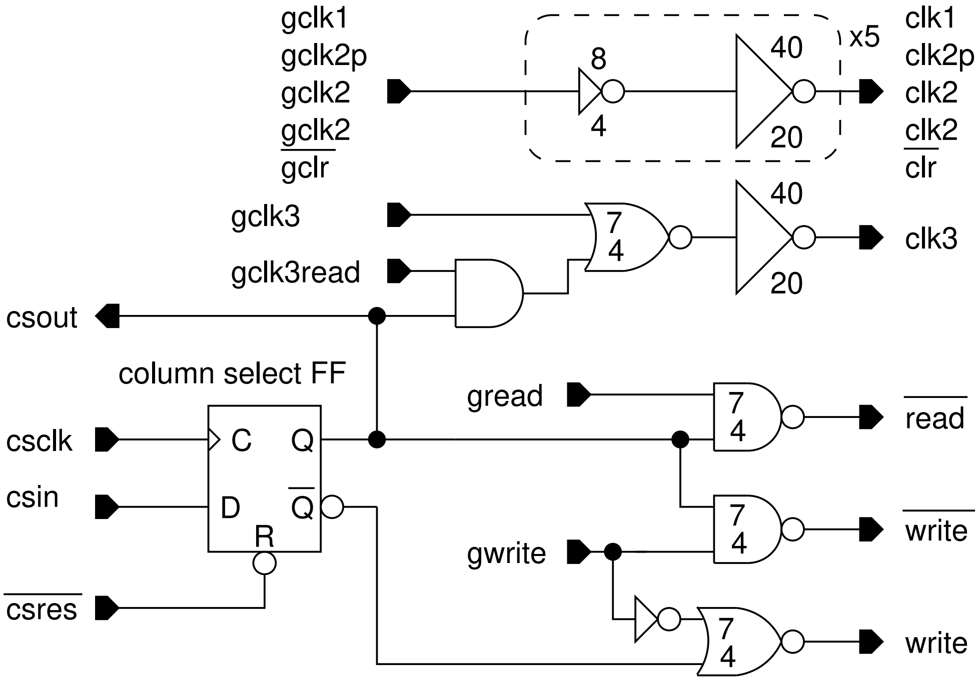



**Figure 2.10:** Schematic diagram of the column circuits.

more than a single column at once. This allows to copy sense amplifier states from one column to another for example. Even though this might not be useful for the resistive fuse network it establishes experimental possibilities.

The large inverters have 40/20 $\mu$m wide transistors. They can deliver enough current to drive one array column with a maximum rise/fall time of 1.5/1 ns[13]. This is 5.7/5 times the width of a single drive strength output from the AMS 0.6 $\mu$m digital standard cell library. Since the charge carrier mobility of the PMOS is only 1/3 compared to the NMOS in this process a 60/20 ratio would have been necessary to get equal rise and fall times[14]. This would have increased the capacitance of the global clock lines by 33 % as well as the column cell area. The total speed increase would have been 20 %. Therefore the standard cell library's ratio of PMOS to NMOS gate widths of 1.75 was only slightly increased to 2 for the column drivers. The $\overline{\text{read}}$, write and $\overline{\text{write}}$ signals are not as speed critical as the clocks. Thus standard drive strength suffices.

### 2.3.2 Row Readout and Storage Circuits

As can be seen in figure 2.3 there are two pairs of signals for each array row: horizontal $V_{\text{out}}/\overline{V_{\text{out}}}$ and vertical $V_{\text{out}}/\overline{V_{\text{out}}}$. They are used for the analog as well as the digital (fuse states) I/O. The terms horizontal and vertical refer to the fuses which use the according $V_{\text{out}}/\overline{V_{\text{out}}}$ lines. Both pairs connect the array cells of one row.

[13]The rise/fall time is the time it takes for the output to change from $V_{\text{T}}$ to $V_{\text{DD}}$-$V_{\text{T}}$ or from $V_{\text{DD}}$-$V_{\text{T}}$ to $V_{\text{T}}$, respectively.

[14]The drain current is proportional to the transconductance parameter $K' = \mu_{\text{x}} C_{\text{Ox}}$. While $C_{\text{Ox}}$ is the same for NMOS and PMOS devices the electron mobility $\mu_{\text{n}}$ is three times the hole mobility $\mu_{\text{p}}$.

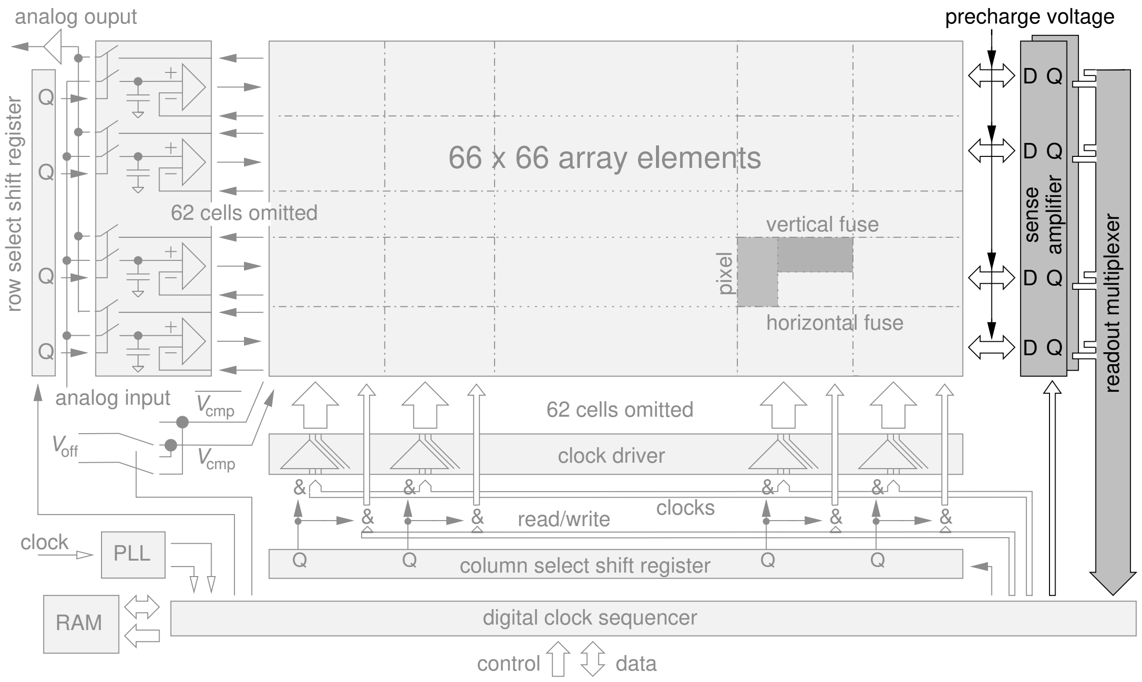



**Figure 2.11:** EDDA block diagram. The fuse state readout and precharge circuits are highlighted.

### Fuse state readout and precharge

In figure 2.11 the fuse state readout and precharge circuits are highlighted. To each array row belongs one readout cell. In the schematic diagram, shown in figure 2.12, the four row signals have been drawn on the left. Each pair connects to two sense amplifiers (SA): the vertical $V_{\text{out}}/\overline{V_{\text{out}}}$ to SA1 and SA3, the horizontal to SA2 and SA4.

**Precharge switch** There are two precharge switches: M1 to M4 and M5 to M8. As can be seen by the W/L ratio M1-M4 are 10 times larger than M5 to M8. Pcon (*precharge on*) enables the large one, rpcon (*readout precharge on*) the small one. Only NMOS transistors are used as $V_{\text{precharge max}}$ is smaller than 2.5 V (see eq.2.13). The large switch is used to provide the precharge voltage to the fuses in the timing modes where clk3 is asserted for all the array columns at once. These modes are the standard and the column-by-column readout timing. The on-resistance of M1 to M4 is low enough to provide the current necessary to precharge all the fuses in one row. In readout mode they have to be enabled and disabled once per phase, i.e. twice every cycle. The sum of their gate capacitances is about 11 pF, thus an energy of 275 pJ[15] is needed for the precharge switch in every phase. In burst readout mode the fuses are precharged column-by-column. This increases the power consumption of M1 to M4 by a factor of 66 per phase. But the on-resistance of the switch can be much larger in burst mode than in column-by-column readout mode since in each precharge cycle clk3 is activated for one column only. Therefore a small precharge switch M5 to

[15]The energy needed to charge a capacitor $C$ to a voltage $V$ is $CV^2$. Half of this energy is dissipated by the charging resistor, the other half when the capacitor is discharged. The total energy is lost (converted to heat in the transistors' channels). Thus the power consumption per cycle is $CV^2$.

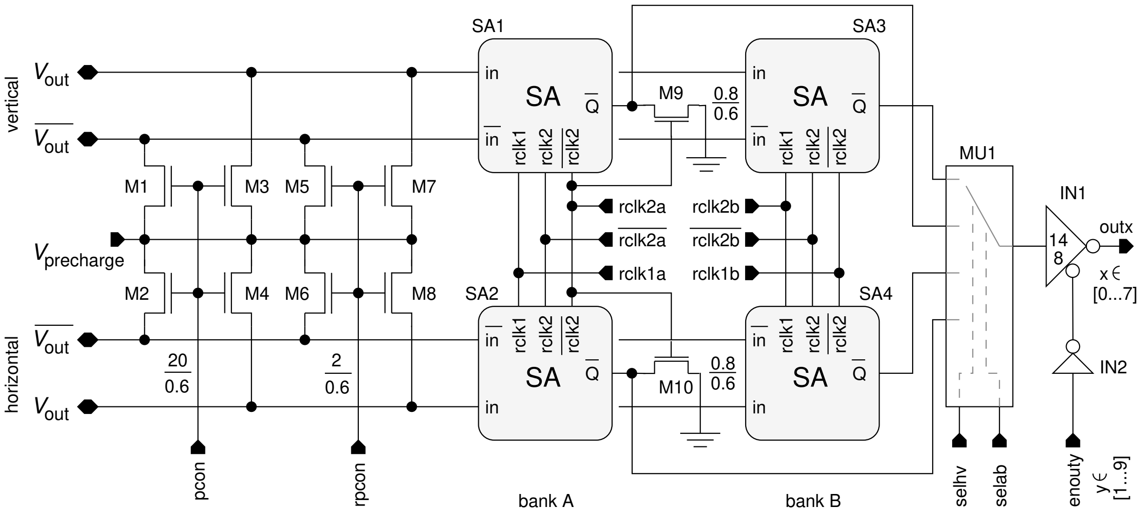



**Figure 2.12:** Schematic diagram of the fuse state readout and the precharge switches (M1-M8).

M8 is connected in parallel to the large one, reducing power consumption by a factor of 10. The wire capacitance of the pcon or rpcon line is about 700 fF. This is the same order of magnitude than the capacitance reduction achieved by further decreasing the transistor width from 2 $\mu$m to the process limit of 0.8 $\mu$m. But this will increase the maximum charge injection error and the worst case on-resistance, therefore 2 $\mu$m has been chosen as a compromise.
The precharge voltage input $V_{\text{precharge}}$ is directly connected to an input pad of the chip.

**Readout sense amplifiers** The purpose of the readout sense amplifiers is to restore the differential fuse output signal $V_{\text{diffout}}$ to the full $V_{\text{DD}}$ level. They also hold this information until it is transferred to the digital section. The sense amplifiers are divided in two groups: bank A (SA1 and SA2) - using rclk1a, rclk2a and $\overline{\text{rclk2a}}$ - and bank B (SA3 and SA4) which is controlled by rclk1b, rclk2b and $\overline{\text{rclk2b}}$. These clock signals are common for all readout cells. In column-by-column readout mode bank A stores the data from phase 1, bank B from phase 2. Both banks together contain the edge data of one array column. In burst readout mode the sense amplifier banks are filled alternately. Thereby the data from one bank can be read out while the other bank stores the fuse information. The schematic diagram of the sense amplifier is shown in figure 2.13. The core circuit - consisting of M1 to M8 - is the same as the fuse sense amplifier. The clocking scheme is similar, besides that $\phi_3$ has been merged with $\phi_1$ (rclk1). This is possible because the input lines (in, $\overline{\text{in}}$ are connected to $V_{\text{out}}/\overline{V}_{\text{out}}$) are at $V_{\text{precharge}}$ when rclk1 gets activated. M7 and M8 will be switched on with rclk1, thereby precharging the gates of M1 to M4. In and $\overline{\text{in}}$ change to $V_{\text{precharge}} \pm V_{\text{diffout}}/2$ after clk3 has activated the outputs of the fuse sense amplifiers. Since rclk1 is still high M1 to M4 follow the potential on $V_{\text{out}}/\overline{V}_{\text{out}}$. With $\phi_2$ (rclk2, $\overline{\text{rclk2}}$) M7 and M8 are switched off and power is applied to the sense amplifier by M5 and M6 thus amplifying $V_{\text{diffout}}$ to $V_{\text{DD}}$. This timing is illustrated in figure 2.14.

One half of the sense amplifier - the drains of M1 and M2 - is connected to the input of a clocked inverter (M9 to M12). Its clock inputs are connected to rclk2/$\overline{\text{rclk2}}$ and it drives the sense amplifier output $\overline{\text{Q}}$. Its purpose is to isolate the sense amplifier nodes from the output wire ca-

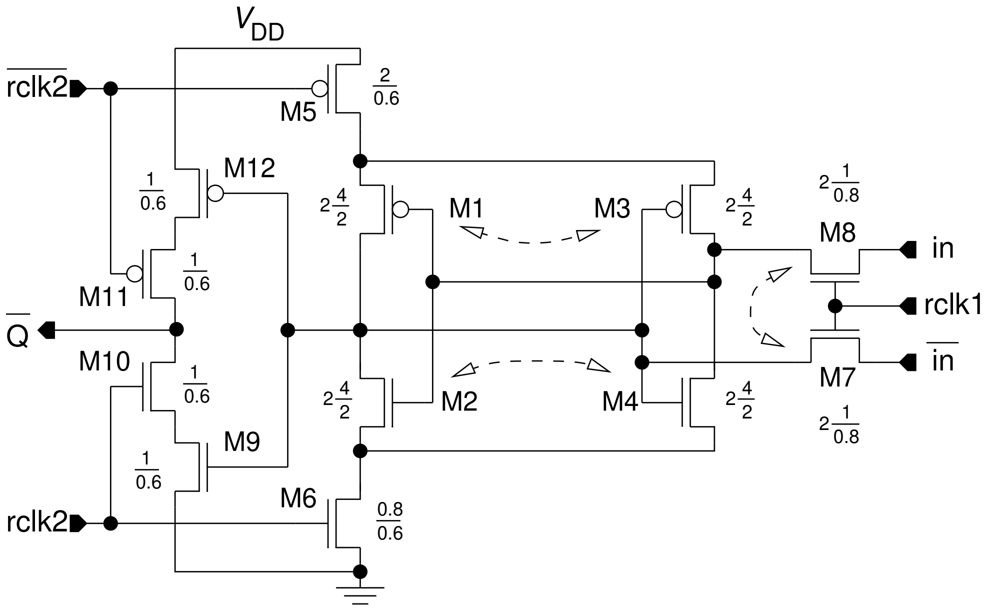



**Figure 2.13:** Schematic diagram of the readout sense amplifier. The dashed arrows depict matching transistors.

pacitance in $\phi_1$. The gate capacitances of M9 and M12 are minimized to keep the sense amplifier capacitance symmetric. M10 and M11 are necessary to prevent any current flow through the inverter if rclk1 is active and the gate voltage is $V_{\text{precharge}}$. This would cause a quiescent current flowing through M9 and M12 since neither of them is shutoff completely. The symmetric realization of the inverter (instead of using the usual PMOS/NMOS gate width ratio) helps to cancel the charge injection of M10 and M11 when rclk2 changes.

After the sense amplifier has made its decision it acts like a static latch. Since rclk1 is low it is isolated from $V_{\text{out}}/\overline{V_{\text{out}}}$ and the edge readout can continue with the other bank of sense amplifiers. The edge data is stored as long as rclk2 is active.

**Output multiplexing** The two banks of sense amplifiers store the edge information for one column (in column-by-column readout mode). Circuits for further processing can be integrated directly next to them. Thereby they can utilize the full data rate of about 550 MByte/s[16](in burst readout mode). The EDDA chip - as it is presented in this thesis - has been designed to evaluate the SC resistive fuse network concept. It does not contain additional image processing hardware yet. Therefore the data stored in the readout sense amplifiers can be read out via the synchronous bus serving as the external interface of the chip. Its width is 8 bit. Parallel to the row readout cells 8 data lines form the output bus for the edge data. The row cells are grouped in 8 clusters, each containing 8 cells connected to the output bus by tristate drivers (IN1 in figure 2.12). The cells of a cluster share a common output enable signal *enouty* (enable output of cluster no. *y*). Since there are 66 rows, the outermost rows form a 9th cluster containing only two cells.

[16]This is the product of the amount of data read out in parallel - 132 bit - and a readout cycle time of 30 ns.

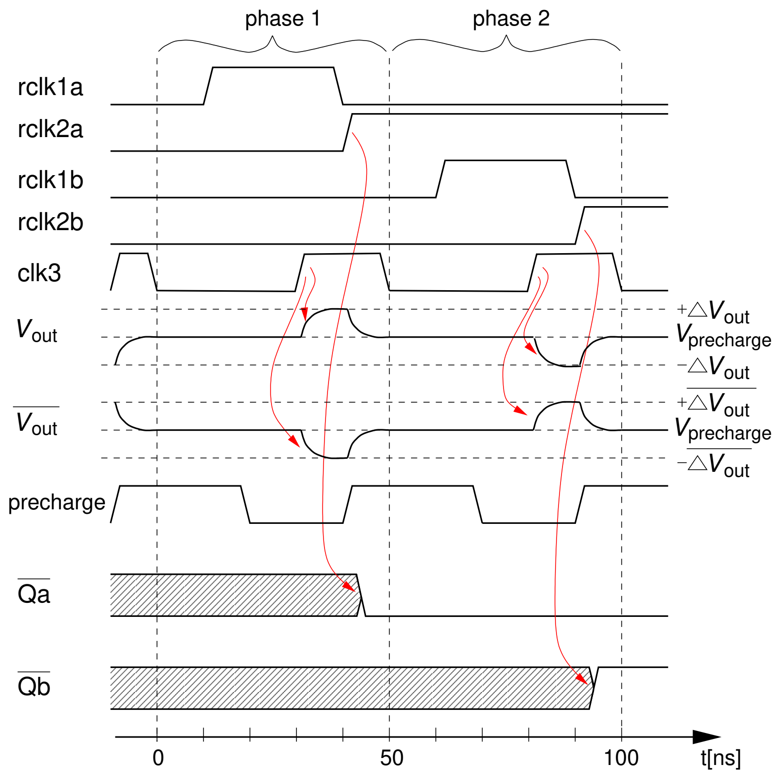



**Figure 2.14:** Timing of the readout sense amplifier clock signals in column-by-column readout mode. Clk3 is shown for an arbitrary column. With rclk1a/rclk2a phase 1 data is stored ($\overline{\text{Qa}}$), phase 2 data ($\overline{\text{Qb}}$) with rclk1b/rclk2b respectively.

The four bits of a row readout cell - horizontal bank A/B and vertical bank A/B - are switched by a 4-to-1 multiplexer (MUX1) to the output driver IN1. Two global select signals, *selhv* (select horizontal/vertical) and *selab* (select bank A/B) determine which sense amplifier output $\overline{\text{Q}}$ is fed to the cell output outx, where $x$ represents the bit number inside the cluster. To avoid floating inputs at the multiplexer if no rclk2 line is active, M9 and M10 connect two of the inputs to ground[17]. The output tristate driver is a clocked inverter circuit. It has double drive strength (14/8) and therefore fourfold gate widths[18]. To reduce the capacitive loading of the enouty lines they are buffered by single drive strength inverters (IN2).

### Analog readout and storage

In figure 2.15 the analog readout and storage circuits are highlighted. Like the digital readout they connect to the $V_{\text{out}}/\overline{V_{\text{out}}}$ lines of the array rows. Their schematic diagram is depicted in figure 2.16. The horizontal $V_{\text{out}}$ is unused. The node voltage readout is done via $\overline{V_{\text{out}}}$. The vertical $V_{\text{out}}/\overline{V_{\text{out}}}$ pair connects the storage amplifier A1 to the network. The row-select flipflop selects the row for storage and readout. Like the column-select FF the flipflops of all rows form a shift

[17]This is not necessary for the remaining two inputs because they are disabled if selab and selhv are low.

[18]In this circuit two transistors are connected in series between the output and $V_{\text{DD}}$ or ground, respectively. To achieve double drive-strength their gates must have four times the standard width.

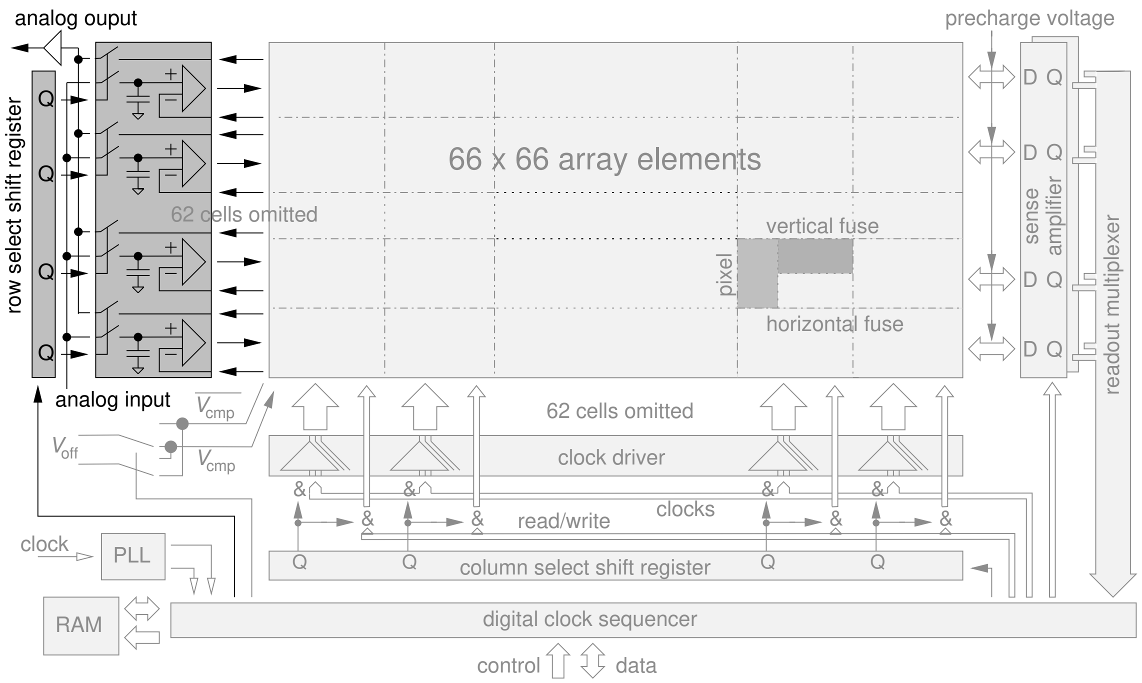



**Figure 2.15:** EDDA block diagram. The analog readout and storage circuits are highlighted.

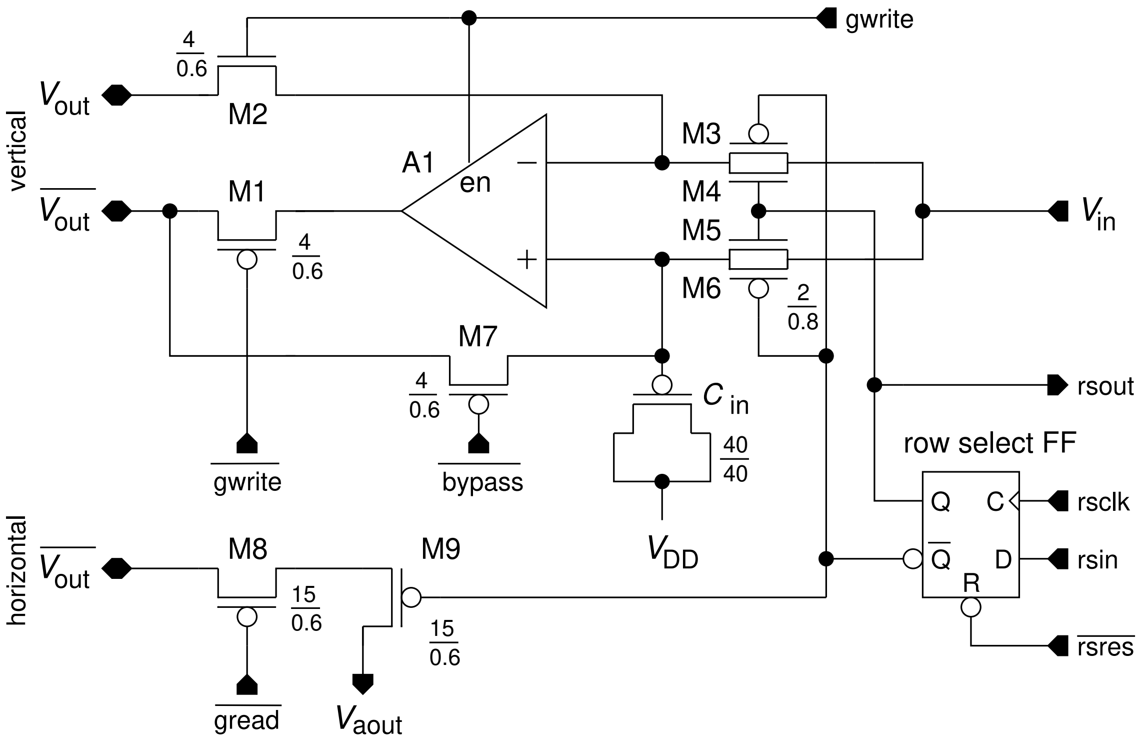



**Figure 2.16:** Schematic diagram of the analog readout and storage circuits. M3 to M6 have the same W/L of 2/0.8.

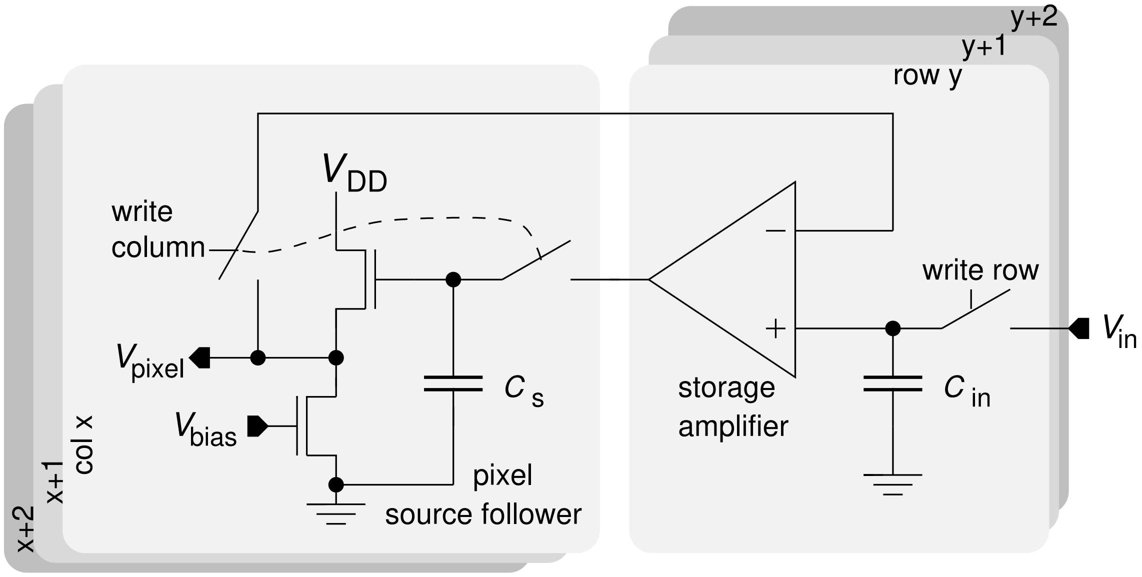



**Figure 2.17:** Operation principle of the input voltage storage circuit.

register with a common clock (rsclk) and reset ($\overline{\text{rsres}}$). The FF at the bottom gets its input (rsin) from the digital control. The output (rsout) is connected to its top neighbor. The shift register has the advantage that both single and multiple rows can be selected.

**Node voltage readout** Transistors M8 and M9 do the readout multiplexing. Each network node has a source follower to output the node voltage. In readout mode - when gread/$\overline{\text{gread}}$ are active - the source followers of one array column are connected to the horizontal $\overline{V_{\text{out}}}$ lines (see 2.2). The row is selected by M9 which gets its gate voltage from the $\overline{\text{Q}}$ output of the row select FF. Both M8 and M9 have a high W/L to reduce their on-resistance. Thereby it is much smaller than the output impedance of the source follower itself. This keeps the voltage drop across M8 and M9 small and avoids row-by-row fixed pattern noise due to variations in M8 or M9 between different rows. In [LOV98] a $I_{\text{DS}}$ mismatch of less than one percent ($\sigma(\Delta I_{\text{DS}})/I_{\text{DS}}$) was measured for comparable devices. This limits the fixed pattern noise caused by M8 and M9 to about 1 mV ($1\sigma$).
M8 isolates $\overline{V_{\text{out}}}$ from M9 if the node voltage readout is off. This allows the sampling of new analog data without disturbing the horizontal output lines. Otherwise the gate-drain capacitance of M9 would inject charge on $\overline{V_{\text{out}}}$ when the FF toggles. Digital readout and the sample-and-hold (S&H) of the analog input data can work at the same time.

**Input voltage storage** The analog input uses a two level storage scheme. It is illustrated in figure 2.17. The chip's analog input $V_{\text{in}}$ is connected to every row storage cell. The capacitor $C_{\text{in}}$ will store $V_{\text{in}}$ if its row is selected. Out of the array columns one is selected for write. The pixel source followers of that column are connected to the row storage amplifiers. The combination of storage amplifier and pixel source follower resembles a unity gain buffer since the pixel output is fed back to the negative input terminal of the operational amplifier. By this way the voltage drop across the source follower gets compensated. As this calibration is done individually for each pixel, it also suppresses pixel-by-pixel variations.

The storage amplifier A1 is a folded cascode type [GEI90-7, GRA93, LAK94-1]. It has an output voltage swing of nearly $V_{\mathrm{DD}}$ to ground and is capable of driving the capacitive load of the $\overline{V_{\mathrm{out}}}$ line - which is about 3 pF - without any frequency compensation. The simulated slew rate is 3.4 V/$\mu$s under this loading condition. Its low frequency gain is about 500. The simulated voltage error of the complete feedback loop is -3.8 mV at $V_{\mathrm{in}} = 0.4$ V and 2.6 mV at $V_{\mathrm{in}} = 3$ V. This is a compromise between low area usage and low power consumption on one side and speed on the other. The complete amplifier needs not more than 30 $\mu$A (including bias generation). Since this current is wasted if the global write signal (gwrite/$\overline{\text{gwrite}}$) is inactive, the amplifier has an enable input. If it is low the bias circuit is disabled and power consumption drops to leakage current levels.

As can be seen in the schematic (figure 2.16) the global write signals will also disconnect A1 from the vertical $V_{\mathrm{out}}/\overline{V_{\mathrm{out}}}$ pair - via transistors M1 and M2 - if they are inactive. This avoids any unbalanced capacitive loading on the output lines during digital readout caused by the disabled amplifier. Their on-resistance is much smaller than the output resistance of A1 and the pixel source follower. Since the output voltage of A1 is 1.2 to 1.5 volts higher (depending on the pixel bias voltage) than $V_{\mathrm{in}}$ M1 is a PMOS-FET.

The S&H circuit consists of $C_{\mathrm{in}}$ together with M5 and M6. They form a transmission gate connecting $V_{\mathrm{in}}$ to the storage capacitor $C_{\mathrm{in}}$ if the row select FF's Q-output is high. The symmetric transmission gate minimizes charge injection. In addition $C_{\mathrm{in}}$ is rather large, its capacitance of approximately 4.4 pF reduces the worst case voltage error due to row-to-row mismatch in M5 and M6 clearly below 1 mV. Like the network capacitors $C_{\mathrm{in}}$ is also a PMOS transistor to cover the input voltage range. Its n-well is isolated from the other circuitry and tied to $V_{\mathrm{DD}}$ by a low impedance connection.

M3 and M4 form a second transmission gate. Its purpose is the speed-up of the feedback loop and the suppression of voltage errors due to the amplifier's input capacitance. It connects the negative input terminal of A1 to $V_{\mathrm{in}}$. By this way the input stage of A1 as well as the $V_{\mathrm{out}}$ line are precharged to nearly their final voltage (since the gain of A1 is about 500 the differential input voltage after the feedback loop has settled to $V_{\mathrm{in}}$ is only a few millivolts). If the negative input terminal is not to be precharged to $V_{\mathrm{in}}$ the feedback loop would change its voltage until both inputs are at nearly equal potential. This can imply a maximum voltage shift of more than 2 volts at the negative input (as large as the total dynamic range of $V_{\mathrm{in}}$). Since this changes the common source voltage ($V_{\mathrm{source}}$) of A1's PMOS input transistors the gate-drain voltage of the positive input's transistor changes by $\Delta V_{\mathrm{DS}} = \Delta V_{\mathrm{source}}$ (since $C_{\mathrm{in}}$ holds its gate at $V_{\mathrm{in}}$). The input capacitance of A1 is 200 fF. Compared to $C_{\mathrm{in}}$ this results in a voltage error of 44 mV/$\Delta V_{\mathrm{DS}}$. The precharging of the negative input solves this problem. After the feedback loop has settled $V_{\mathrm{source}}$ is the same as before, i.e. $\Delta V_{\mathrm{DS}} = 0$.

The pixel source follower is the slowest part of the feedback loop. Making it as fast as A1 would strongly increase the chip's power consumption. Also it is not possible to increase only the bias of the selected column because the voltage drop between the storage capacitor $C_{\mathrm{s}}$ and the pixel output voltage is bias-dependent. This is overcome by M3 and M4. By precharging $V_{\mathrm{out}}$ to its final voltage the pixel source follower must only deliver the current for small voltage corrections instead of those covering the full dynamic range.

M7 bypasses the storage amplifier A1. First this is for testing the analog storage circuitry. For example it is possible to measure the voltage drop of the pixel source follower by comparing the

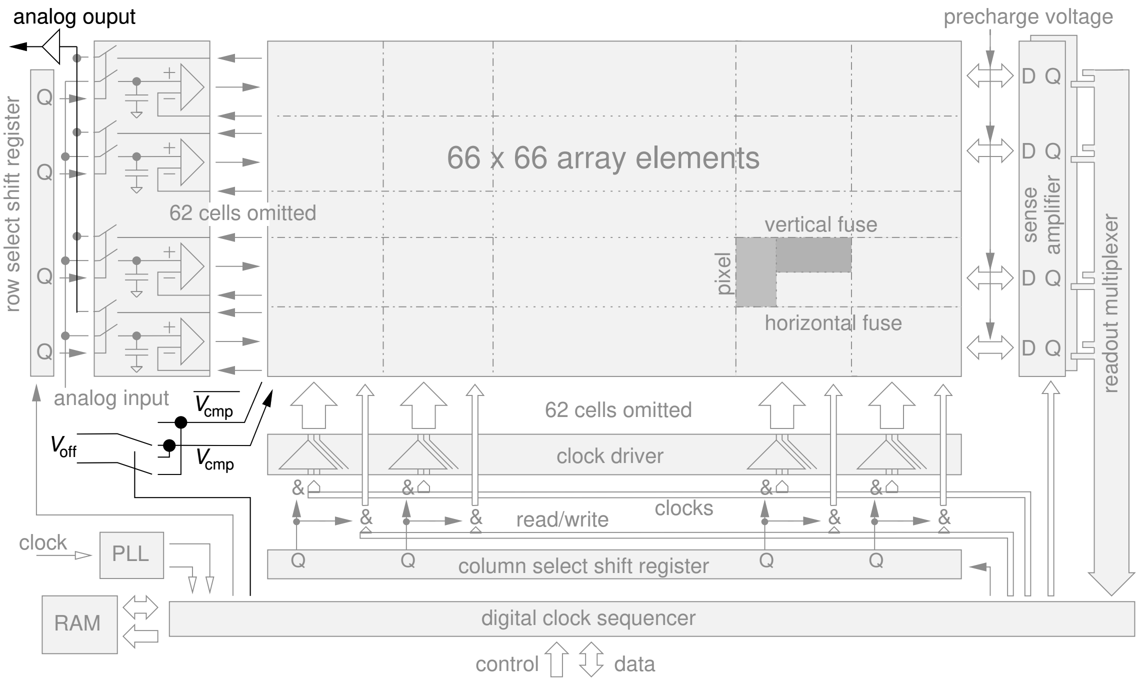



**Figure 2.18:** EDDA block diagram. The $V_{\text{off}}$ inputs and switches as well as the analog readout buffer are highlighted.

node voltage with or without A1. Second it makes it possible to input a differential voltage to the sense amplifiers - either of the fuses or the row-readout - connected to the vertical output lines. This is done by first precharging $V_{\text{out}}$ and $\overline{V_{\text{out}}}$ while storing $V_{\text{precharge}} \pm \Delta V_{\text{test}}$ on the $C_{\text{in}}$'s of all rows. By activating $\overline{\text{bypass}}$ $\overline{V_{\text{out}}}$ changes accordingly. Thereby the voltage resolution and the input offset distribution of the sense amplifiers can be measured.

## 2.4 Global Analog I/O

This section covers the analog circuitry common to the whole array, i.e. not organized in rows or columns. As can be seen in figure 2.18 that includes the analog output amplifier and the switches for the fuses' $V_{\text{cmp}}$ and $\overline{V_{\text{cmp}}}$ inputs.

### 2.4.1 Comparator Voltage Switch

The purpose of the comparator voltage switch is to reverse the polarity of the global $V_{\text{cmp}}$ and $\overline{V_{\text{cmp}}}$ signals. Figure 2.19 shows its schematic diagram. Two single-pole/double-throw analog switches are build from two transmission gates each. M1/M2 and M3/M4 are activated in phase 1 of the array element's clock cycle, connecting $+V_{\text{off}}$ to $V_{\text{cmp}}$ and $-V_{\text{off}}$ to $\overline{V_{\text{cmp}}}$. M5/M6 and M7/M8 are their phase 2 counterpart: $+V_{\text{off}}$ is tied to $\overline{V_{\text{cmp}}}$ and $-V_{\text{off}}$ to $V_{\text{cmp}}$. The phase 1 and phase 2 clock signals must be non-overlapping to avoid a short-circuit of the $V_{\text{off}}$ voltage source.

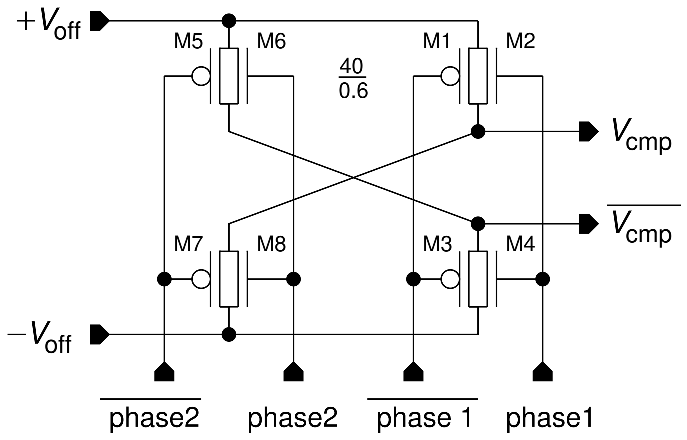



**Figure 2.19:** Schematic diagram of the comparator voltage switch. M1 to M8 have a W/L of 40/0.6.

### 2.4.2 Output Amplifier

The output amplifier buffers the analog node voltage readout signals. It provides a low output impedance and is connected to an output pad of the chip. It also contains the common load transistor for the total of the network's node voltage source followers. This NMOS transistor has a W/L of 200/5 to ensure a good linearity. Its gate is connected to an input pad, thus the readout current can be adjusted. This bias voltage is also used to control the current of the output stage of the buffer. A current mirror ensures that its quiescent current is 20 times higher than that of the node voltage source follower.

The output amplifier has been simulated[19] with an off-chip load capacitance of 50 pF. With current settings of 33 $\mu$A (node readout) and 0.723 mA (output stage) the output voltage settles to 1 % within 500 ns for the rising or 350 ns for the falling edge of the input signal. The bias voltage was 3.8 V. A node voltage swing from 0.8 to 2.6 V was used as the input signal[20], resulting in an output amplitude of 1.2 V. If the bias voltage is increased to 4 V the current is reduced to approximately 1/10. 1 % settling times are 3.5 and 1.1 $\mu$s.

## 2.5 Digital Control and I/O

At the bottom of figure 2.20 the digital control section is shown, consisting of three distinct circuits: RAM, PLL and clock sequencer. The RAM and the PLL are full custom circuits whereas the clock sequencer logic has been written mostly in the hardware description language *Verilog* [CDS97-1]. The final circuit was synthesized by the *Synergy* logic synthesizer software [CDS97-2] using the AMS 0.6 $\mu$m standard cell library [AMS98-4] as target. The asynchronous part was directly entered as a schematic since it contains combinatorial feedback loops which can not be generated by Synergy.

[19]The Spectre circuit simulator from Cadence has been used for the transient and the AC analysis of the circuits used in the EDDA chip.

[20]Referring to eq.1.24 this is the node voltage limit for a $V_{\text{precharge}}$ range from 1.2 to 2 V and $V_{\text{in}}$ between 0.4 and 3.2 V.

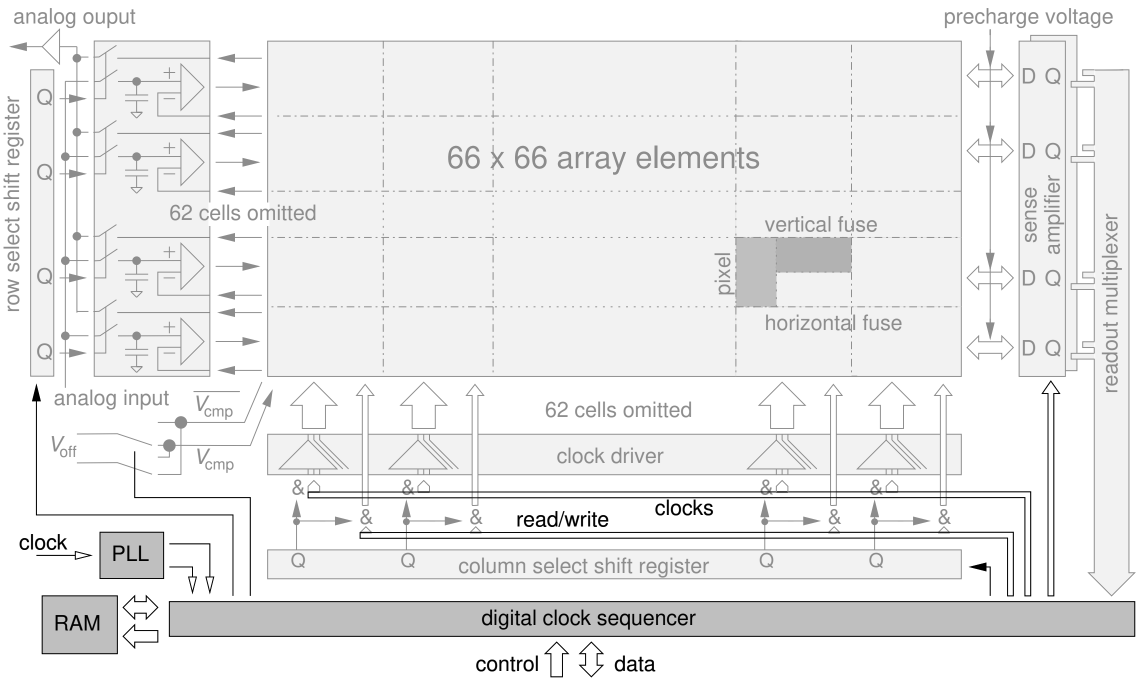



**Figure 2.20:** EDDA block diagram. The digital control part - consisting of clock sequencer, RAM and PLL - is highlighted.

### 2.5.1 Clock Sequencer

The main functionality of the clock sequencer is the generation of the 19 non-overlapping clock signals of EDDA. It also includes the logic for the eight bit bi-directional bus connecting the chip to external circuits. The timing diagrams in the previous sections (2.6, 2.7, 2.8, 2.14) use a timescale where every signal changes on a 10 ns boundary. This is an idealization. 10 ns per clock is the maximum speed the clock circuits have been designed for. If some of the process parameters are worse than their typical mean values it may be necessary to slow down certain clocks. Also it is possible that extending the duration of some signals leads to less noise and better analog accuracy. Furthermore it may be interesting to change the timing for experimental purposes, altering the characteristic length for example. For these reasons the sequence of the clock signals is not hardwired but freely programmable. A digital sequencer circuit executes a series of instructions stored in an on-chip memory. Those instructions are either load operations, transferring a clock pattern from the RAM to an output register, or flow control statements coding for conditional jumps. Appendix C describes the individual sequencer instructions and the internal registers of EDDA.

Since the EDDA chip has been designed for a maximum clock speed of 100 MHz the instruction decoder must be either simple or pipelined. A pipelined sequencer will add a penalty of one or more cycles - depending on the pipeline depth - to every jump. This could be reduced by a branch prediction logic but only at the expense of an increased complexity of the sequencer circuit. The additional power consumption of such a solution was the reason to implement the clock sequencer

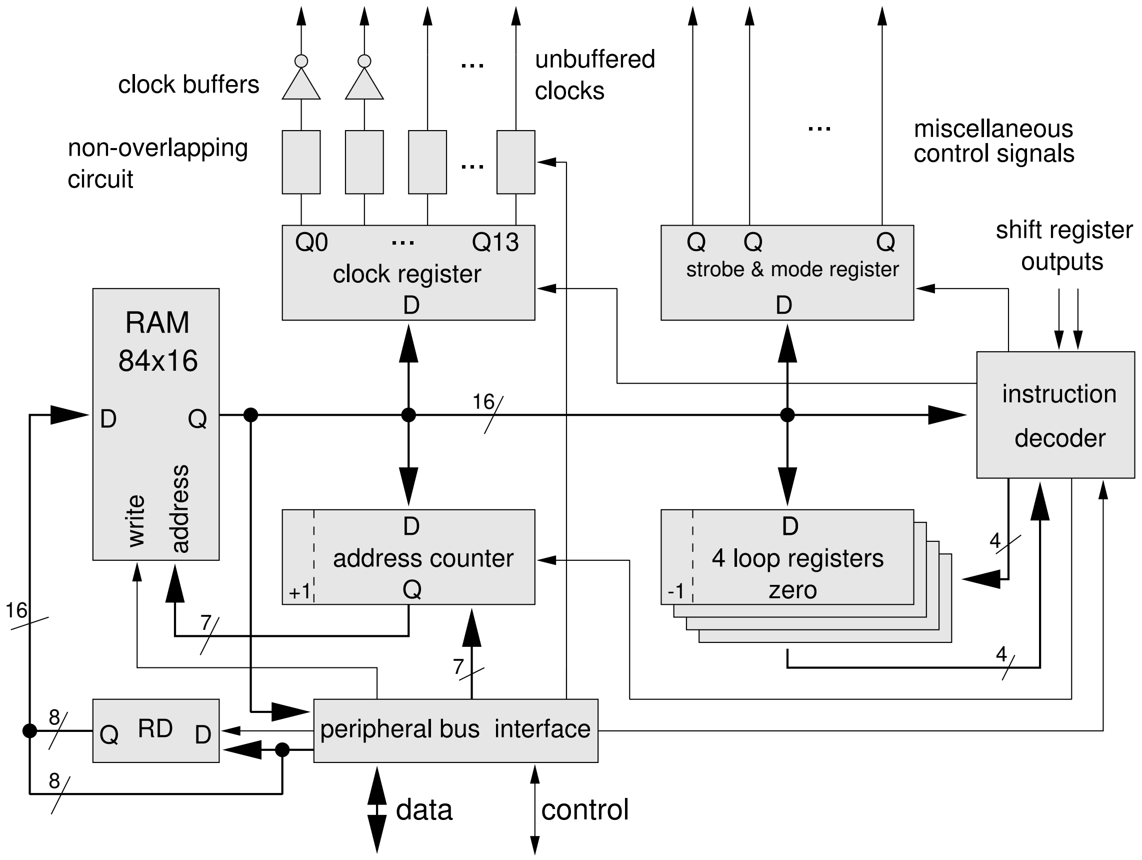



**Figure 2.21:** Block diagram of the digital control circuit. For the address and data busses the number of bits is drawn next to them.

without pipelining. The clock sequencer is subdivided into several functional blocks. They are depicted in figure 2.21.

The RAM is 84 words deep, with separate input and output ports. The width of a data word is 16 bit. It is optimized for speed. This is done by integrating an output driver in every bit cell. The address to output time is less than 2 ns. Its implementation is fully static, i.e. the power consumption will be zero (neglecting leakage currents) if there are no input signal transitions. The area occupied by one bit cell is 186 $\mu$m$^2$. Appendix A covers the implementation and the layout of the RAM in detail.

The RAM output is connected to the different registers of the sequencer as well as the instruction decoder. Each data word contains one instruction. Depending on the individual instruction, a part of the RAM word is used as immediate data for one of the registers. For each clock cycle the decoder decides which registers have to be updated.

The different sequencer registers have the following tasks:

**address counter** The address counter stores the RAM address of the actual instruction. In the case that a conditional jump is true it is loaded with the new address. If no jump has taken place the address counter increments the actual address to point to the next instruction.

**clock register** The clock register stores the actual state of the 14 independent non-overlapping clock signals. The remaining 5 clocks can be derived from them. At the output of this register they are not yet non-overlapping. This is ensured by an asynchronous circuit that delays the active edges[21] of the individual clocks for an adjustable time. The outputs of these delay circuits have three-times the standard drive-strength. The clocks signals for which this is not enough have additional buffers with 16 times drive capability inserted. These heavily loaded clocks are the global array clocks (see section 2.3.1) as well as the clocks for the large precharge switch (section 2.3.2).

**strobe register** This register stores output signals only for one clock cycle. This eliminates the need for a second instruction to reset them. Strobe signals are the shift register clock and reset signals for example.

**mode register** In the mode register miscellaneous control bits - like the global read and write signals - are stored.

**loop register** The four loop registers serve simultaneously the purpose of an ALU[22] and a general purpose register bank. Each can be loaded with an arbitrary 7 bit value. Their zero-output is used as a jump condition in certain instructions. Depending on the exact kind of instruction they are decremented either if the jump takes place or not. By this way up to four nested loops can be constructed. Another instruction that makes usage of a loop register is the delay command that loads one of the loop registers with a delay value and then inhibits the incrementation of the address register until the loop register value has reached zero.

The bus interface unit handles the peripheral bus protocol. It also contains some configuration registers that control the action of the clock sequencer and the delay circuit for the non-overlapping clocks. The RAM data (RD) register is used to store the first half of the 16 bit RAM data words as the peripheral bus is only 8 bits wide. The data written into the RAM appears immediately on the RAM output lines. Every sequencer register can be accessed by this way. Via the bus interface the address counter can be written directly. Another function of it is the selection of the digital readout cluster (see section 2.3.2). The eight bit digital readout data is routed through the interface to the peripheral bus. The digital readout can be synchronized to the sequencer by handshake signals.

### 2.5.2 Generation of the Non-Overlapping Clocks

In a complex and distributed SC circuit like the resistive fuse network there are more than two clocks that must not overlap each other. In EDDA 19 clock signals are concerned. The usual solution - to inhibit one clock signal until the other one is inactive [GEI90-8] - is impractical because some of the clocks have additional buffers distributed along the array. It would be necessary to fed their outputs back to the clock generator. Also it is difficult to determine the voltage level at which a clock signal can be considered inactive[23]. This depends on the circuits connected to it. For the NMOS pass transistors this voltage depends on their source potential. If the gate voltage becomes smaller than $V_{\mathrm{S}} - V_{\mathrm{T}}$ (neglecting the bulk effect) they cease to conduct.

[21] The active edge depends on the polarity of the clock signal. If it is active-high the rising edge will be the active one, for an active-low signal it will be the falling edge.

[22] Arithmetic Logical Unit

[23] The switched capacitor circuits controlled by the clock signals are level-sensitive, not edge-sensitive like the synchronous registers.

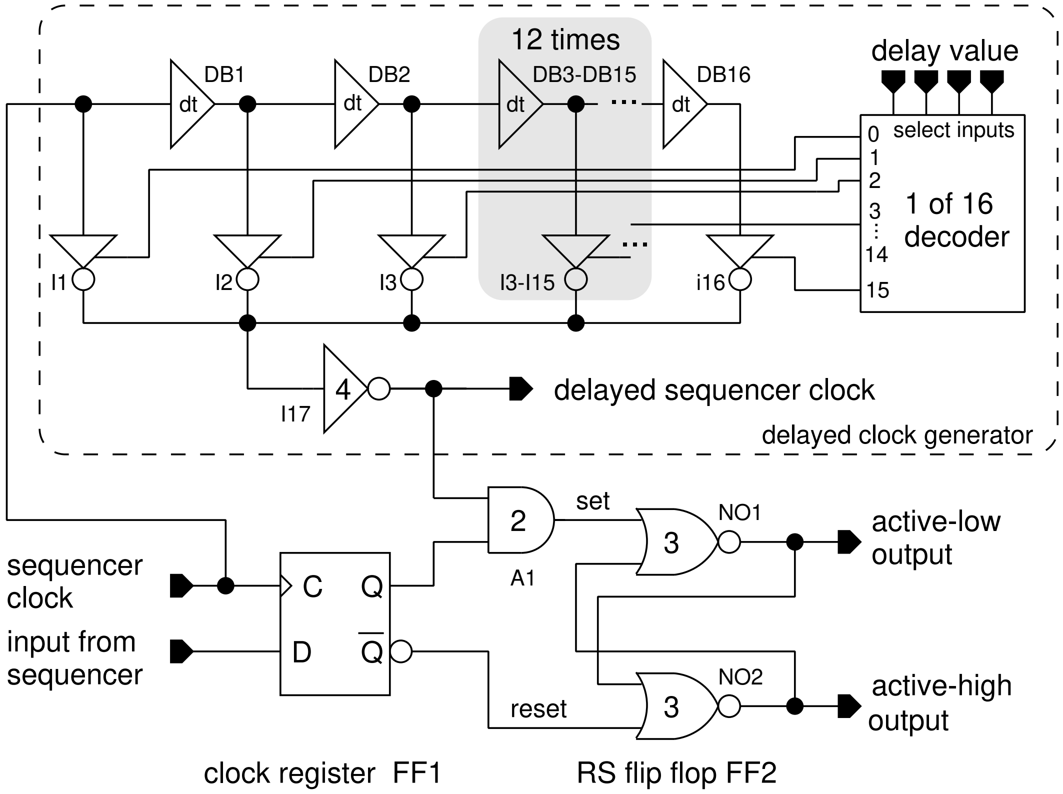



**Figure 2.22:** Schematic diagram of the circuit used to generate 19 non-overlapping clock signals. The numbers inside the components are the relative drive strength compared to the unit inverter of the AMS digital standard cell library.

Some of the 19 clocks are allowed to overlap with certain other ones, but not with all of them. Additionally there are clocks that must be active while others are changing. For example in column-by-column readout mode the gclk3read signal must not overlap gclk2 and gclk1 but should stay active while clk3 changes from low to high.

The circuit used in EDDA to generate the non-overlapping clocks therefore does not depend on the clock outputs. Instead it delays the active edge of any clock signal for a certain amount of time. This delay can be adjusted for every individual chip and sequencer program to minimize the time at which no clock is active.

Figure 2.22 shows the schematic diagram of this circuit. It is based on an asynchronous RS-flipflop FF2 in series with the output of the synchronous clock register FF1. The sequencer clock is delayed and inhibits the set-input of FF2. Thus an active (FF2 is set) clock output is not influenced by the delay any more. Also no delay is added to the reset-input of FF2. The delayed clock generation circuit is needed only once. It consists of a chain of slow buffers, DB1 to DB15. The slowest buffer from the standard cell library has a delay of 0.13 ns if it is connected in series. To avoid the high power consumption of multiple fast buffers a new standard cell has been designed: the delay buffer (DB). It consists of two inverters. The first one has its W/L reduced by a factor of 3 compared to the unit inverter. This results in a typical delay of 0.4 ns for the complete buffer (if it is part of a delay buffer chain). The length of the delay chain is selected by the tristate-inverters

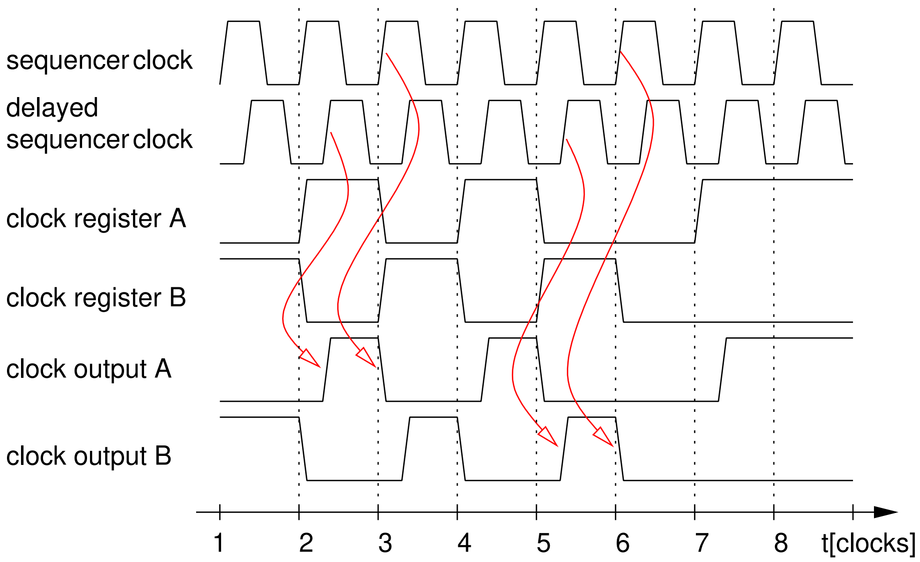



**Figure 2.23:** Timing of two example clock signals A and B. At the top the normal and the delayed sequencer clocks are shown. Below them the outputs of the synchronous clock register and the asynchronous RS-FF are depicted for both clocks. While the clock register outputs change simultaneously with the sequencer clock the final clock signals are non-overlapping.

I1 to I16. One of them is enabled by the 1-of-16 decoder. A delay value from 0 to 15 delay buffers can be chosen. I17 is an output buffer for the delayed sequencer clock signal. Figure 2.23 illustrates the timing of this circuit.

EDDA contains 15 of these gated RS-FFs. Each clock register bit uses one except of the $V_{\mathrm{off}}$ polarity bit that is connected to two RS-FFs. By this way the four control signals of the comparator voltage switch can be generated from one bit coding for the phase (see section 2.4.1). Since the individual clock outputs are complementary the same circuit can be used for active-high or active-low as well as complementary clock signals while the clock register bits are always active-high. By clearing the clock register all the clock signals become inactive. This is automatically done by the power-on reset to ensure that the analog circuits are inactive until a valid sequencer program is loaded and executed.

### 2.5.3 Main Clock Generation

The sequencer clock is generated by a PLL circuit. This makes it easy to connect to the peripheral bus since the bus frequency can be lower than the sequencer clock. The PLL multiplies the signal from the external clock input by a factor of 2, 3 or 4. It is also possible to bypass the PLL, using the external clock both as the sequencer and the bus clock. The PLL circuit is described in appendix B.

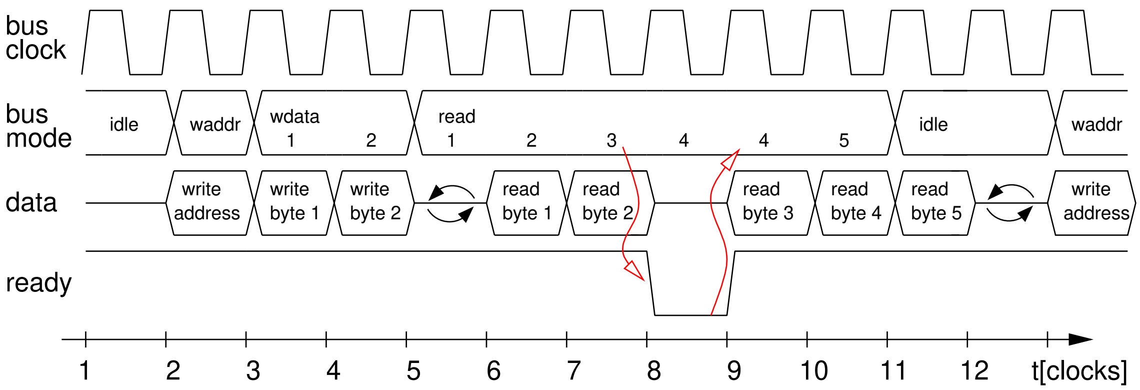



**Figure 2.24:** Timing of the peripheral bus of EDDA. The bus modes are abbreviated as follows: write address: *waddr*, write data: *wdata* and read data: *read*. The small arrows symbolize the changeover of the data drivers.

### 2.5.4 External Signals

**Peripheral bus**

The peripheral bus of EDDA is a bidirectional eight bit bus synchronous to the bus clock. An external bus master initiates read or write cycles. Write cycles are distinguished between address and data writes. If EDDA is not able to complete a data transfer in one cycle it will insert wait cycles. The maximum transmission speed is equal to the bus clock (in bytes/s). The bus uses the following signals:

**Bus mode** Two bits encode the bus mode. These are inputs of EDDA. They determine the kind of bus cycle:

**Idle** The idle mode signals that no new requests are initiated in this cycle.

**Write Address** Initiates an address write cycle.

**Write Data** Initiates a data write cycle.

**Read Data** Initiates a data read cycle.

**Ready** Handshake signal from EDDA. If it is high the actual cycle has been completed by EDDA.

**Data** Eight bidirectional data lines. If no read requests are pending the data bus is in a high-impedance state. It is driven by the bus master if the bus mode changes from idle or read to write. After EDDA has received a read command it starts driving the data bus.

Figure 2.24 illustrates the peripheral bus timing. During write cycles the data is transmitted together with the bus mode. EDDA can complete every write command without waits. If the bus mode codes for a read command the data gets valid after the next clock edge. If multiple bytes should be read the next read command must be issued by the bus master before the data of the first one was received. If the data is not ready EDDA can delay the read cycle by negating the ready signal. An eventually pending read command is executed after the delayed read cycle has been completed. In the example shown in the figure the third read command can not be completed without wait. Thus EDDA sets ready to zero after it has sampled the third read. The bus master issues the fourth read since it has not yet received the inactive ready signal. EDDA ignores this read command. The bus master does not increase its internal transfer counter until the ready line has returned to one. When it receives this condition it reads byte three from the data bus. Since EDDA has no further wait requests the bus master sets the bus mode to idle after it has issued read command five.

If the bus mode changes from write to read the data bus is idle for one cycle. This time is needed to change the direction of the data bus drivers. After the last read cycle the bus master must wait one clock before issuing a write command and enabling its data drivers.

#### Analog camera connection

If an analog camera is connected to EDDA the sequencer program that stores the analog input data must know the beginning of a frame, a line and a pixel. If the program knows the resolution of the image and the timing of the camera it will be adequate to signal the beginning of a new frame only. If the camera uses a time-base that is not synchronized with the bus clock of EDDA it is better to transmit a pixel clock together with the analog image. EDDA has two digital outputs and one input for handshaking with asynchronous image sources. If the image data is synchronous to the peripheral bus is is sufficient to start the sequencer program at the correct time via a normal bus write cycle.

The handshake input of EDDA is synchronized to the sequencer clock and can be included in the condition of a jump instruction. The outputs are part of the strobe register, thus a handshake pulse needs only one sequencer instruction. They can be individually changed to standard register behavior if a permanent output is desired.

## 2.6 Layout

The layout of the circuits described in the previous sections plays an important role for the performance of the EDDA chip. The area and speed optimization of the network cells makes them sensitive to parasitic capacitances. The capacitors used in the circuits are very small. The capacitive coupling between clock and signal wires can noticeably change the internal voltages. Of equal importance is the matching of the components both locally and globally. An example for components that should match locally are the sense amplifier inputs. Any matching error introduces a differential offset voltage. Transistor variations between different array elements may lead to non-uniformities in network parameters like $\lambda_{\mathrm{off}}$ for example.

### 2.6.1 Matching Considerations

There are several guidelines that help to reduce matching errors [AMS94, DRO99, LOV98, LAK94-2]:

**Orientation and surrounding** If two transistors should match the most important thing is to create them identically. The orientation of their channels and the direction of the currents flowing through them must be the same. Also their surrounding should be as similar as possible. This is most important for the drain and source regions, but every other difference in substrate doping in immediate vicinity may introduce deviations. If capacitive coupling is also considered the neighboring metal and poly traces should be created equally. Metal traces running across the gates may introduce an additional electrical field in the channel (depending on their potential). If it is not possible to avoid them at all they should be of the same shape and on the same potential for all devices. Also the highest possible layer should be used, i.e. the one most distant from the channel. In a repetitive structure like the resistive fuse network the individual elements are all copies of the same layout. Therefore only the cells at the edges have a different surrounding. By adding a frame of dummy cells to the array this can be solved. Another possibility is to ignore the outputs of the outermost cells, i.e. the array is designed larger by two in each dimension.

**Large devices** There is a $1/\sqrt{WL}$ dependency of the mismatch (either $V_{\mathrm{T}}$ or $\beta$[24]) caused by device geometry variations. For short channel devices (i.e. $L < 1\mu$) it is even worse, especially for $\beta$. Therefore short channel transistors should be avoided for critical analog circuits like the sense amplifiers or the readout source followers. The gate-capacitance also varies with the channel geometry introducing a charge injection error, i.e. the charge injection differs for equal devices. Increasing their size enlarges the absolute charge injection but reduces not only the relative but also the absolute difference. Since only the error caused by random geometry variations reduces by $1/\sqrt{WL}$ care must be taken to avoid other errors (see below).

**Minimum distance** The closer two components are to each other, the lower the effects of global process gradients and temperature variations.

**Same temperature** Matching transistors should be placed on the same isotherm of the chip to avoid temperature differences.

**Multiple equal devices** If transistors with different W/L ratios should match a unit device must be designed from which both matching devices can be built. To get a ratio of 7 to 2 for example 9 unit transistors are used.

**Common centroid layout** To match transistors with the same W/L ratio it is also useful to split them in multiple small ones. With more transistors a layout with a common centroid geometry is possible. Global errors are averaged out. This can vastly reduce the influence of process parameter gradients. It is important to remember that the configuration must also be symmetric with respect to the channel current directions. Either there is only one current direction for all transistors or - for an interdigit layout - the channel orientation is alternating, but with the same number of gates per direction for each transistor.

**Preferring NMOS-FETs** The main source of $V_{\mathrm{T}}$ errors are bulk doping level variations. Since the threshold voltage adjustment of PMOS transistors needs more ion-implantation steps, PMOS match worse than NMOS.

**Charge cancelation** The charge injection of switches can be reduced by adding a second clock signal with the opposite phase and connect it to the signal with the same capacitance. This can be done either by using equally sized NMOS and PMOS transistors - forming a transmission gate - or by adding a transistor of the same type as the switch transistor to the critical node. Usually this dummy transistor has half the width of the switch and both its drain and source are connected to the node.

[24] $\beta = K' \frac{W}{L} = \mu C_{\mathrm{Ox}} \frac{W}{L}$

### 2.6.2 Array Element Layout

Figure 2.25 shows the layout drawing of the array element. It is shown as a part of the complete network. The size of an array element is 57 × 48.5 $\mu$m$^2$. The clock signals are routed vertically and close to each other. This reduces the parasitic coupling between the clocks and the analog circuitry at the expense of an increased coupling between the clocks themselves. To the left of the clocks the fuse circuits are placed. Vertical and horizontal fuses are alternating. To avoid differences between them their layouts are as identical as possible. The only differences are the network connections, since the vertical fuse connects to the top and the horizontal to the right neighbor of the array element.

To the left of the clock column the pixel and node circuits are located. At the top the pixel source follower can be seen (M1/M2 in fig. 2.4). Below it the storage capacitor $C_\mathrm{s}$ is drawn. The storage switch (M3/M4 in fig. 2.4) is build of one transistor with split gates and a dummy transistor in the center to achieve at least a minimum of common centroid geometry in the limited area available. To avoid capacitive coupling from the nearby clock traces $C_\mathrm{s}$ is covered by metal1 connected to ground. The pixel capacitor $C_\mathrm{p}$, the node capacitor $C_\mathrm{n}$ and the fuse capacitors $C_\mathrm{a}$/$C_\mathrm{b}$ are located in the middle between the clock channel and the fuse circuits. They share an n-well together with the storage and readout circuits but isolated from the fuses. Since storage and readout are inactive in normal network operation this does not increase the noise level. The $V_\mathrm{DD}$ connection of the n-well has a low resistance (6 Ω from the middle of the array to the edge power busses) and low noise level since the fuses have separate power and ground connections. Only the quiescent current of the pixel source follower must be supplied by it.

Below $C_\mathrm{n}$ the readout source follower (M1/M2 in fig. 2.2) is located. To reduce matching errors the active transistor is relative large with a channel length of 2 $\mu$m to avoid short channel matching degradation. Its surrounding and orientation are the same for each array element. The transistors forming the $C_\mathrm{p}$ switch (M6/M7 in fig. 2.4) must have the same charge injection to avoid pixel voltage deviations between different cells. Therefore their split gate common centroid configuration. The small diffusion area protruding out of the bottom transistor is the counterpart of the diffusion trace connecting the top device to $C_\mathrm{p}$. The output resistance of the pixel source follower isolates the pixel $V_\mathrm{DD}$ and ground lines from the switching noise of M6 to M8 (fig. 2.4).

#### Fuse layout

In figure 2.26 the layout of a fuse is magnified. To the right the $C_\mathrm{a}$ and $C_\mathrm{b}$ capacitors can be seen. As mentioned before the $V_\mathrm{DD}$ and ground traces running above the capacitors on metal3 do not supply the fuse. Its power lines are shown in the middle (fuse ground and fuse $V_\mathrm{DD}$). The transistors that switch on the sense amplifier power (M5/ M6 in fig. 2.5) will inject charge onto them if clk2/$\overline{\text{clk2}}$ toggles. This will lead to a significant high frequency switching noise. Therefore the substrate is connected to the pixel ground instead of the fuse ground to avoid noise injection via the substrate contacts.

The central latch of the sense amplifier consists of two cross coupled inverters driving the SA and $\overline{\text{SA}}$ nodes. Since their matching is crucial for the EDDA performance they occupy a large portion of the fuses area. Their layout style is also common centroid with two gates per transistor. The PMOS-FETs (M1/M3 in fig. 2.5) are at the top of the dashed boxes, the NMOS (M2/M4) at the bottom. They are not covered by metal1 at all and only a fraction of them is covered by metal2. For area reasons it was not possible to avoid the metal3 power busses running across them. Since their potential is constant it is compensated by the charge that flows onto the gates in $\phi_1$.

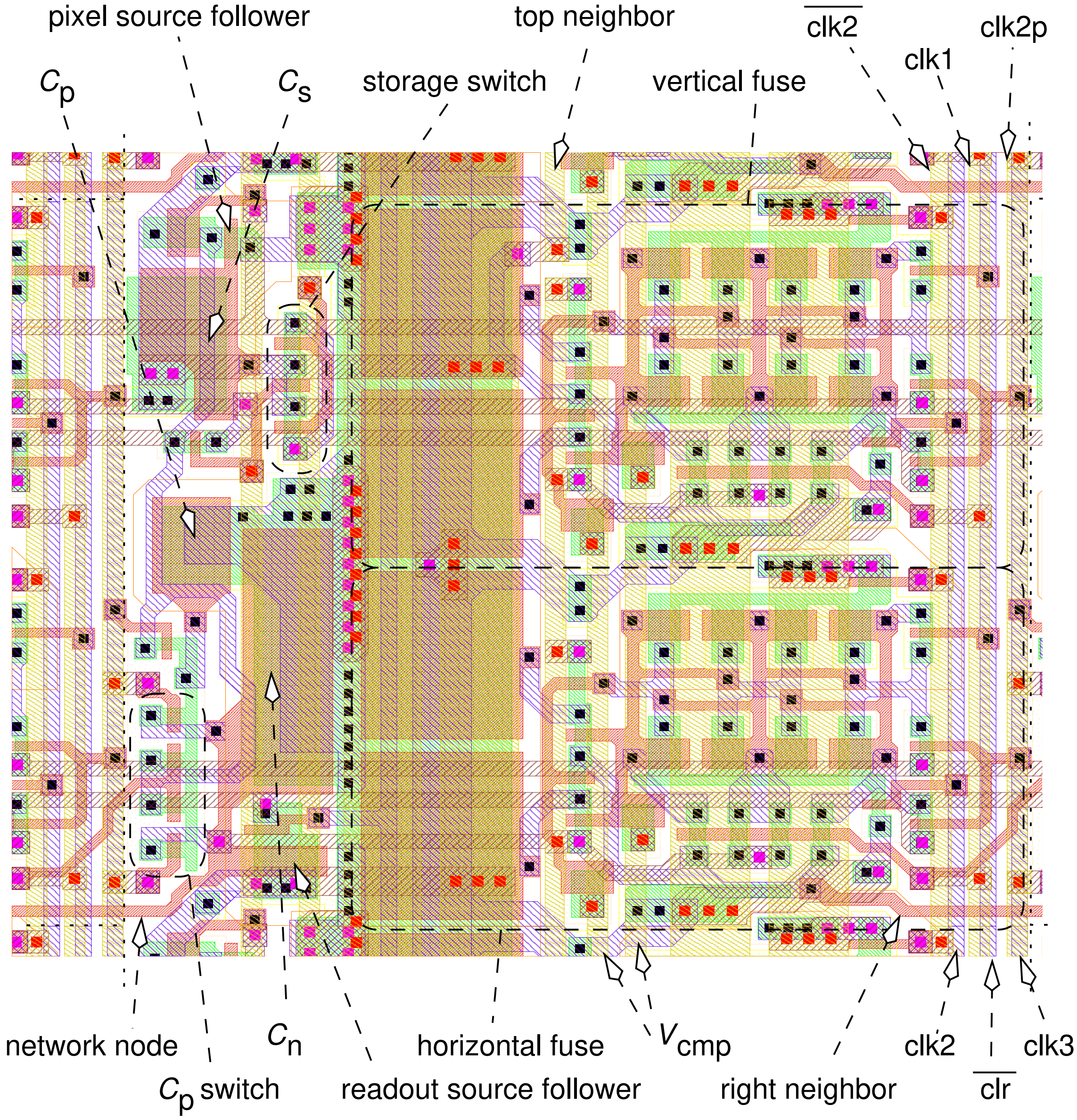



**Figure 2.25:** Cutout of the network layout showing one array element. The dotted lines to the left and the right depict the boundaries to the neighboring cells. A legend for the different layers can be found in appendix E. The visible area is 69 × 52 $\mu$m$^2$.

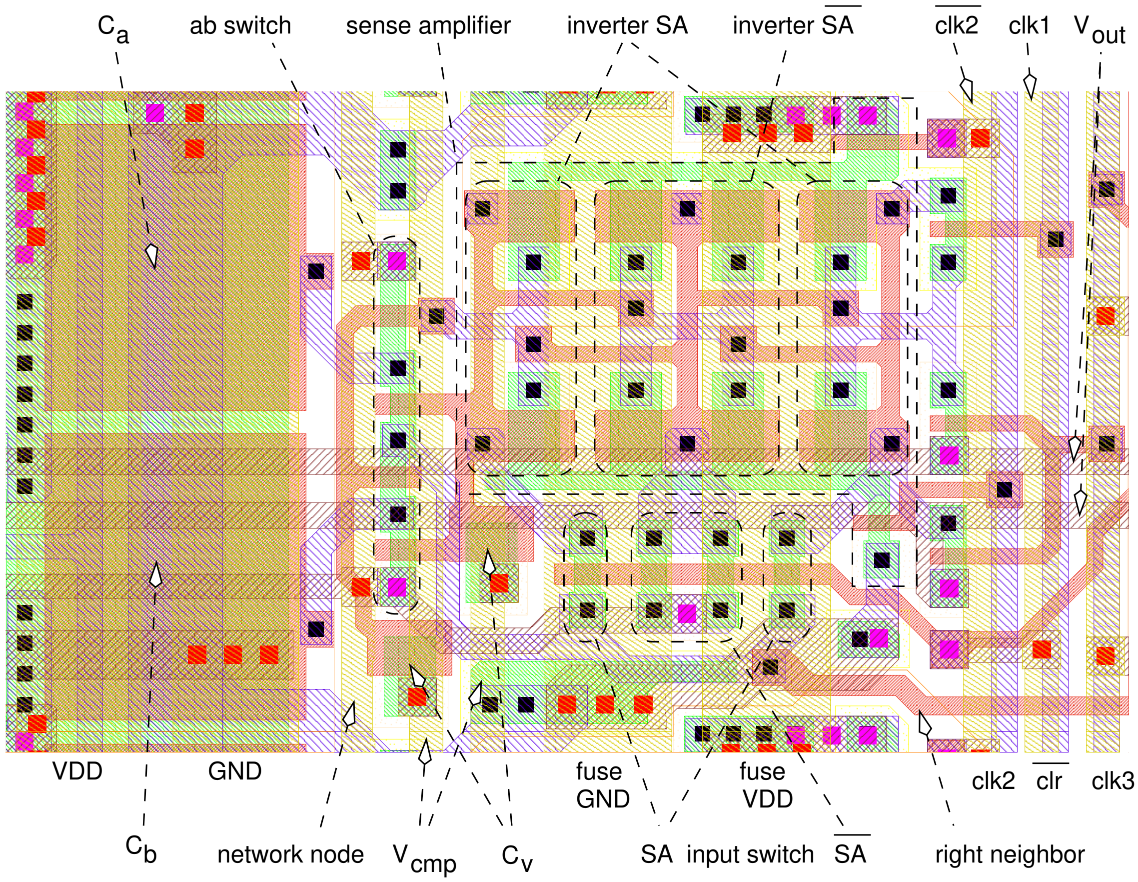



**Figure 2.26:** Layout drawing of the horizontal fuse. The visible area is $53 \times 25\ \mu\mathrm{m}^2$. For a layer legend see appendix E.

The input switches (M7/M8 in fig. 2.5) are directly below the sense amplifier. Their layout has the same symmetry. Since they are smaller they are equally covered by the metal3 power lines. Therefore any influence of them on the gate-channel capacitance leads to the same amount of charge injection at the SA and the $\overline{\mathrm{SA}}$ side. To the left of the sense amplifier the switches (M9 to M12) for the $C_\mathrm{a}$ and $C_\mathrm{b}$ capacitors can be seen. Since each fuse capacitor is much larger than the capacitance of the sense amplifier the voltage error due to the charge injection of M9 to M12 is accordingly smaller than that of the input switch. Each network node is surrounded by four fuses. Thus four transistors are connected in parallel. This reduces the charge injection error due to statistical geometry fluctuations by a factor of two.

### 2.6.3 EDDA Layout

The layout drawing of the complete EDDA chip is shown in figure 2.27. A photograph of it is depicted in figure 2.28. The largest part of the chip is occupied by the array of resistive fuse cells. It is enclosed by a ring of wide $V_\mathrm{DD}$ and ground traces, labeled as power busses in the figure. The substrate area beneath them is used for PMOS blocking capacitors. Their total capacitance is 1.5 nF. They act as a fast local charge reservoir for the high transient currents needed by the array in the moment when all the sense amplifiers are activated in $\phi_2$. Thereby the switching noise on the analog power supply lines gets reduced. The column clock drivers are located below the resistive fuse array in one row parallel to the array elements. By this way the transistors in each sense amplifier are always on the same isothermal line regarding the substrate temperature gradient caused by the clock drivers. They have their own power bus which includes also blocking capacitors with a total capacitance of 0.5 nF. In the case of the clock drivers these capacitors are essential. The maximum transient current needed to drive clk1 is 700 mA[25] with a slope of 30 A/$\mu$s in contrast to an average current of 24 mA in normal network operation at 100 MHz.

Between the clock driver power lines and the bond pads the different parts of the control logic can be seen: the PLL, the RAM and the standard cell block containing the clock sequencer and the bus interface. The remaining analog circuits are located on the right of the array: row readout and storage, the output amplifier and the $V_{\mathrm{cmp}}$ switch. The analog signals, including the analog power supply, use the bond pads to the left and the right, whereas the ones for the peripheral bus and the digital power supply reside at the bottom of the chip. To keep the substrate noise low the substrate contacts of the standard cells and the other digital circuits are not connected with ground but use a separate net for this purpose instead. It has a bond pad of its own.

[25] This value is based on a simulation including the total capacitive load of the clock line.

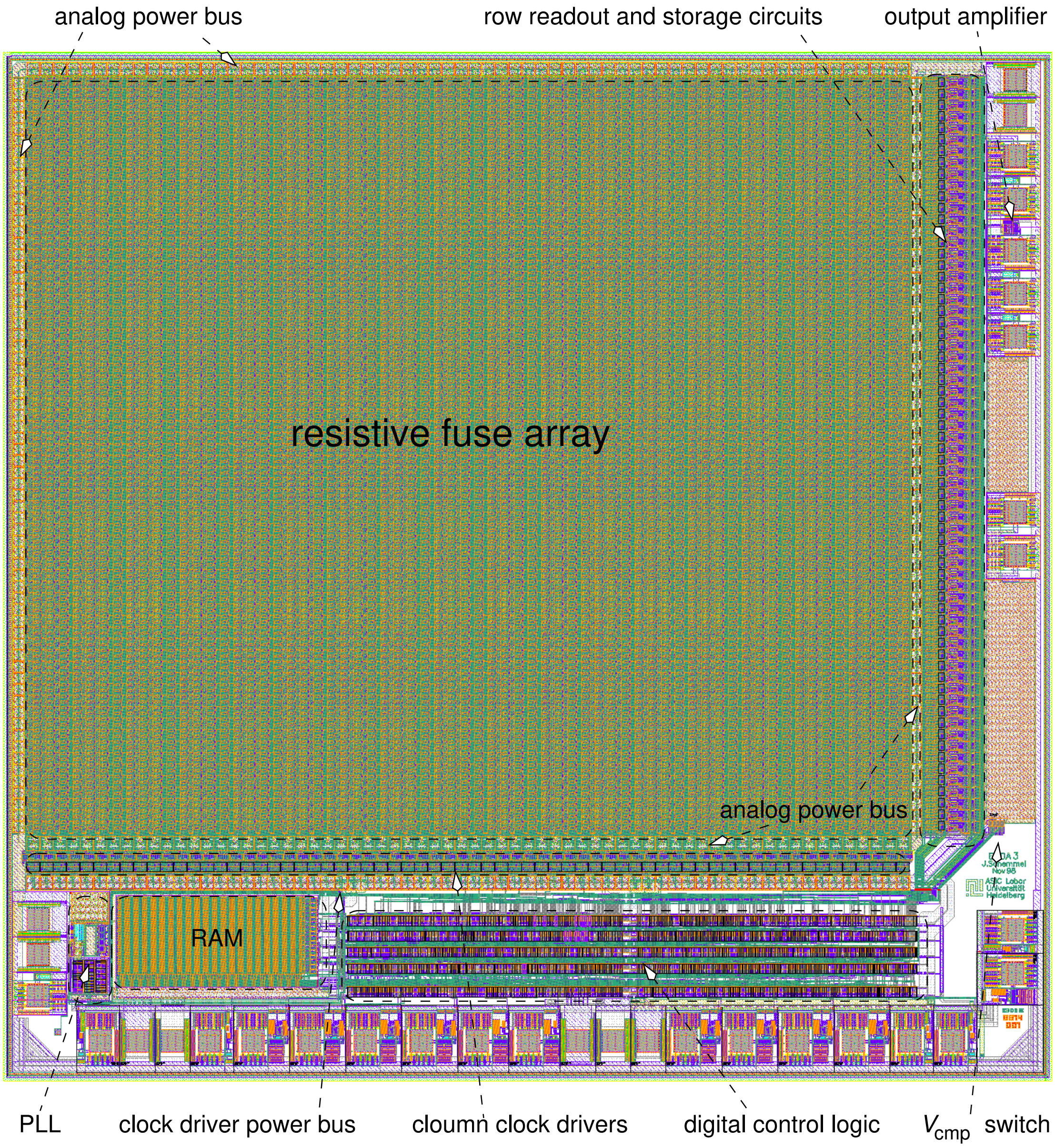



**Figure 2.27:** Layout drawing of the complete EDDA chip. The die size is $4.4 \times 4.3$ mm$^2$.

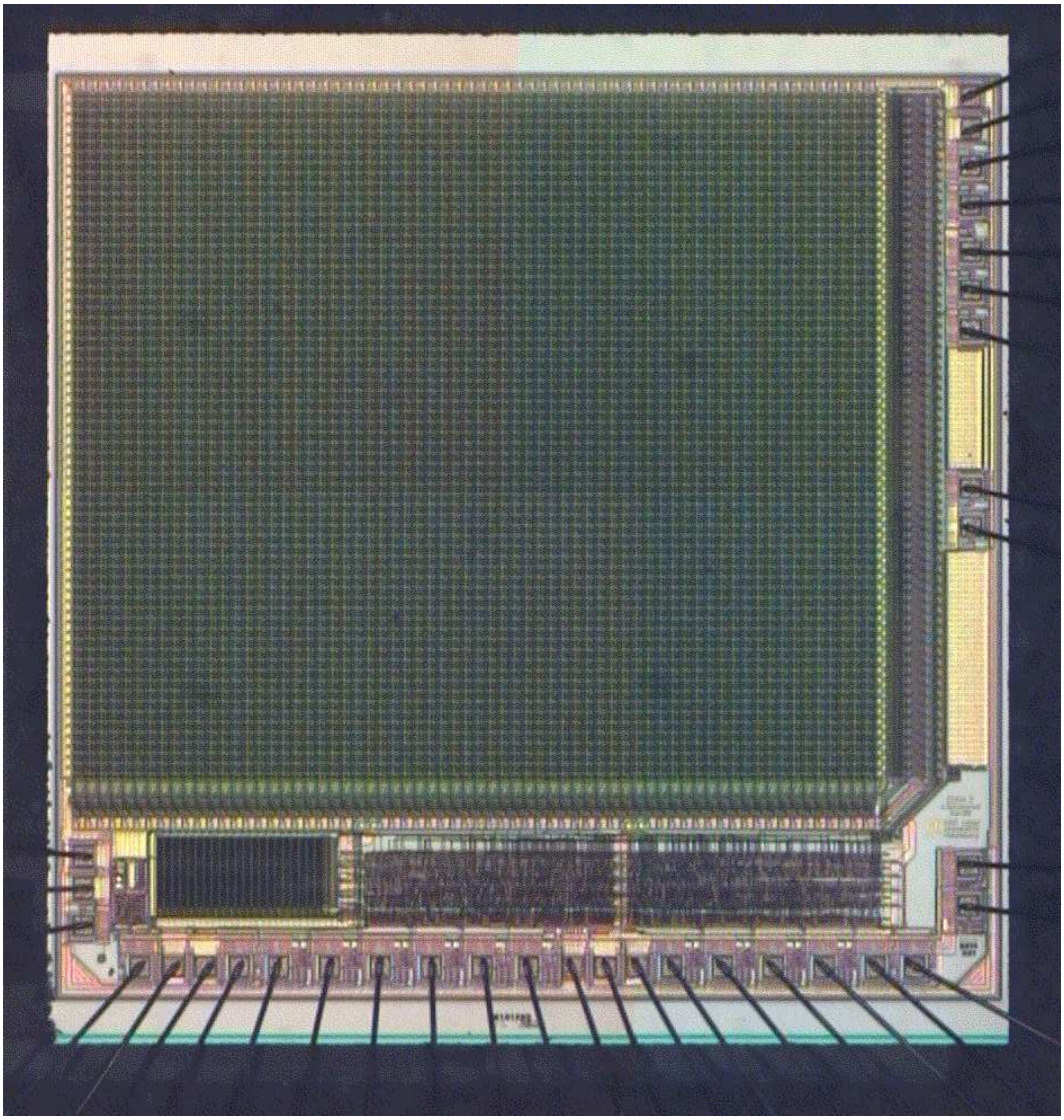


**Figure 2.28:** Die photograph of the EDDA chip. The bond wires connecting the chip to the package can be seen as black shadows (due to the illumination).

# Chapter 3

# Test Setup

*This chapter explains the test environment for the evaluation of the EDDA chip. At first it shows the hardware components used, including their mechanical setup. The programmable logic array that controls the EDDA chip is introduced afterwards. The different levels of software - ranging from the graphical user interface down to the hardware access - are described in the second part of this chapter.*

## 3.1 Hardware

The test setup for EDDA must be able to generate the necessary digital and analog input signal patterns and measure and store the output of the chip. Therefore a mixed signal test environment with the following capabilities is needed:

- Generation of the bus clock. The frequency should be adjustable to measure the speed limit and frequency dependence of EDDA.
- A digital I/O that can be connected to the external bus of EDDA. It must be able to supply the chip with the necessary sequencer programs. It should communicate with it to be aware of the state of the program. If an edge readout is part of the sequencer program it must read the edge data that EDDA transmits and store them for further evaluation.
- Analog outputs for the generation of the analog data input, bias and $V_{\mathrm{off}}$ voltages. The analog output must simulate a camera image. It should be synchronous to the digital I/O. It is also desirable to have an arbitrary $V_{\mathrm{off}}$ waveform. Thereby it is possible to measure the network's response to a changing $V_{\mathrm{off}}$ voltage. It might be that better edge detection results can be obtained by using a certain $V_{\mathrm{off}}$ waveform instead of a constant $V_{\mathrm{off}}$ level.
- An analog input to measure the node voltages. It is necessary to digitize and store those values to compare them with the edge data. This analog readout must also be synchronous to the digital I/O.

This is similar to the hardware needed to use the EDDA chip in an image processing environment. The main difference is that the analog input signal will not be simulated but produced by a real camera instead. Since the generation of a test image can be done by the same digital-to-analog converter (DAC) than the $V_{\mathrm{off}}$ waveform generation the analog part is nearly the same for the evaluation of the EDDA chip as for its usage in image processing applications. To be able to reuse the design of the digital hardware a concept based on industrial standards and commercially available components has been chosen: a Xilinx XC4028X [XIL97] field programmable gate array (FPGA) is connected via a PCI-bridge to the PCI[1] bus. EDDA and the A/D as well as the D/A converters are controlled by the Xilinx. This has the following advantages:

- The test logic can be entirely written in a hardware description language and is fully synchronous to the EDDA bus clock. Therefore it can be implemented in different FPGAs with minor modifications[2].
- The maximum bandwidth of PCI is 132 Mbyte/s (at a clock frequency of 33 MHz and a bus width of 32 bits). This is equal to 60k edge images per second and about three time the maximum data rate achievable with EDDA running at a bus clock of 40 MHz. Therefore the PCI bus will not be a bottleneck neither in testing nor in image processing as long as burst transfers[3] are used.
- As a widely accepted industrial standard PCI can be found in desktop computers as well as in miniaturized single board computers like those conforming to the PC104-plus standard. These PC[4] compatible systems measure less than $10 \times 10$ cm$^2$ and are well suited for portable image processing systems[5]
- The PCI-to-FPGA bridge[6] contains the necessary FIFOs[7] to use a clock that is asynchronous to the PCI clock without speed penalty. Frequencies up to 40 MHz are allowed at the FPGA side. Together with a programmable frequency synthesizer the EDDA bus clock can be freely adjusted.

### 3.1.1 Components of the Test System

Figure 3.1 shows the block diagram of the test system. At the top the host computer is drawn. For the measurements this has been a standard IBM compatible computer based on an Intel Pentium II CPU[8]. Since the rest of the test environment is connected to the host via the PCI bus a system with

[1] Personal Computer Interconnect. The specification of the PCI standard can be found in [PCI95]

[2] VHDL has been used in this case. It is a hardware description language comparable to Verilog-XL. The IEEE standard can be found at [VHD97].

[3] A PCI burst transfer reduces the overhead of bus arbitration and addressing. Multiple data cycles are performed after one address cycle.

[4] In this thesis the notion 'personal computer' is used for the descendants of the original IBM-AT computer. These include all the systems with an Intel x86 or equivalent microprocessor that are compatible to the Microsoft Windows or Linux operating systems.

[5] They also include the energy saving features of modern PCs, i.e. they can be put in suspend mode. The PCI bus is also power efficient since it does not use resistive termination.

[6] The part used is a PCI9080 from PLX Technologies.

[7] First In First Out

[8] Central Processing Unit. The exact configuration was a 233 MHz Intel Pentium II CPU on an Intel LX mainboard (66 MHz frontside bus) with 128 Mbytes SDRAM.

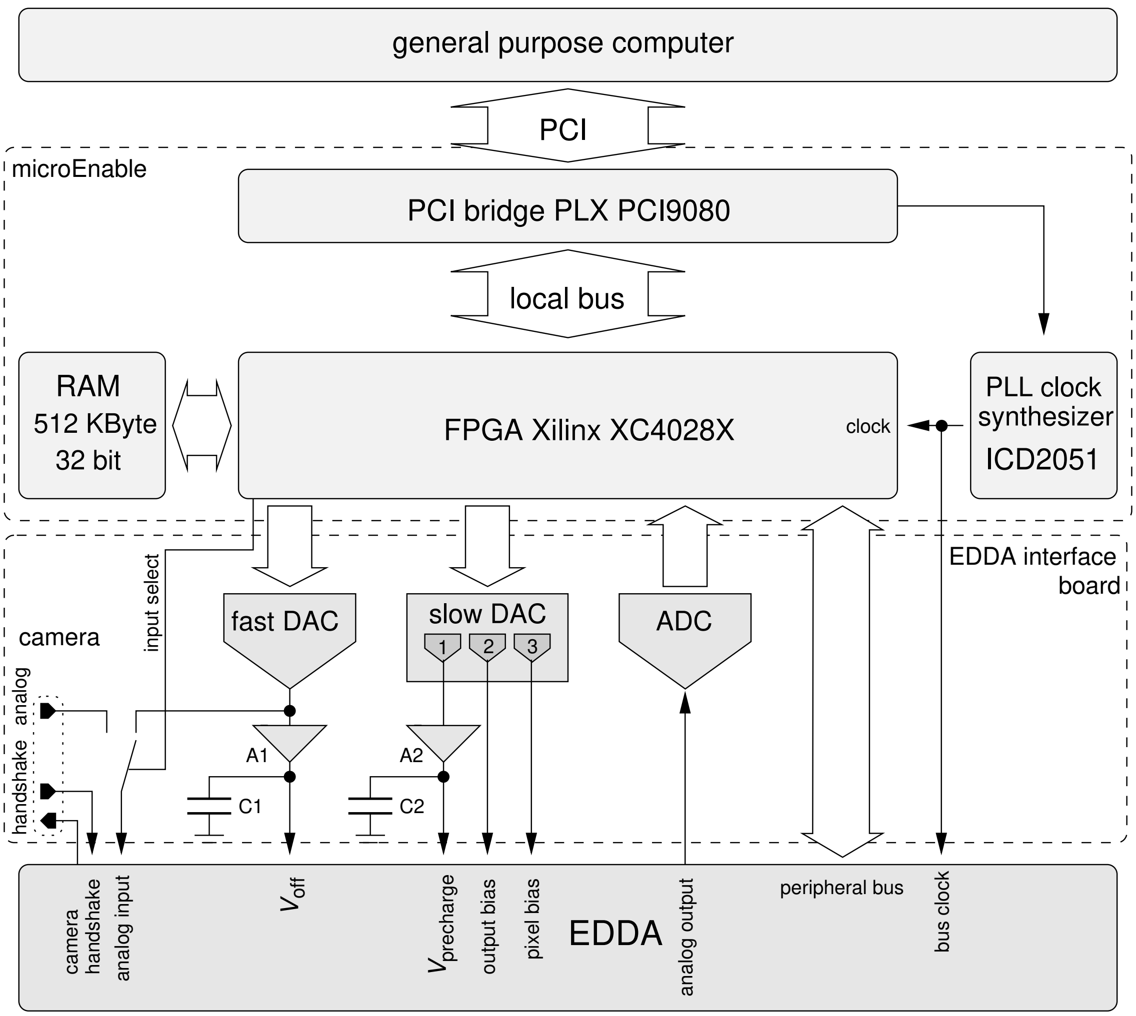



**Figure 3.1:** Block diagram of the test system used for the evaluation of the EDDA chip. The components enclosed by the upper dashed box are located on the microEnable card. The lower one contains the circuits that interface with EDDA's ports (mounted on the EDDA interface board). The signals of the camera connector are shown inside the dotted rectangle.

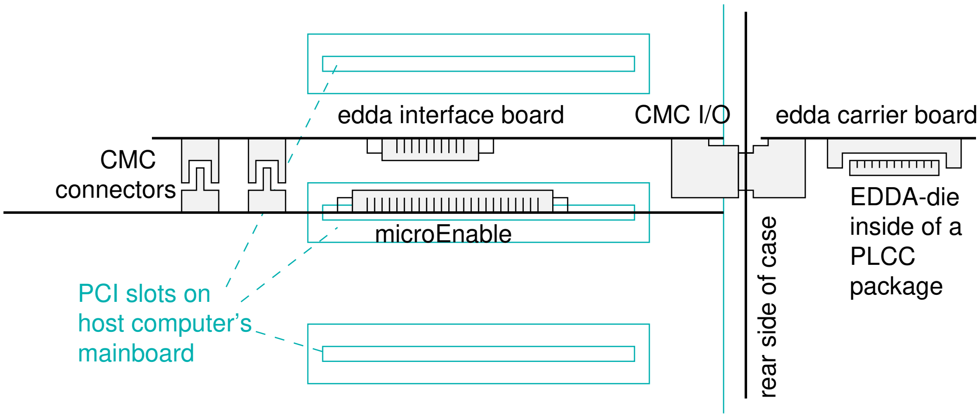



**Figure 3.2:** Mechanical setup of the different printed circuit boards used for the evaluation of EDDA (top view with respect to the host computer's mainboard). The EDDA chip is plugged into the EDDA carrier board which is connected to the EDDA interface board through the rear cover of the computer's case. The interface board is located on top of the microEnable via the CMC connectors. The microEnable itself is plugged into one of the mainboard's PCI slots.

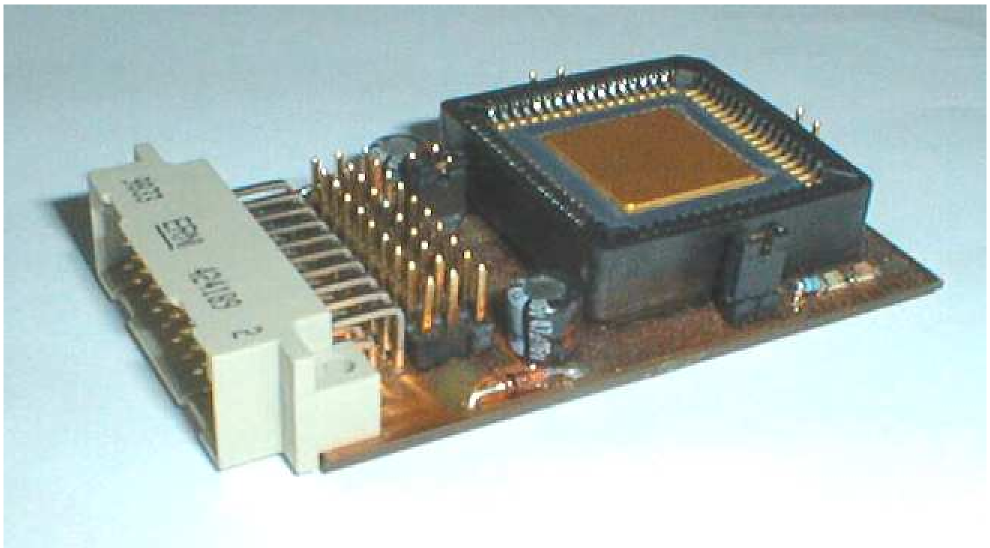


**Figure 3.3:** Photograph of the EDDA carrier board.

**Figure 3.4:** Photograph of the EDDA interface board on top of the microEnable card, which is plugged into the host computer's mainboard.

a different CPU could also be used. The PCI bridge, the Xilinx FPGA and the clock frequency synthesizer are located on a commercially available PC expansion card, called *microEnable* [MIC99]. It also includes 512 Kbytes of static RAM that can be accessed by the FPGA. The microEnable provides a slot for a CMC[9] daughterboard. It uses two CMC connectors that carry 64 general purpose I/O signals of the Xilinx. The analog components are located on the EDDA interface board. This board has been designed for the evaluation of the EDDA chip and fits into the CMC connectors. The CMC standard defines an I/O space on the CMC board. On the microEnable the CMC card is located in a way that this I/O region is accessible from the rear side of the computer case.

The EDDA-die itself is packaged in a 68 pin PLCC[10] at first which is then plugged into a socket on a small card. This EDDA carrier board can be connected to the EDDA interface card from the rear of the host computer. Thereby the EDDA chip can be exchanged without switching off the power of the computer or opening its case. Figure 3.2 illustrates this mechanical setup. Figure 3.3 shows a photo of the EDDA carrier board and the EDDA interface board on top of the microEnable card can be seen in figure 3.4.

The EDDA interface board contains the following analog circuits:

**Fast DAC** A 16 bit digital-to-analog converter capable of 30 MS/s (million samples per second) produces the analog input and the $V_{\text{off}}$ voltages. Its output voltage range is 0 to 5 V, the LSB[11] corresponds to only 80 $\mu$V. The $\pm V_{\text{off}}$ inputs of EDDA are connected asymmetrically: $-V_{\text{off}}$ to GND and $+V_{\text{off}}$ to $V_{\text{off}}$. The $V_{\text{off}}$ voltage is buffered by an operational amplifier A1 (see figure 3.1) that is capable of driving the capacitance C1. This capacitor is necessary to block the high transient currents when the comparator voltage switch changes the polarity of $V_{\text{cmp}}$. Its value depends on the amplifier used as A1 and the bandwidth of the desired $V_{\text{off}}$ signal. In the test setup it has been 2.2 nF. The analog input can be switched between the output of the fast DAC and the camera input.

[9] Common Mezzanine Card. A standard for small add-on cards [CMC95]. It covers the mechanical aspects and the power and clock signals.

[10] Plastic Leadless Chip Carrier. A chip package that can be either soldered directly onto a printed circuit board or combined with a socket.

[11] Least Significant Bit

**Slow DAC** This triple 8 bit digital-to-analog converter has an output voltage range of 0 to 5 V with a LSB of 20 mV. It is used to generate the analog input voltages that stay constant while an EDDA program is running: the pixel bias, the output amplifier bias and the precharge voltage. The latter is buffered by A2 and blocked by a capacitor C2 (2 nF) to stabilize $V_{\text{precharge}}$.

**ADC** The analog-to-digital converter samples the analog output voltage at a rate of 1 MHz with 8 bits of resolution. Its input voltage range can be adjusted to cover exactly EDDA's output voltage range.

**Analog power supply** Multiple integrated voltage regulators together with RC-filters provide the analog power supply for EDDA and the analog components on the interface board. This isolates the sensitive components from the digital switching noise. As an input they use the $\pm$ 12 V that are provided at the PCI connectors.

The digital bus of EDDA, the DACs and the ADC are connected to general purpose I/Os of the Xilinx FPGA via the CMC connectors of the microEnable board. The output of the frequency synthesizer chip is used as EDDA's bus clock and as FPGA clock.

### 3.1.2 Programmable Logic Array

The functionality of the FPGA is determined by the VHDL code used to program it. The source code must be converted into a netlist containing only library elements. This has been done with Synopsys, a commercially available logic synthesizer software [SYN98]. The binary data file - called the *bitstream* - necessary to configure the FPGA is generated from the netlist by the 'place-and-route' software from Xilinx (called M1). It is loaded via the PCI bus into the FPGA. Different bitstream files can be stored in the host computer and used alternately since the upload time is only 74 ms[12]. In an image processing system the FPGA can be used as a co-processor that does not only control the EDDA chip but does some post-processing of the edge data as well. For the evaluation of EDDA a VHDL code has been developed that contains the core functionality for the usage of the EDDA chip. It needs about 30 % of the configurable logic resources of the FPGA. Therefore multiple FPGA configurations with different data processing extensions of that core may be present in an image processing system.

The FPGA controls the complete test sequence. The RAM that is connected to it stores the input image, the $V_{\text{off}}$ waveform, the EDDA sequencer programs as well as the results from the readout of the digital edge data and the analog node voltages that have been digitized by the ADC. After the host computer has filled the RAM with the input data the complete image processing cycle runs automatically under the control of the FPGA. It reads the RAM data and transfer it to EDDA and the fast DAC. It also monitors the EDDA sequencer and stores EDDA's output. When it has finished the host can read back the results out of the RAM. By this way the timing of the host computer does not matter and there is no need for a real-time operating system.

The basic structure of the VHDL code is similar to that of the digital control of EDDA (see section 2.5.1): a sequencer reads instructions out of the RAM, decodes them and writes their immediate data to the appropriate output register. Additionally the FPGA sequencer can read data from EDDA and the ADC. A data word from the RAM is 32 bits wide. One word is used per

[12]This value was measured on the test system with the program 'config.c' from SiliconSoftware, the manufacturer of the microEnable.

instruction. Since both input ports are only 8 bits wide, 4 data bytes are packed together before they are written as one word into the RAM. To be able to access the RAM simultaneously from the PCI bridge and the sequencer's read and write channels an arbitration logic is included in the VHDL code that emulates a triple ported RAM. The read instructions can read up to 254 bytes at once, doing the handshake with EDDA automatically. For the $V_{\text{off}}$ waveform generation the instruction that writes a value to the fast DAC has a build-in delay capability. The number of EDDA bus clock cycles before the DAC value is written into the output register is also part of the instruction word. This reduces the number of instructions necessary for waveforms that do not need the full speed of the sequencer (it runs with the EDDA bus clock frequency).
The following FPGA sequencer programs are typically used, each in conjunction with the appropriate EDDA code:

1. Load a program into the EDDA RAM
2. Load a test picture or a camera image into EDDA
3. Generate a $V_{\text{off}}$ waveform while EDDA performs standard network cycles
4. Read out the edge information (usually with burst readout)
5. Read out the analog node voltages

They are executed in the same order as they appear in the list. At position two there are in fact two different programs. Which one is used depends on the kind of input desired. If the EDDA programs used fit together into the 84 words of RAM they can be loaded once at the beginning. The start address of the appropriate EDDA program must then be loaded into the address register of EDDA at the start of the FPGA program. If the EDDA programs do not fit they can be individually loaded by the FPGA sequencer before the actual program is started. The selected FPGA programs must be concatenated by jump instructions to execute multiple programs automatically. For debugging purposes every program can be started individually via PCI. It is also possible to communicate with the EDDA chip, the ADC and the DACs by ordinary PCI read or write cycles.

## 3.2 Software

### 3.2.1 Visor Program

The operating system used on the host computer was Windows NT 4.0 from Microsoft. The test software has been written in C++[13] and can be divided into two parts: a program for the evaluation of image processing algorithms - called the *Visor* program - and the EDDA module for it. Visor uses a graphical user interface written with the object windows library (OWL) from Inprise[14]. Its development had begun in 1996 (by the author of this thesis) but it has been strongly enhanced in the scope of this thesis. It was originally written for the simulation and test of a CMOS camera that used adaption in the time-domain [SCH97]. For this reason the Visor program is suited for analyzing image sequences. The concept of this software is a tree of C++ objects,

[13] For an excellent introduction to the programming language C++ see [CPP98]

[14] Inprise was formerly known as Borland and the OWL library was part of the Borland C++ integrated development environment. Both products are discontinued at the time of this writing.

called *filter modules*, that hierarchically process an input image sequence. The root of this tree gets periodically called to process the actual frame. Its output is used as input for the module in the next level of the tree. By this way different modules can be directly compared at every level of the image processing tree. The input image can come either from an external source - like a camera or via TCP/IP[15] - or it is simulated internally by moving a virtual camera across a bitmap. The image data are interpreted as grayscale images with 8 bit/pixel. The resolution of the filter modules is arbitrary[16] and can also be changed inside the tree by inserting an appropriate scaling module.
The filter module classes are derived from a common superclass that includes the standard functionality of every module:

- the graphical user interface including the automatic generation of parameter dialogs
- the parameter storage and retrieval
- the image data interface to the modules in higher and lower tree levels.
- the synchronized recording of the output of an arbitrary set of modules that can be stored in the file system and played back later

The derived subclasses need to supply only the image processing function itself. The simplest possible case would be a direct copy of the input image into the output image. Standard filter modules include linear and nonlinear spatial filters, a module for binarization and more complex edge enhancement filters like the Deriche [MEN99] as well as modules for the generation of histograms and cuts for example. The EDDA module is also a filter subclass, implementing the resistive network filter operation. Whether it uses the simulation (see section 1.4) or the real chip can be selected by the user.

Figure 3.5 shows an example screenshot of the Visor program. It illustrates the comparison of the outputs from the simulation and the real EDDA chip. The input image, depicted in the window titled 'SimVisor', was taken with the logarithmic camera (see introduction). Five windows represent filter modules: testpic, eddasim, eddachip, cut and histo. Each filter module can have one or more associated parameter selection windows. The window 'Edda parameters' belongs to the EDDA filter module 'eddachip'. The button labeled 'chip' in the first line selects the real EDDA chip for the image processing of the module. The parameter selection windows are constructed at runtime from descriptions contained in each filter module. They can be customized by the user.

The Visor program depends strongly on the OWL library which is only available for the Microsoft Windows operating system. Even though it would be possible to switch from OWL to a platform independent library like Troll Tech's QT for example it might not be desirable to use a graphical user interface at all with a portable image processing system. Therefore the complete functionality of the EDDA algorithm and the access to the hardware of the test setup and the EDDA chip itself have been hidden inside an own class hierarchy based only on standard C++[17] together with the C++ standard library [JOS96], especially the standard template library (STL). The EDDA module of the Visor program instantiates an EDDA object and provides the mapping of the elements of the graphical user interface onto the appropriate methods of this class.

[15]Transport Control Protocol/Internet Protocol. Lowest protocol levels of the Internet.

[16]Is is only limited by the resources of the computer and the fact that 16 bit integers are used for the x- and y-coordinates.

[17]The international standard for the C++ programming language is ISO/IEC 14882

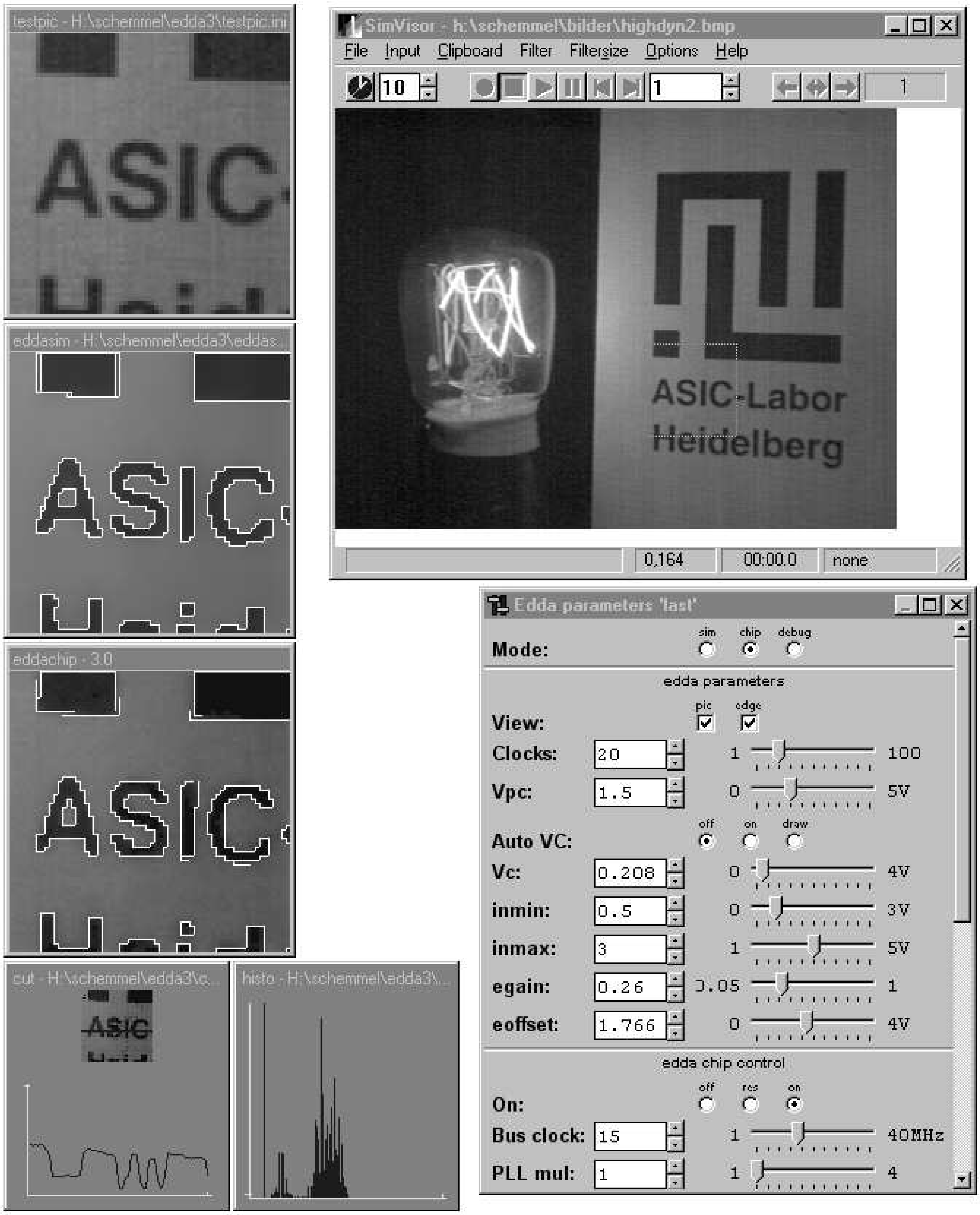



**Figure 3.5:** Example screenshot of the Visor program. The following windows are shown: 'SimVisor': Main window of the Visor program. It shows the actual input image and the position of the virtual camera. 'testpic': Shows the image taken with the virtual camera. 'eddasim': Edges and node voltages produced by the resistive fuse simulation. 'eddachip': The output of the real EDDA chip. 'Edda parameters': a parameter selection window belonging to the eddachip window. 'cut': Cut through the gray level image of the eddachip window. 'histo': histogram of the eddachip image.

| Member function | Description |
|---|---|
| Setpixel | Stores the input image of the network. Converts graylevels to voltages. |
| Getnodes | Reads the node voltages. Converts voltages to graylevels. |
| Run | Executes network cycles. One of its arguments is a reference to a vector[20]of floats, defining the $V_{\mathrm{off}}$ waveform. The number of elements in this vector determines the number of network cycles. The last cycle can be a burst edge readout. TEdda provides a virtual function to retrieve the storage address of the edge data afterwards. |

**Table 3.1:** Pure virtual member functions defined by the TEdda class that encapsulate the functionality of the resistive fuse network and must be defined in any derived class.

### 3.2.2 EDDA Classes

The base class of any EDDA class hierarchy is called *TEdda*[18] and provides the methods (i.e. the member functions) needed to use the SC resistive fuse network. It is an abstract class[19] that defines only the interface for the two classes that are derived from it:

**TEddachip** Uses the EDDA chip to perform the network operation. Therefore it only supports a resolution of 64 × 64 pixel. To avoid edge effects the outermost row of pixels is set to a common gray level (the chip has 66 × 66 network cells).

**TEddasim** Implements the simulation algorithm for the SC resistive fuse network and supports arbitrary resolutions. It also uses one extra row of network cells around the edges of the array.

Table 3.1 lists the virtual member functions of the TEdda class that encapsulate the functionality of the resistive fuse network. The conversion between graylevels and voltages is done automatically by the functions that store or retrieve source or node voltages, respectively. The function that sets the voltages for the minimum (black) and maximum (white) graylevel and the conversion functions itself are part of the TEdda class and not listed in the table (they are not virtual). TEdda contains also some functions for the error handling.

TEddasim and TEddachip use a completely different code to process an image. TEddasim stores the input voltages in an array variable and will use them as source voltages for the simulation if the Run-function is called. TEddachip creates an FPGA program to transfer the input image into the EDDA chip. If Run is called, this program is linked together with the programs for the $V_{\mathrm{off}}$ generation and - if selected - the edge and node readout. The FPGA sequencer is then started and the image is processed without any software intervention. The functions for the readout of the edges and node voltages return the values that have been already transferred out of EDDA by the FPGA and stored in the RAM on the microEnable board. The TEddachip class adds additional member functions to TEdda that are needed to set the hardware parameters and administer the

[18]The preceding capital 'T' is an abbreviation for *type*.

[19]A class with one ore more pure virtual functions is an abstract class. No objects can be created of an abstract class.

[20]The term 'vector' refers to the STL vector class.

different EDDA and FPGA programs. The EDDA filter module provides the link between these functions and the graphical parameter dialogs that are part of the Visor program. This separation between the graphical user interface and the image processing functionality of the different descendants of TEdda makes it possible to reuse nearly all of the source code that controls EDDA in a portable image processing system.

There is no software restriction concerning the number of EDDA modules. If the hardware is provided, i.e. if multiple microEnable boards are installed, more than one TEddachip class can be used. The number of TEddasim instantiations is not limited. Therefore the results of the simulation and the chip or multiple simulations with different parameters can be directly compared within the Visor software.

### 3.2.3 Hardware Encapsulation

The TEddachip class uses a custom class hierarchy to encapsulate the access to the different hardware modules used in the test setup. It has also been developed in this thesis and its use is not restricted to the control of the EDDA hardware setup only. The base class is called TAccess, since its descendants administer the access to the hardware. The idea is to instantiate one TAccess object for every part of the hardware that can store information or - to express it the other way round - whose actual state may depend on its history, i.e. what has been written to it before. The data flow between those parts is represented by read and write functions. Their internal organization is mapped onto an address space. The addresses are part of the read and write function calls. Since there may be functionalities that do not fit into the read and write framework, any class that inherits from TAccess can define additional functions to fulfill the needs of their associated hardware. The TAReg class, for example, adds functions to control a sequencer. Classes derived from TAReg, that represent hardware containing sequencer functionality like TAEddareg, redefine those functions by using the appropriate read and write calls to program the hardware.

The TAccess objects are organized as a tree. The TAccess object at the root of this tree represents the hardware module that acts as the interface between the computer and the other parts of the hardware. It contains links to all the TAccess objects that represent those components connected to the first module. In the case of the EDDA test setup the root of the TAccess tree is the class representing the microEnable board together with the FPGA. The second level is formed by the FPGA registers[21] and several objects representing parts of the RAM of the microEnable. Each FPGA sequencer program and the buffers for the readout of the edge data and the node voltages have their associated objects. Figure 3.6 shows the complete TAccess object tree as it is used by the TEddachip class to control the hardware of the EDDA test setup. The C++ class hierarchy of TAccess and its descendants is depicted in figure 3.7. It is also listed in table 3.2. Two distinct hierarchies are used: the C++ hierarchy and the linkage of the individual TAccess instantiations in the TAccess object tree. While the former is based on modeling from the general to the concrete (from am arbitrary RAM to the EDDA RAM for example) the latter is based on the electrical signal flow.

Every TAccess class knows the address mapping and the interface of the hardware it represents. For example the TAEdda class knows that EDDA contains registers that are 8 bits wide and a RAM that uses 16 data bits. A write access to the internal RAM is done via the external bus of EDDA. The first cycle transfers the 8 bit address, the second and third one half of the 16 bit data each.

[21]The FPGA uses several registers that are accessible via the PCI bridge to provide a programming interface for its internal functions.

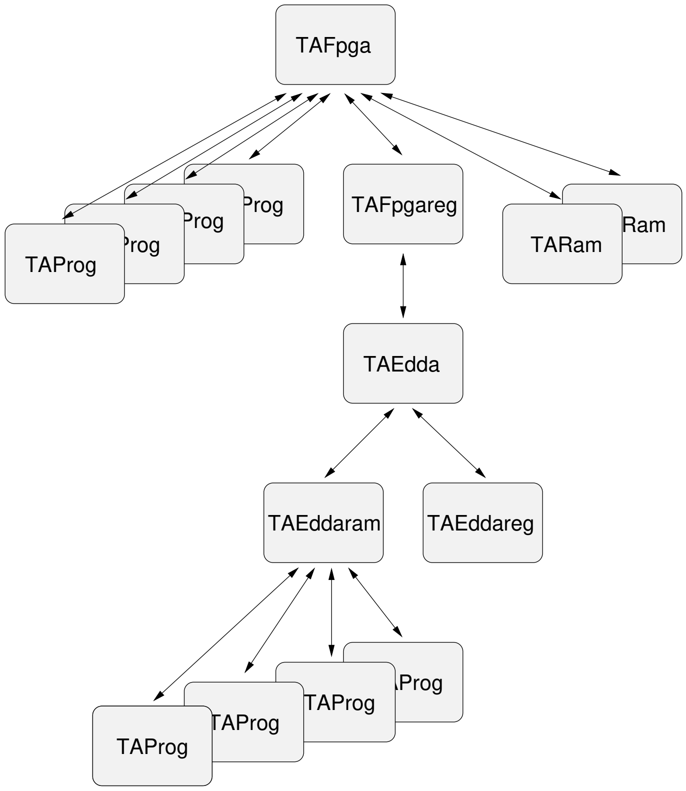



**Figure 3.6:** Communication structure of the TAccess objects encapsulating the hardware accesses in TEddachip. The number of TAProg instantiations depends on the number of EDDA or FPGA programs used, respectively.

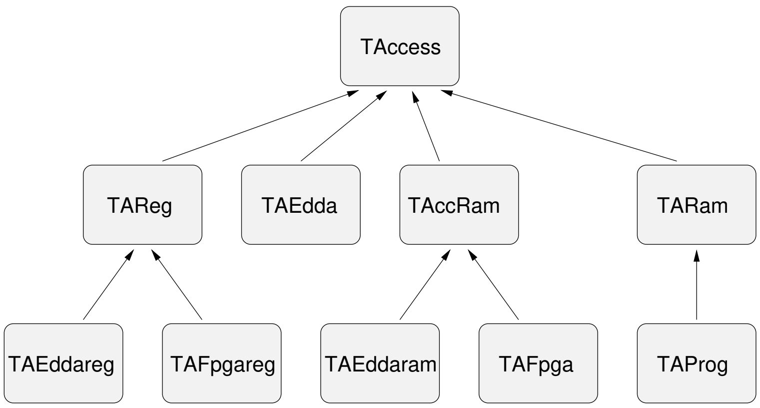



**Figure 3.7:** Graphical representation of the C++ class hierarchy of TAccess and its descendants.

| class name | base class | represented hardware object(s) |
|---|---|---|
| TAccess | none | Common base class. |
| TAEdda | TAccess | Provides the access to the internal registers and the RAM of EDDA via the FPGA-EDDA bus. |
| TAReg | TAccess | Abstract model of a configuration register space including a generalized sequencer control. |
| TAEddareg | TAReg | The EDDA register space. |
| TAFpgareg | TAReg | The FPGA register space, including the ADC and the DACs. |
| TAccRam | TAccess | Can represent an arbitrary RAM, includes tracking of memory usage by providing member functions to allocate and free memory regions. |
| TAEddaram | TAccRam | Represents the EDDA RAM. |
| TAFpga | TAccRam | Controls the configuration of the FPGA, the selection of the clock synthesizer frequency and the RAM on the microEnable board. |
| TARam | TAccess | An abstract representation of an allocated memory region. A TARam class needs a parent (in the TAccess tree) that is a descendant of TAccRam, since it dynamically allocates the memory region it represents from this class. |
| TAProg | TARam | Administers a sequencer program, i.e. stores the text representation in the computers main memory as well as the translated binary image in the EDDA- or FPGA-RAM. It includes the assembler for the sequencer programs and a mechanism to pass arguments from the C++ code to the programs. |

**Table 3.2:** List of the classes derived from TAccess that are used for the control of the EDDA test setup.

It also knows that EDDA stores the address, therefore it is not necessary to transfer the address twice if the same register is accessed repeatedly. To access the EDDA chip via a read/write call the TAEdda class must know a way to tell the class representing the FPGA side of the external bus of EDDA - TAFpgareg - to initiate those bus cycles. This is the only information that needs to be shared between two TAccess classes in the tree. For standard tasks this is done by the specification of addresses. TAFpgareg defines addresses that execute both an address cycle and a data cycle on the EDDA bus (if used in the read/write calls). In the same way the TEddaram class knows the base address of the EDDA RAM and translates a read/write to the RAM (with no base offset) to the correct address for the EDDA bus. It represents the RAM while TAEdda encapsulates the bus interface logic inside the EDDA chip (TAEddareg controls the sequencer and the configuration registers).

If one word is written to the EDDA RAM the following sequence of write calls will be performed ($x$ is the address inside the EDDA RAM and $y$ the data to be written). The word *call* above an arrow means that an object calls the write function of its parent in the object tree. Since TAccess is a polymorphic type, i.e. `Write()` and `Read()` are virtual functions, the calling class does not need to know the exact type of its parent:

| | | |
|---|---|---|
| TAEddaram.write($x,y$) | $\xrightarrow{\text{call}}$ | TAEdda.write($x' := x+$base address of EDDA RAM,$y$) |
| TAEdda.write($x',y$) | $\xrightarrow{\text{call}}$ | TAFpgareg.write($X_a$ :=EDDA address cycle,$x'$) |
| | | TAFpgareg.write($X_d$ :=EDDA data cycle,$ly$ :=lower byte of $y$) |
| | | TAFpgareg.write($X_d$ :=EDDA data cycle,$uy$ :=upper byte of $y$) |
| TAFpgareg.write($X_a,x'$) | $\xrightarrow{\text{call}}$ | TFpga.write($X'_a := X_a+$FPGA register base,$x'$) |
| TAFpgareg.write($X_d,ly$) | $\xrightarrow{\text{call}}$ | TFpga.write($X'_d := X_d+$FPGA register base,$ly$) |
| TAFpgareg.write($X_d,uy$) | $\xrightarrow{\text{call}}$ | TFpga.write($X'_d := X_d+$FPGA register base,$uy$) |
| TFpga.write($X'_a,x'$) | $\xrightarrow{\text{C stat.}}$ | `microEnablebase[`$X'_a$`]`$= x'$ |
| TFpga.write($X'_d,ly$) | $\xrightarrow{\text{C stat.}}$ | `microEnablebase[`$X'_a$`]`$= ly$ |
| TFpga.write($X'_d,uy$) | $\xrightarrow{\text{C stat.}}$ | `microEnablebase[`$X'_a$`]`$= uy$ |

The abbreviation '*C stat.*' above an arrow means *C language statement*. TFpga is the root of the TAccess tree. It is the only class accessing hardware directly. `MicroEnablebase` is the base address of the memory region the PCI bridge chip has been mapped to. It is defined as: `unsigned long *microEnablebase`. After initialization this chip translates PCI cycles to local bus cycles transparently for the software. Therefore it can be directly accessed by C language statements. To access the EDDA RAM via a TARam object, the address must be translated several times. In the above example the different steps are:

1. TAEddaram adds the base address of the EDDA RAM to the desired address: $x \to x'$.
2. TAEdda generates three EDDA bus cycles. The address becomes part of the data. $X_a$ and $X_d$ are the new addresses used, coding for the address and data cycles on the EDDA bus.
3. TAFpgareg adds the base address of the register space inside the FPGA to the actual address: $X \to X'$.
4. TAFpga does the real hardware access using the microEnable base address together with $X'$.

The read/write calls are executed in the opposite direction as the hardware data flow. This method uses a distinct class for every address or data translation necessary. The appropriate TAccess class is constructed by redefining the read and write functions doing this translation. Each part of the hardware has its corresponding class derived from TAccess, containing only the code needed to access this particular part. No information about the other hardware is needed except its own addresses as defined by its parent node in the TAccess object tree. The TAccess base class defines also the error handling using C++ exceptions and the management of flags encoding the hardware state, i.e. if it is initialized, if the data in the TAccess class is up to date with the contents of the hardware registers or not etc.

The big advantage of the local structure of the information flow is that parts of the hardware can be changed without having to modifiy more than the TAccess classes directly concerned with it. If the microEnable board is exchanged against another PCI interface only a few lines of source code must be reprogrammed in the TFpga class. The TAccess object tree is build at runtime by passing the parent address to the constructor of each TAccess class. Therefore an exchange of hardware components could be done while the program is running. This may be useful for example to exchange the type of camera used in an image processing system. Another possibility is to insert a TAccess class that logs the hardware accesses at every node in the tree.

The generation of the sequencer programs is another example for the usefulness of the modular concept. The FPGA sequencer can generate the EDDA address and data cycles automatically. If the TAProg class is used as parent of the Edda class instead of TFpgareg the read and write cycles will be recorded as a sequencer program that can be executed later. By this way it is possible to test an EDDA programming sequence first by doing each access with the CPU and transferring it later to the FPGA sequencer without having to write any new code. The TAProg class is an example for the dynamic hardware representation possible with the TAccess classes. The TARam class (the base of TAProg in the class hierarchy) represents a part of the RAM of its parent in the tree. If the source code of the sequencer program changes TAProg will translate it again. If its size has changed the old memory will be freed and a new block will be allocated. Thus the RAM region represented by this TAProg instantiation has changed.

The obvious disadvantage of the TAccess concept is that multiple calls of virtual functions are necessary for each read/write transfer. On fast systems (like the Pentium II used in the EDDA evaluation) this is no major concern if only CPU speed is considered. But the structure of modern bus architectures like PCI makes it necessary to perform hardware transfers with larger blocks of data instead of single words to exploit their potential performance. Therefore TAccess defines also block read and write functions. In the base class these are mapped onto loops of single word read/write calls. Every class that needs fast data transfer can redefine these block transfer functions (TFpga for example to allow fast RAM access) but it is not necessary to do this work for TAccess descendants that are only accessed for configuration or initialization (e.g. TFpgareg, since every fast EDDA access is done by the sequencer, which programs and data are accessed via TFpga). The user of a TAccess class does not need to know whether the block functions are redefined by this particular class because the implementation from the base class will be automatically used otherwise.

# Chapter 4

# Measurements

---

*The evaluation of the manufactured EDDA chips is described in this chapter. The results from the measurements of the dynamic range and the temporal and fixed pattern noise of the individual parts of the array elements (e.g. the pixel or the sense amplifier) are reported in the first part. The second part studies the behavior of the resistive fuse network as a single entity using test images as well as real world photographs. The dynamic range, the fixed pattern noise, the characteristic length and the power consumption of the network are evaluated.*

---

## 4.1 Initial Tests

The EDDA chips have been delivered by the manufacturer AMS as unpackaged dies. Therefore the first step - after an optical inspection - was to package them. 68 pin ceramic PLCC packages have been used. The pad for the substrate contacts is bonded directly into the cavity which is connected to the ground of the EDDA carrier board by multiple bond wires and package contacts. After closing the lid of the package the first test was to measure the resistance between the $V_{\mathrm{DD}}$ and ground pins of the analog and the digital supply inputs. If no short-circuit was found EDDA was plugged into its socket on the carrier board which subsequently was connected to the interface board.

All of the measurements described in this chapter have been made with the TEddachip filter module of the Visor software. The first test performed by it is a RAM-test. It is automatically executed every time the EDDA chip is initialized. Therefore the correct function of EDDAs digital bus and part of its internal digital logic is ensured. The test uses a random pattern generated by the pseudo random number generator of the C standard library (`rand()`) to fill the RAM. To make sure that every test uses a new pattern the initial seed value for the `rand()` function is derived from the system time.

The chip had to pass a functional test with the sequencer programs used for the resistive network operation. It was checked by using some test pictures and qualitatively evaluating the edge and node voltage images. Only fully functional chips have been selected for further quantitative measurements. In the following two sections one of these chips has been used. The last section of this chapter shows comparisons between different chips. The outermost pixel row of EDDA is not included in the measurements, i.e. only the inner square of $64 \times 64$ pixels is considered as active area.

It was not possible to fully evaluate the high frequency behavior of the chips because of two problems that showed up:

- The FPGA was not able to run the necessary VHDL program faster than at 22 MHz. Since the maximum PLL multiplier is four, the highest frequency used for the evaluation of the network was therefore 88 MHz. The PLL itself was able to generate 160 MHz with each of the three chips tested. This was measured by setting the FPGA frequency to 40 MHz after initializing the EDDA chip. The maximum sequencer clock speed was tested similarly. A test program that toggles a camera handshake output was loaded and afterwards the clock frequency was increased until the output signal vanishes. At 100 MHz each of the three test chips still worked. The limit was about 105 to 110 MHz.

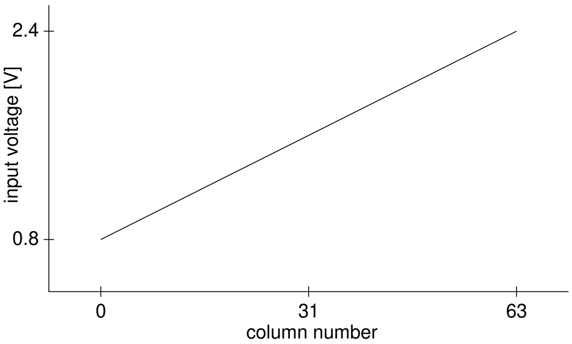
- The phase of the PLL output signal started to get instable if data was read from the digital bus of EDDA. This manifests in setup and hold time violations leading to read errors. The highest frequency where edge images could be read back has been 60 MHz. This was the frequency used for the real image tests at the end of this chapter. To avoid errors in quantitative measurements the PLL was disabled by setting the multiplier to one which bypasses the PLL. The reason for this instability is a design error. The digital power supply of the PLL is connected to the same bond pads as the power of the digital output buffers. The phase detector's power gets disturbed every time the output buffers draw current to output some value.

## 4.2 Performance of the Array Elements

### 4.2.1 Large Signal Behavior of the Pixel Source Followers and the Output Buffer

The first measurement serves two purposes: first to check the linearity of the node readout source followers across their dynamic range and second to measure the gain at a given bias voltage to be able to calculate the internal node voltages from the analog output values. For this purpose a ramp pattern is stored in the chip. Figure 4.1 shows its profile and summarizes the parameters used. The voltage range of 0.8 to 2.4 is larger than the node voltage range in normal operation. Although the pixel source follower output voltage range is about 0.4 to 3.2 V the influence of the precharged sense amplifier reduces the node voltage range (see eq. 1.24) to less than that used in this measurement. All pixels in a column have the same input voltage. Therefore non-uniformities of the storage amplifiers and the analog readout switches located in the row circuits are averaged. Their mean value influences every column in the same way. If the row circuits introduce any nonlinearities into the characteristic curves of the readout and pixel source followers this will be measured, but not the fixed pattern noise of these circuits.

To be able to read out the pixel source followers directly by the node readout source followers a sequencer program is needed that enables only clk3 and clk2p. This ensures that the node is isolated from their neighbors (since clk2/$\overline{\text{clk2}}$ are inactive) and from the sense amplifiers (clk1 inactive) but connected to the pixel source follower. The test program keeps clk3 and clk2p high while reading out all the pixels. To suppress temporal noise the sequence of storing the ramp and reading out the node voltages is repeated 200 times.




| parameter | value |
|---|---|
| readout bias | 3.98 V |
| pixel bias | 1 V |
| input range | 0.8 - 2.4 V |
| clock pattern | both clk3 and clk2p |
| input data rate | 2 MHz |
| readout rate | 0.5 MHz |

**Figure 4.1:** Input voltage ramp (left) and parameters (right) used to measure the linearity of the source followers.

This measurement results in $4096 \times 200$ data points. The mean values are calculated for each column and a linear regression is carried out with the remaining 64 values. The correlation coefficient is 0.99997 and the root mean square error 2.65 mV. The resulting gain of the readout source follower and the analog output buffer is 0.70. The readout bias voltage is kept constant in the following measurements. Any analog output voltage can be converted into their according node voltage:

$$V_{\text{node}} = 1.43 \cdot V_{\text{output}} - 1.358\text{V} \tag{4.1}$$

### 4.2.2 Noise of the Analog Storage and Readout Circuits

There are two kinds of noise that are examined in this section: spatial (fixed pattern) and temporal. The configuration with clk3 and clk2p activated during the readout is the one with the lowest number of noise sources involved. The array is not clocked, thus the substrate noise level should be low. Sources for fixed pattern noise are the row storage amplifier, the pixel storage switches, the readout source followers and the row and pixel readout switches. The sense amplifiers and the network resistors are completely disabled. The major sources for temporal noise are not located in the EDDA chip itself but in the analog test setup. Digital noise from the host computer couples into the analog input and output connections and the power supply lines.

The input voltage was set to 1.6 V and all the pixels were stored and read back 200 times (with 10 frames/second). To determine the temporal noise the standard deviation $\sigma$ of each pixel's 200 data points has been calculated. The mean value of the 4096 $\sigma$ values is 0.76 mV. The width of the $\sigma$ distribution 55 $\mu$V. For the fixed pattern noise the temporal mean values of the pixels have been computed. A histogram of the deviation of these values from their common mean is shown in figure 4.2. The resulting $\sigma$ is 5.44 mV.

The noise of the ADC was measured with EDDA switched off. For this purpose the clock signal for the EDDA and the outputs of the FPGA that control the digital bus are put into a high ohmic state. All the internal clock signals are disabled. The full scale range of the ADC was 1 V. The EDDA output was preamplified by a factor of 10. Therefore the LSB of the ADC corresponded to 391 $\mu$V. This was the configuration for all the noise measurements in this chapter. Measurements across the full dynamic range of EDDA were carried out without the preamplifier. 200 frames were recorded. The output voltage of EDDA's analog output buffer in this case was 2.9 V and the mean

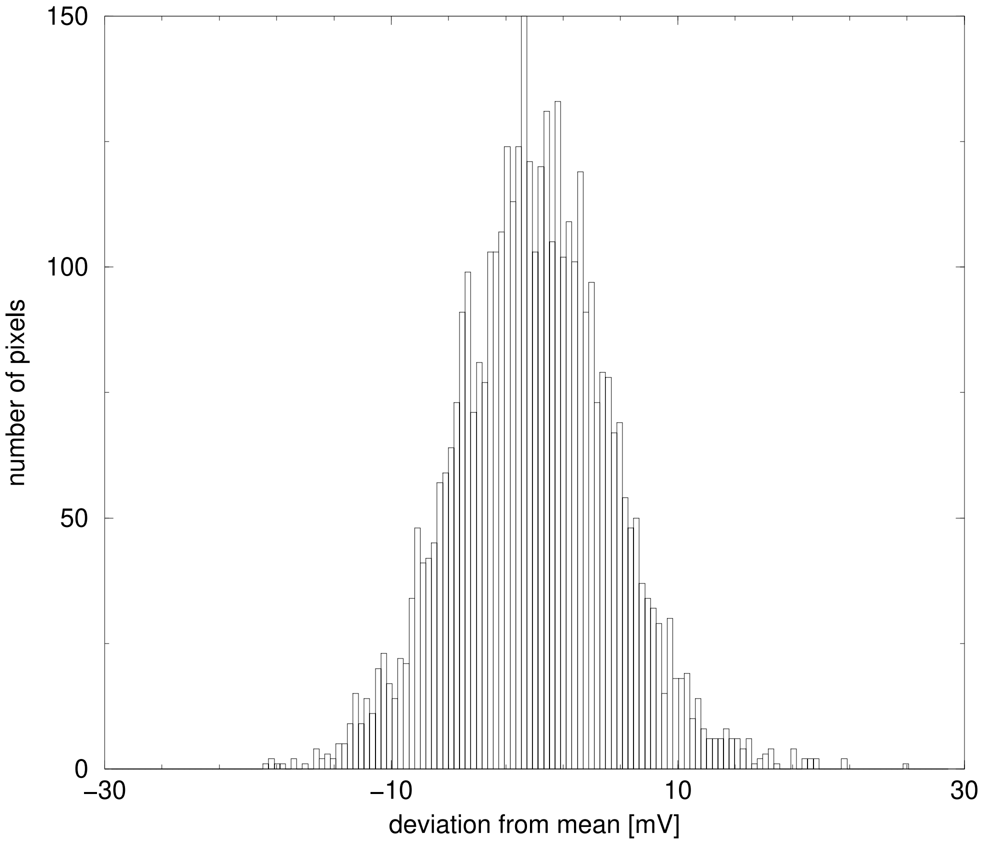



**Figure 4.2:** Histogram of the deviations of the readout voltage from its mean value for the 4096 pixels without any clock signals applied to the network. The root mean square is 5.44 mV. The input voltage was 1.6 V.

value of the ADC was 109.23 counts. The standard deviation of the $200 \times 4096$ data points is 1.022 LSB or 401 $\mu$V.

### Fixed pattern noise

To analyze how the fixed pattern noise is composed, the row and column mean values have been calculated. They are plotted in figure 4.3. It can be seen that the row variations are larger than the column ones. A global drift is superimposed on the local variations[1]. To compensate this drift a linear regression has been carried out for both curves. The resulting standard deviation is 2.38 mV for the rows versus a column value of 1.23 mV. The standard deviation of the row mean values can be interpreted as being composed of the pixel errors and the fixed pattern noise of the row circuits. Since there are no analog circuits that can introduce a significant column-by-column fixed pattern

[1] The fixed pattern noise is composed of Gaussian distributed variations, caused by the sum of the local inaccuracies of the different process steps, and spatially correlated deviations due to the imperfect control of the process parameters on the wafer scale. The latter are less interfering in the EDDA chip since the resistive fuse network works on the base of local differences.

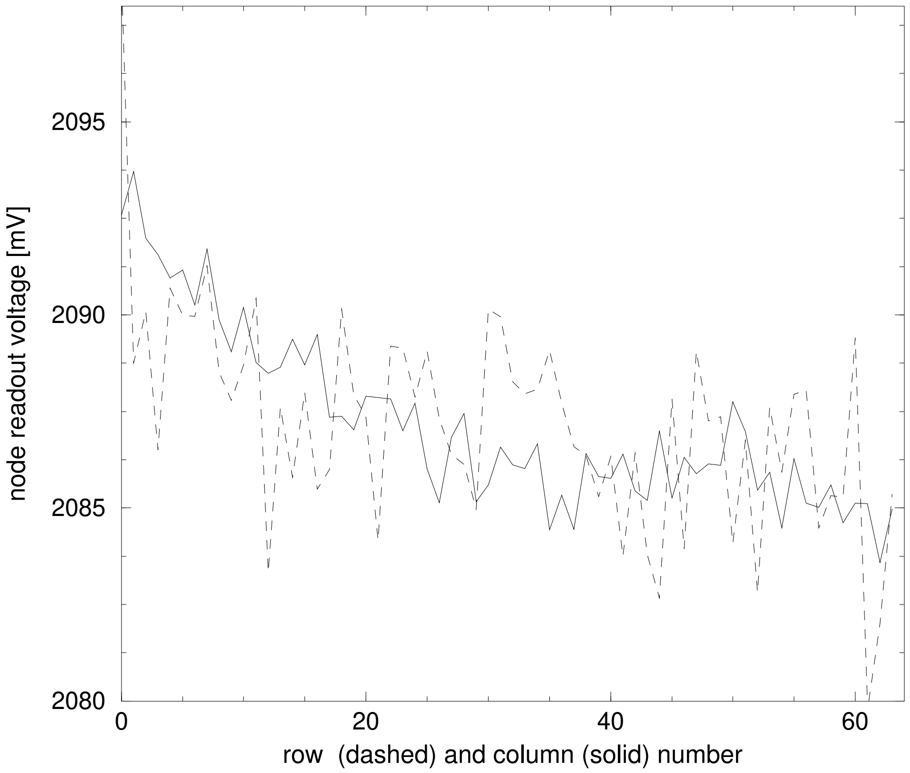



**Figure 4.3:** Comparison of row and column variations of the node readout voltage.

noise the known standard deviation of the columns can be used instead of the unknown one of the rows. With this assumption the standard deviation of the row circuits can be calculated:

$$\sigma_{\text{row circuits}} = \sqrt{\sigma_{\text{rows}}{}^2 - \sigma_{\text{columns}}{}^2} = 2.04\text{mV} \tag{4.2}$$

Each pixel contains two sources of fixed pattern noise. To measure the errors of the pixel storage switch and the readout source followers separately a different configuration of the network is needed. There is the possibility to read out the common precharge voltage of all sense amplifiers individually at each pixel. Thus the output of the readout source followers becomes independent of the pixel circuits. Clk1, clk3, $\overline{\text{clr}}$ and the precharge switch are activated together without the clk2/$\overline{\text{clk2}}$/clk2p signals. This makes a connection between the readout source follower and the precharge voltage input of EDDA via three NMOS pass transistors. The only problem is that the analog readout needs one of the $V_{\text{out}}$ lines and therefore the precharge switch and clk3 must be deactivated before the readout starts. The nodes are still connected to each other via the sense amplifiers as long as $\overline{\text{clr}}$ stays active but are floating at large. 200 frames have been recorded in this way with $V_{\text{precharge}}$ at 1.6 V. The standard deviation of the pixels' temporal mean values is 4.74 mV. Comparing this value with figure 4.2 shows that the major part of the fixed pattern noise is caused by the pixel source followers. This is a consequence of the fact that they are not needed for the edge detection operation of the array and therefore have been implemented on a small silicon area without any methods to reduce their fixed pattern noise besides transistor size. The error caused by the analog storage circuits can be calculated according to equation 4.2:

$$\sigma_{\text{analog storage circuits}} = \sqrt{5.44^2 - 4.74^2}\ \text{mV} = 2.67\ \text{mV} \tag{4.3}$$

This can be verified by taking the difference of the data points from both measurements. The standard deviation of this new data set is 2.61 mV. This is in accordance with eq. 4.3. To isolate the contribution of the pixel storage switches the respective row mean value has been subtracted from each pixel in the new data set. The standard deviation of the resulting data points is 1.08 mV. After subtracting the column instead of the row mean values it is 2.50 mV. This shows that there are also some column correlations in the data. Most likely this is the result of the global gradient superimposed on the random pixel variations.

The subtraction of the row averages does not only suppress the row-wise correlations in the data but also further reduces the width of the original distribution for statistical reasons. The standard deviation $\sigma_{\text{result}}$ of the data resulting from such a subtraction is:

$$\sigma_{\text{result}} = \sqrt{\sigma_{\text{true}}^2 - \sigma_{\text{rows}}^2} \tag{4.4}$$

$\sigma_{\text{true}}$ is the - unknown - standard deviation of the data and $\sigma_{\text{rows}}$ the standard deviation of all the 64 row mean values. For 64 rows of spatially uncorrelated data $\sigma_{\text{rows}}$ is always equal to:

$$\sigma_{\text{rows}} = \frac{1}{\sqrt{64}}\sigma_{\text{true}} \tag{4.5}$$

Therefore $\sigma_{\text{true}}$ can be calculated from $\sigma_{\text{result}}$ as follows:

$$\sigma_{\text{result}} = \sqrt{\sigma_{\text{true}}^2 - \frac{1}{64}\sigma_{\text{true}}^2} = \sqrt{\frac{63}{64}}\sigma_{\text{true}} \tag{4.6}$$

Since this correction is below 1 % it is ignored throughout this section.

The column variations are caused mostly by the node readout source followers. From the analog storage circuits only the row variations produced by the storage amplifiers have a significant contribution to the fixed pattern noise. Figure 4.4 shows a comparison between both cases. The row mean values are depicted on the left, the column ones on the right. In both graphs the dotted curve was measured with the pixel voltage sources as input and the solid one with the precharge voltage, respectively. Each curve is centered around their mean value and the voltages have been converted into the internal node voltages by equation 4.1. It can be seen that the column curves are nearly identical while the row traces show the additional variations caused by the storage amplifiers. The row variations in the case without the analog storage circuits are not different from the column ones. To compensate the global drift a linear regression has been carried out for the four curves. The root mean squares of the dashed curves are 3.4 mV (rows) and 1.8 mV (columns), of the solid ones 1.8 and 1.7 mV, respectively. The fact that there is no significant difference between the row and column rms in the measurement using the precharge voltage as input is an experimental verification that the variation of the voltage drop across the row readout switches is below 1 mV (see section 2.3.2).

### Charge injection

The absolute node voltage difference between both measurements (calculated from the mean value of all pixels) is 37.4 mV. This is caused by the charge injection of the transistors that connect the

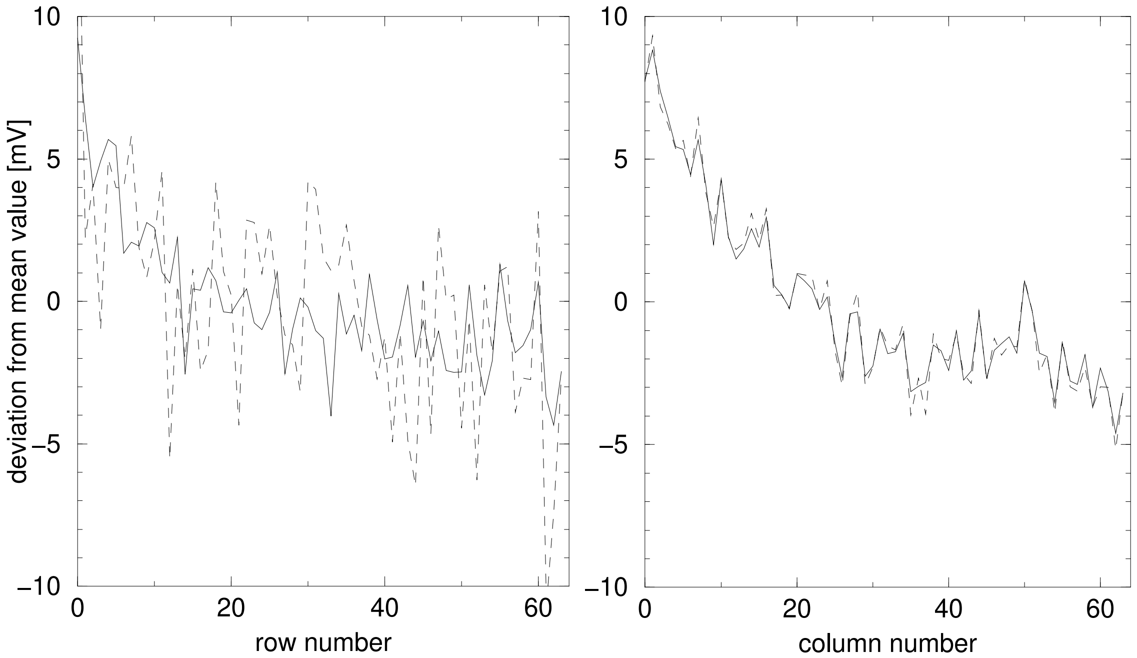



**Figure 4.4:** Comparison of row and column variations. The dotted curves are measured with the pixel voltage sources, the solid ones with the precharge voltage as input signal. The mean values of the source follower signals perpendicular to the direction plotted on the x-axes are shown in the graphs. The four curves are shifted and scaled in a way that their mean value is zero and their amplitude represents the internal node voltage deviation from this mean.

sense amplifiers to the $V_{\mathrm{out}}$ lines (M13 and M14 in figure 2.5). As mentioned before they must be switched off when the readout starts. The differences in the charge injection between the individual transistors are measured with a modified sequencer program: Before the readout starts $\overline{\mathrm{clr}}$ is deactivated prior to clk3. Therefore the nodes are isolated from each other. Four sense amplifiers are connected to each node and the charge injection of four transistors adds up at the readout source follower. The precharge voltage was 1.6 V and 200 images have been recorded. The temporal mean values of the pixels are calculated. The mean value of the total of the pixels is subtracted from each data point. The same has been done with the data taken with $\overline{\mathrm{clr}}$ active. Afterwards both data sets are subtracted from each other. The remaining standard deviation is 2.06 mV. This is the error caused by four transistors, the width of the distribution of the individual transistor's errors is smaller by a factor of $\sqrt{\#\text{ of transistors}}$, i.e. 1.03 mV. To check this indirect method the data with $\overline{\mathrm{clr}}$ active has been measured a second time on a different day. They were processed in the same way (subtracted form the original data set with $\overline{\mathrm{clr}}$ active). This resulted in a standard deviation of 178 $\mu$V.

The charge injection variation caused by the transistors activated by clk1 (M7 and M8 in figure 2.5) can be measured alike. The sequencer program switches off clk1 before disconnecting the sense amplifier from the $V_{\mathrm{out}}$ lines. The total change of the node voltage caused by the charge injection is 84.9 mV. The difference to the value of M13/M14 is explained by the different transistor sizes and the smaller capacitance. If clk3 is deactivated the node and one side of four sense amplifiers will be connected. In the case of clk1 only the node capacitance remains. The standard deviation of the charge injection error is 2.03 mV, i.e. 1.015 mV for the individual transistor. Expressed in percent of the total charge injection the error of M13/M14 is 4.8 % and of M7/M8 9.1 %. To compare the absolute charge injection error of both transistors the different capacitances must be taken into account. Since they can not be measured directly only an approximation based on the layout values can be made. The node capacitance is almost equal to four times one sense amplifier half, thus the charge injection error of M7/M8 must be divided by two. Therefore the absolute size of the charge injection error will be reduced by a factor of two if the transistor is increased from a W/L of 0.8/0.6 to 2/0.8.

### 4.2.3 Dynamic Range of the Clocked Network

For the previous measurements the network has not been clocked repeatedly. In this section the standard network timing is used but the sense amplifiers do not get a clk2/$\overline{\text{clk2}}$ signal. This keeps the switches for the capacitors $C_\text{a}$ and $C_\text{b}$ closed (see fig. 2.5) and isolates the nodes from each other. Due to the precharged sense amplifier the node voltage is reduced by $\lambda_\text{sa}$ according to eq. 1.24. This has been verified by using the graylevel ramp shown in figure 4.1 again. The input voltage range has been extended from 0 to 4 V. The bias voltages are the same as before: 1 V for the pixel and 3.98 V for the readout source followers. The precharge voltage has been set to 1.4 V and $V_\text{off}$ to 0 V. The sequencer clock frequency will be 20 MHz if not stated otherwise. 100 network cycles were executed before the node voltages have been read out. This was repeated 200 times and the temporal mean values were calculated. Figure 4.5 shows the node voltage as a function of the input voltage. The results of four different measurements are drawn. The upper solid trace shows the $V_\text{precharge}$ dependence of the node voltage. The lower one was recorded with the parameters stated above. In the case of the trace marked with circles the clock frequency was increased to 88 MHz and the last one - denoted by rhombi - was measured with an increased pixel bias voltage of 1.2 V. The three curves with the same precharge voltage are nearly identical within an input voltage range up to 3 V. The higher pixel bias voltage limits the dynamic range of the pixel source follower since the higher current leads to an increased gate-source voltage. It must be subtracted from the maximum output voltage of the storage amplifier - which is about 5 V - to get the upper limit of the pixel source follower's output voltage. The lower limit of 0.4 V is the same for the three curves. It is caused by the storage switch PMOS transistor (M3 in fig. 2.4). The bulk effect rises its threshold voltage above 1.3 V. The opposite is true for the NMOS pixel source follower. At a source voltage below 0.5 V the bulk effect vanishes. This reduces the necessary input voltage. At 0.4 V the input voltage falls below the threshold voltage of the storage switch. This sets the lower limit of the input voltage.

The parameter $\lambda_\text{sa}$ has been calculated from the data in the input voltage interval from 0.4 to 3.2 V (lower solid curve). The slope of this curve is equal to $1-\lambda_\text{sa}$. A linear regression has been carried out to calculate $\lambda_\text{sa} = 0.612$. According to eq. 1.24 the node voltage axis intercept should be equal to $\lambda_\text{sa}V_\text{precharge}$. The result from the regression is 0.876 V and $\lambda_\text{sa}V_\text{precharge} = 0.857$ V in good accordance. To verify eq. 1.24 the measurement with $V_\text{precharge} = 1.8$ V has been analyzed in the same way, resulting in $\lambda_\text{sa} = 0.607$. The calculated intercept is $0.607 \times 1.8$ V= 1.093 V, compared to a measured one of 1.096 V. The root mean square error of the regression was 1.6 mV in the first and 2.4 mV in the second case.

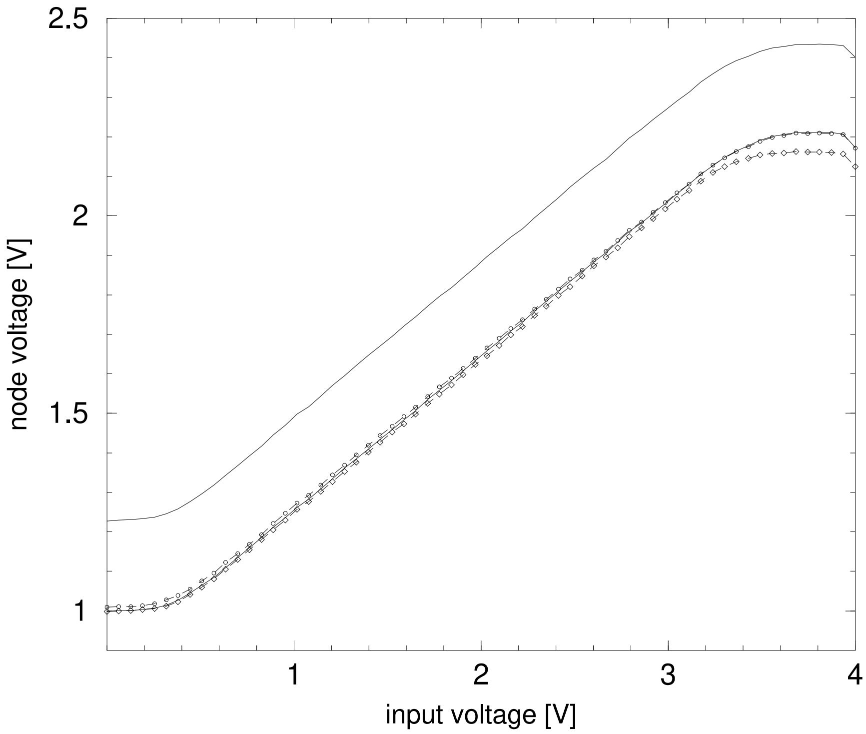



**Figure 4.5:** Analog input voltage versus node voltage. Four cases are shown: $V_{\text{precharge}}$ = 1.8 V (top solid), $V_{\text{precharge}}$ = 1.4 V (solid), clock = 88 MHz (dashed, circles) and pixel bias voltage = 1.2 V (dashed, diamonds). If not stated otherwise the clock frequency is 20 MHz, the pixel bias 1 V and $V_{\text{precharge}}$ 1.4 V.

### 4.2.4 Fixed Pattern Noise of the Clocked Network

The fixed pattern noise of the clocked network was measured for the following four different operation conditions:

1. The clock cycle is composed of clk1, clk2p and clk3/$\overline{\text{clr}}$.
2. As above, but clk2/$\overline{\text{clk2}}$ are also activated.
3. Standard network timing as described in section 2.2.3 with $V_{\text{off}} = 0.1$ V.
4. As above, but $V_{\text{off}} = 0$ V.

The input voltage was 1.8 V, the clock frequency 20 MHz and the precharge voltage 1.4 V. The $V_{\text{off}}$ voltage of 0.1 V is high enough to ensure that all the resistive fuses are in their linear region while at $V_{\text{off}} = 0$ V the fixed pattern noise of the sense amplifiers and the pixels makes nearly all the fuses blow. They would only stay intact if the difference between two neighboring pixels became smaller than the temporal noise seen by the respective sense amplifier. This is unlikely since the fixed pattern noise is more than ten times larger than the temporal fluctuations. A problem showed up during this measurements. As soon as clk2 and $\overline{\text{clk2}}$ are used together with the other

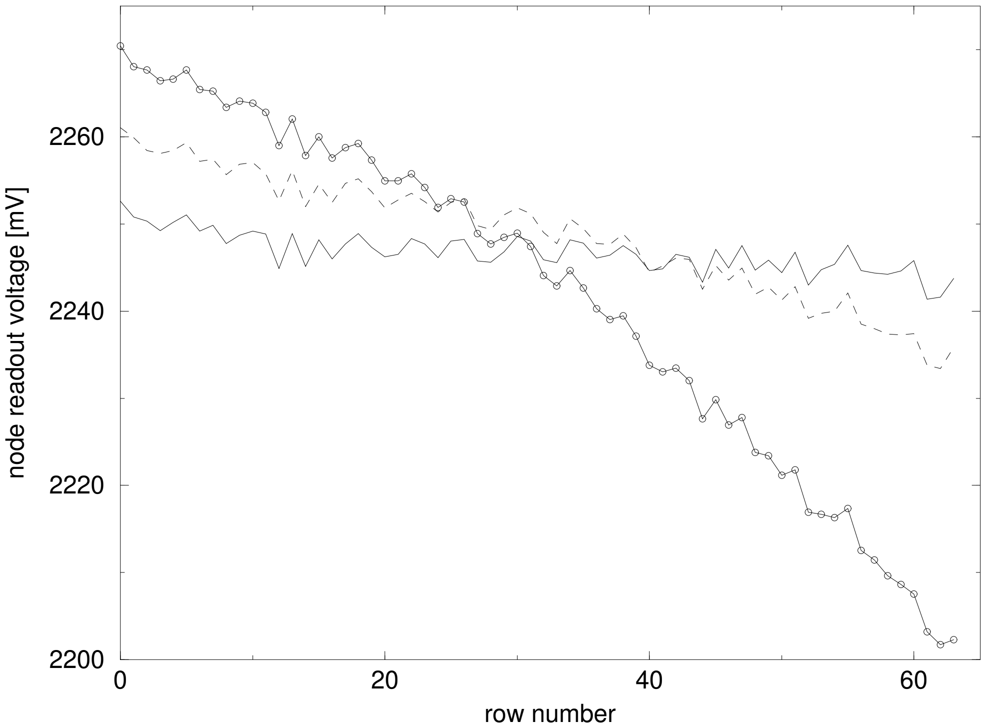



**Figure 4.6:** Node readout voltage versus row number for different clock frequencies. Solid trace: clock frequency is 20 MHz and clk2/$\overline{\text{clk2}}$ are inactive. Dashed: standard network operation with 20 MHz. Solid with circles: as before, but with 40 MHz. The input voltage is 1.8 V.

clocks a gradient can be observed in the data. The higher the clock frequency the more the node values are reduced towards higher row numbers. Figure 4.6 shows the row mean values for three different configurations. The solid trace is the data from the network without clk2/$\overline{\text{clk2}}$ at 20 MHz, the dashed one uses the standard network timing (at 20 MHz) and the one marked by circles was measured with a clock frequency of 40 MHz. The orientation of the rows on the EDDA chip is in such a way that the column clock drivers are located below row 63, i.e. on the right side of the curves in the figure. This fact, together with the frequency and clock signal dependence, may lead to the conclusion that the power dissipation of the clock drivers produces a temperature gradient in the row direction of the chip that affects the node readout source followers. The hole mobility in the channel of this PMOS transistors increases with temperature. This reduces the gate-source voltage necessary to sink the current sourced by the readout current source. Since this is common for all the pixels its temperature influences only the common offset voltage of all the pixel. Therefore the source voltage of the readout source follower decreases with temperature. Figure 4.7 shows a gray scale representation of the pixel data from the measurement at 40 MHz. The gradient in the row direction and the fixed pattern noise can be seen. The input source followers in the pixels are not affected from this. They are comprised of two transistors that are located on a common line orthogonal to the gradient. Their temperature dependence cancels (at least in first order).

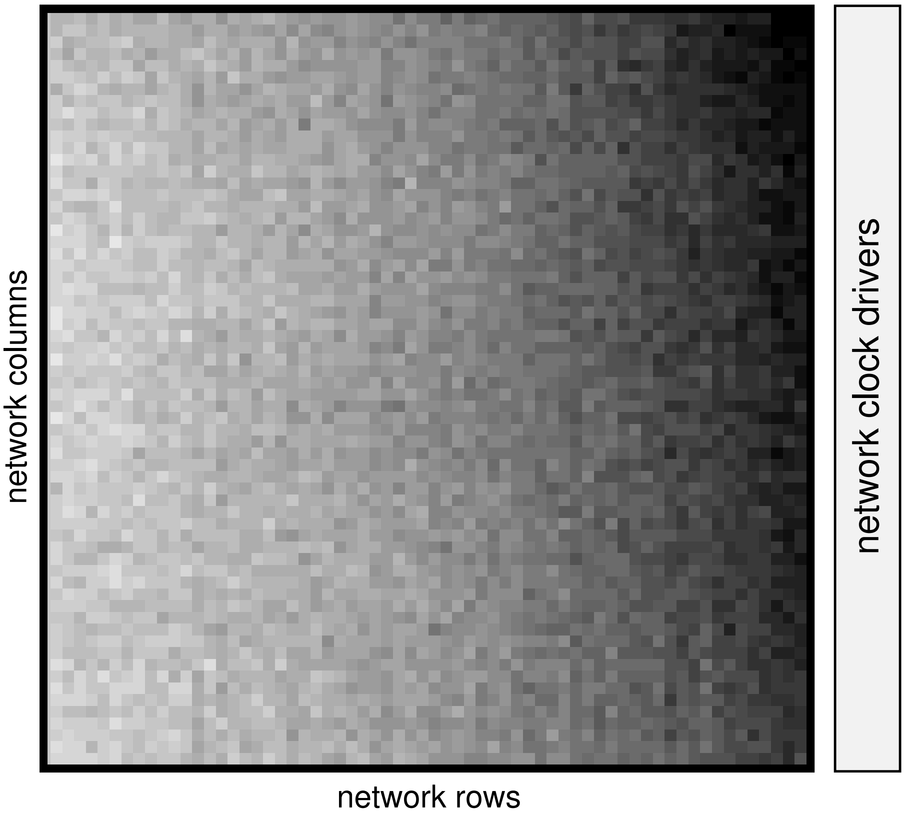



**Figure 4.7:** Gray scale image of the node readout voltages. The clock frequency has been 40 MHz. Black pixels correspond to a readout voltage of 2.2 V, white to 2.3 V. The clock drivers that cause the temperature gradient are located on the right.

To compensate this thermal gradient the row mean values have been subtracted from the measured data. Figure 4.8 shows histograms of both the original data (solid line) and the corrected ones for the fourth case of the list above. Converted from readout to node voltage the rms are 12.2 mV and 6.9 mV. The drift towards lower pixel values can be seen in the original data. A Gaussian fit has been carried out with the compensated data. It is shown as a dashed curve in the figure. The fitted width is 6.7 mV in good correspondence to the rms calculated from the data directly. The corrected results for the cases one to three are similar and listed in table 4.1. The difference between the uncorrected root mean square of number one and the remaining three cases is due to the temperature gradient caused by the clock drivers. If only four clocks are used the power dissipation of the clock drivers is not high enough to produce a significant temperature gradient across the chip. The quiescent currents of the pixel source followers determine the temperature of the chip in this case, and they are evenly spread across the array.

The fixed pattern noise of the analog readout does not matter at all for the edge detection performance. The total fixed pattern noise of the network circuits without the analog readout is:

$$\sigma_{\text{network}} = \sqrt{\sigma_{\text{all clocks}}{}^2 - \sigma_{\text{analog readout}}{}^2} = 2.8 \text{ mV}. \tag{4.7}$$

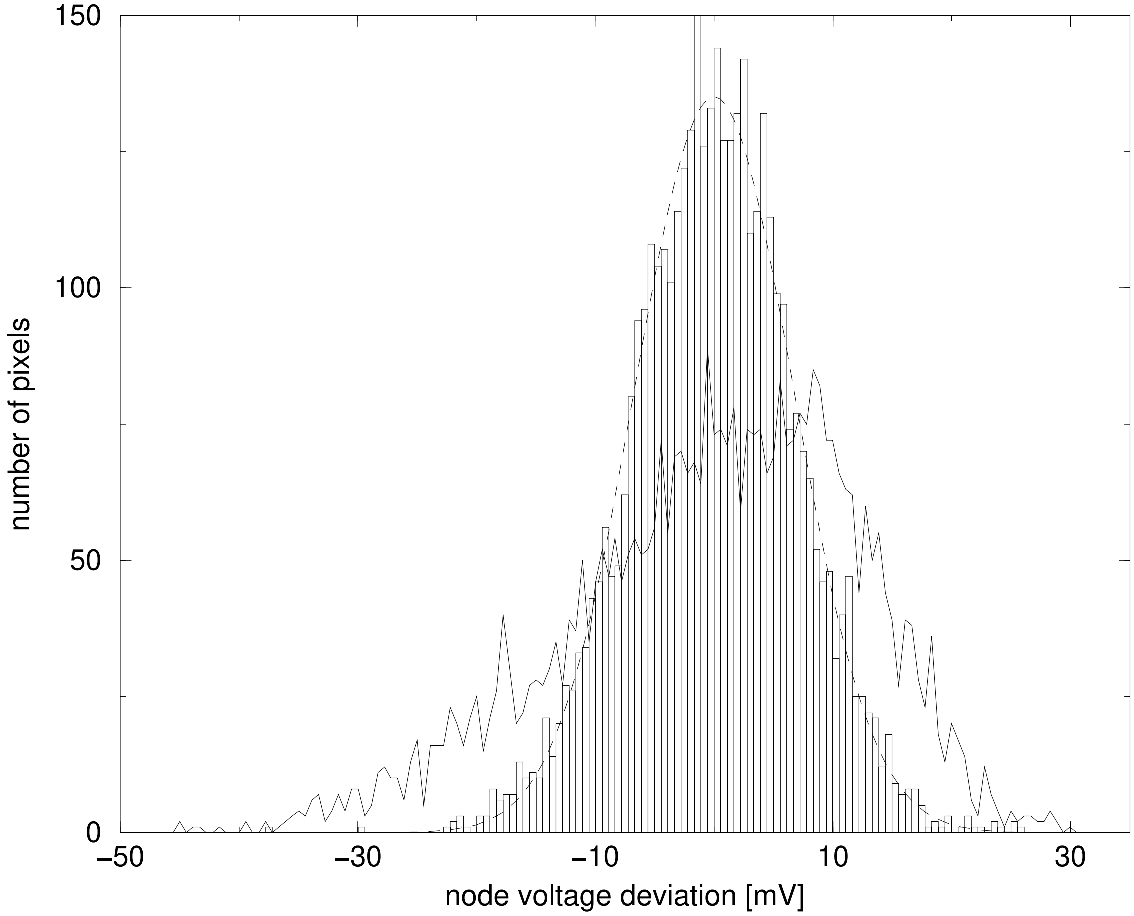



**Figure 4.8:** Histogram of the node voltage deviations for the 4096 pixels with clock signals applied to the network. The histogram depicted by the solid line is not corrected for the thermal gradient superimposed on the pixel variations. Its rms is 12.2 mV while the width of the corrected one is 6.9 mV. The dashed curve shows a Gaussian fit with $\sigma = 6.7$ mV.

| no. | rms orig. [mV] | rms corr. [mV] | clocks and parameters |
|---|---|---|---|
| 1 | 7.3 | 6.5 | clk1, clk2p and clk3/$\overline{\text{clr}}$ |
| 2 | 11.0 | 6.3 | clk2/$\overline{\text{clk2}}$ |
| 3 | 12.2 | 6.3 | all, $V_{\text{off}} = 0.1$ V |
| 4 | 12.2 | 6.9 | all, $V_{\text{off}} = 0$ V |

**Table 4.1:** rms results of the fixed pattern noise measurements of the clocked network.

$\sigma_{\text{all clocks}}$ is taken from the fourth row of table 4.1 and $\sigma_{\text{analog readout}}$ is the measured standard deviation of the readout source followers using the precharge voltage as input. The row mean values have also been subtracted from the readout source follower data before $\sigma_{\text{analog readout}}$ has been calculated.

### 4.2.5 Fixed Pattern Noise of the Sense Amplifiers

The previous section dealt with the dynamic range and the noise of the analog storage and readout as well as the different switches contained in the resistive fuse and pixel circuits. The fuse has not been used yet. For the performance of the resistive fuse network the offsets of the fuse circuit are of major importance. They are composed from the offsets of the sense amplifiers (transistors M1/M2 and M3/M4 in figure 2.5) and the variations of the $C_{\text{v}}$ capacitors.

**Distribution of the static offsets of the sense amplifiers**

The sense amplifier offsets can be measured by using the $V_{\text{out}}$ and $\overline{V_{\text{out}}}$ lines together with clk3 to input a differential voltage. Usually they will be both set to $V_{\text{precharge}}$ if the precharge switch is activated. By the mean of an extra transistor it is possible to pass the analog input voltage onto the vertical $V_{\text{out}}$ line[2]. The offset distribution of the vertical sense amplifiers has been measured with a special sequencer program that first activates clk3 and precharge to set the sense amplifiers to $V_{\text{precharge}}$. Afterwards precharge is disabled and the vertical $V_{\text{out}}$ line is set to $V_{\text{precharge}} + V_{\text{offset}}$. At last clk3 is switched off and clk2/$\overline{\text{clk2}}$ are activated. One half of a burst readout cycle is used to read back the sense amplifier states, i.e. phase2 is omitted. The clock frequency was 10 MHz. To make sure that only the static offsets were measured and every possible dynamic effect was suppressed the precharge voltage has been applied for 100 $\mu$s and the analog input voltage for 2 $\mu$s.

Figure 4.9 shows the results. $V_{\text{precharge}}$ has been 1.6 V. The offset voltage was swept from $V_{\text{precharge}} - 50$ mV to $V_{\text{precharge}} + 50$ mV in steps of 0.5 mV. 50 measurements have been carried out at each offset voltage, totaling in $41 \cdot 10^6$ sense amplifier state bits. The number of amplifiers that have switched to their high active side ($V_{\text{SA}} = V_{\text{DD}}$) were calculated for each measurement. The mean value of the 50 measurements per offset voltage is plotted versus $V_{\text{offset}}$. The error bars represent the standard deviations of the 50 data points. At the left side, at offset voltages smaller than -20 mV every sense amplifier switches to the low active side ($V_{\text{SA}}$ =GND). With increasing offset value the number of amplifiers that flipped gets larger. At an offset value of zero half the amplifiers switch to the high active side. If the offset is increased beyond 20 mV the total of 4096 sense amplifiers have switched that way.

The resulting curve is the integral of a Gaussian distribution, since at a given $V_{\text{offset}}$ all the sense amplifiers with an offset voltage smaller than $V_{\text{offset}}$ have switched to the same side. Therefore the slope at the point of inflection is proportional to the width $\sigma$ of the offset voltage distribution. The Error function has been fitted against the data to evaluate $\sigma$. The Error function erf$(x)$ is defined as the integral of the Gaussian distribution:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt \tag{4.8}$$

[2] M7 in figure 2.16 bypasses the analog storage amplifier in the row storage circuit. The global write signal stays deactivated and the row shift register is filled with high-bits to enable the analog input transmission gate of every row.

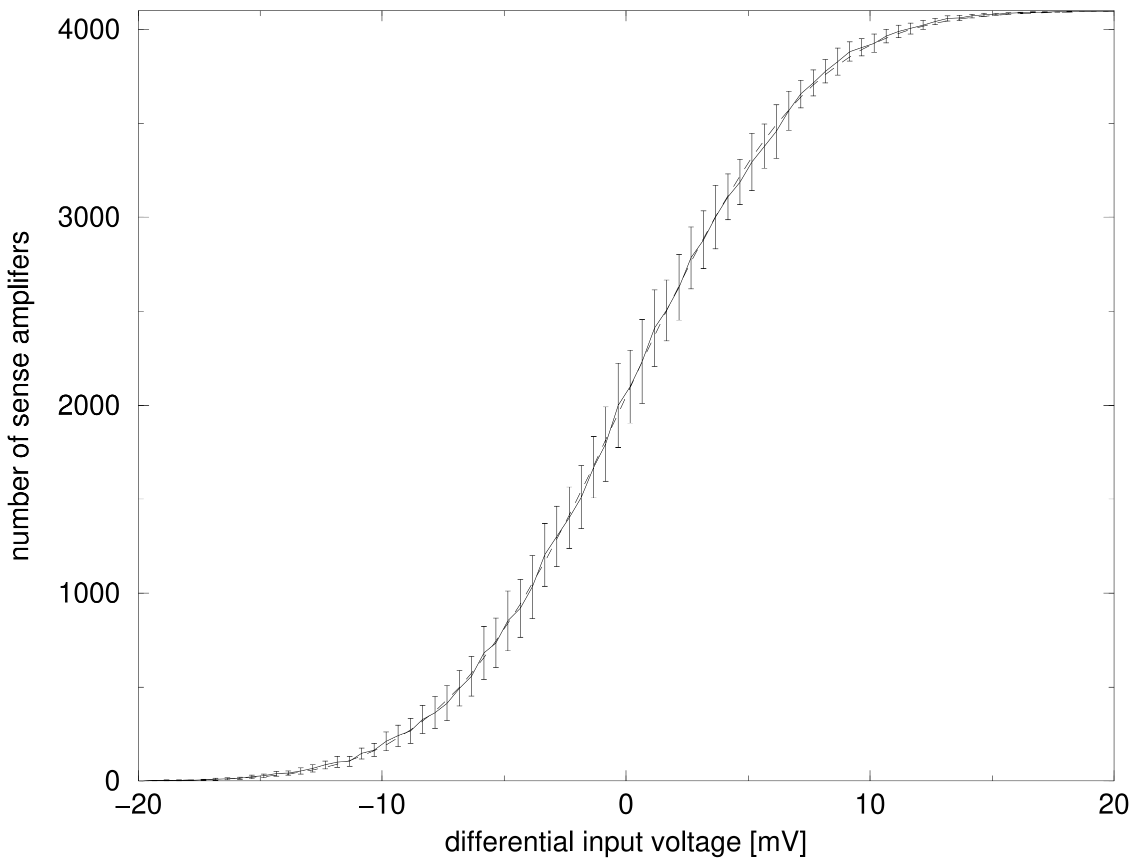



**Figure 4.9:** Number of sense amplifiers that have switched to their high active side versus their differential input voltage. The dashed line shows the Error function fitted to the measured data. The common-mode voltage was 1.6 V.

The data has been fitted against the function $f(x) = 2048 \cdot \mathrm{erf}((x - A_0)/A_1) + 2048$ to get the parameters $A_0$ and $A_1$. The dashed curve shows the result. It can be seen that it is in very good accordance to the measured data. The x-axis has been shifted to $A_0$ which is equal to $V_{\mathrm{precharge}}$. $A_1$ is equal to $\sqrt{2}\sigma$ due to the definition of $\mathrm{erf}(x)$. The resulting $\sigma$ is 5.87 mV. The same measurement has been carried out again with different common-mode input voltages $V_{\mathrm{precharge}}$ at 1.4 V and 2.5 V. Table 4.2 summarizes the results. A significant part of this error is due to the charge injection variation of the clk3 transistors and not caused by the sense amplifier itself. The width of their expected error distribution is 2.9 mV (see section 4.2.2). In the third column of table 4.2 this value has been subtracted from the measured data to obtain the width of the sense amplifier offset distribution.

#### $\lambda_{\mathrm{off}}$ measurement

The $C_{\mathrm{v}}$ capacitors in the sense amplifier form a capacitive divider together with the total capacitance of the sense amplifier circuitry. This ratio $\lambda_{\mathrm{off}}$ can be measured by applying a fixed differential input voltage $\Delta\, V$ to the sense amplifiers and varying $V_{\mathrm{off}}$. What side the sense amplifier

| common-mode voltage [V] | $\sigma_{\text{measured}}$ [mV] | $\sigma_{\text{sense amplifier}}$ [mV] |
|---|---|---|
| 1.4 | 6.0 | 5.3 |
| 1.6 | 5.7 | 4.9 |
| 2.0 | 5.8 | 5.0 |
| 2.4 | 6.1 | 5.4 |

**Table 4.2:** Width of the offset distribution of the sense amplifiers.

switches to is governed by the following equation (see section 1.3.1, eq. 1.18)

$$\Delta V' = 2\lambda_{\text{off}} V_{\text{off}} - \Delta V \tag{4.9}$$

If $\Delta V'$ is positive, $V_{\text{sa}}$ will go to $V_{\text{DD}}$. $\Delta V$ is generated by storing a chess board pattern in the analog storage capacitors of the pixels. Thereby $\Delta V$ is positive for one half of the sense amplifiers. The sequencer program used for this measurement enables both clk2p and clk1 to pass this pattern onto the sense amplifiers gates. The polarity of $V_{\text{cmp}}$ is reversed next. Thus the sense amplifier voltage $\Delta V$ changes to $\Delta V'$. Afterwards clk2/$\overline{\text{clk2}}$ are activated and the sense amplifier states are read back. For small $V_{\text{off}}$ values all the sense amplifiers switch to their low active side. As $2\lambda_{\text{off}} V_{\text{off}}$ comes near $\Delta V$ more and more sense amplifiers start to switch to their high active side. At high $V_{\text{off}}$ values one half of them has switched to this side. The other half - with negative $\Delta V$ - will not switch if $V_{\text{off}}$ is increased. The errors of both the $C_{\text{v}}$ capacitances as well as the sense amplifiers determine the slope of this transition. As described in the previous paragraph it can be approximated by the Error function to calculate the point where $2\lambda_{\text{off}} V_{\text{off}} = \Delta V$ and the width of the offset distribution.

The measurement has been carried out six times with different common-mode input voltages, ranging from 1.25 to 2.5 V. The differential input voltage has been 30 mV. Figure 4.10 shows a plot of the number of sense amplifiers with positive $\Delta V'$ versus the $V_{\text{off}}$ value for the six measurements. The maximum number of sense amplifiers is $62^2 = 3844$ since the sense amplifiers in the outermost row - of the $64 \times 64$ array - do not get the correct input pattern due to their location at the edge of the array and therefore are not counted. The dashed curve depicts the fitted Error function for 2 V. The error bars are the standard deviation of the 25 data points taken at each $V_{\text{off}}$ voltage. For common-mode voltages smaller than 1.25 V the capacitance of the $C_{\text{v}}$ capacitors becomes strongly voltage dependent because the transistor channel begins to vanish and the gate-channel capacitance drops. This effect can be seen for the 1.25 V curve. Since $V_{\text{off}}$ must be subtracted from the common-mode voltage the $C_{\text{v}}$ capacitance decreases for increasing $V_{\text{off}}$. Therefore $\lambda_{\text{off}}$ is shifted to a lower value and the curve is slanted to the right. Beyond 2.5 V the output voltage range of the pixel source followers is exceeded and no reliable results could be obtained.

The $\lambda_{\text{off}}$ values calculated from the fit results as well as the $\sigma$ of the fuse circuit derived from the measured $V_{\text{off}}$ distribution are shown in figure 4.11. The large increase in $\sigma$ at 1.25 V is a direct consequence of the strong voltage dependence of $C_{\text{v}}$ because the Error function does not fit well to the slanted curve. The voltage dependence of $\lambda_{\text{off}}$ can be explained by the sum of the following effects:

- $C_{\text{v}}$ increases with the common-mode voltage.
- Drain/source-bulk junction capacitances increase (PMOS) or decrease (NMOS) with the common-mode voltage.

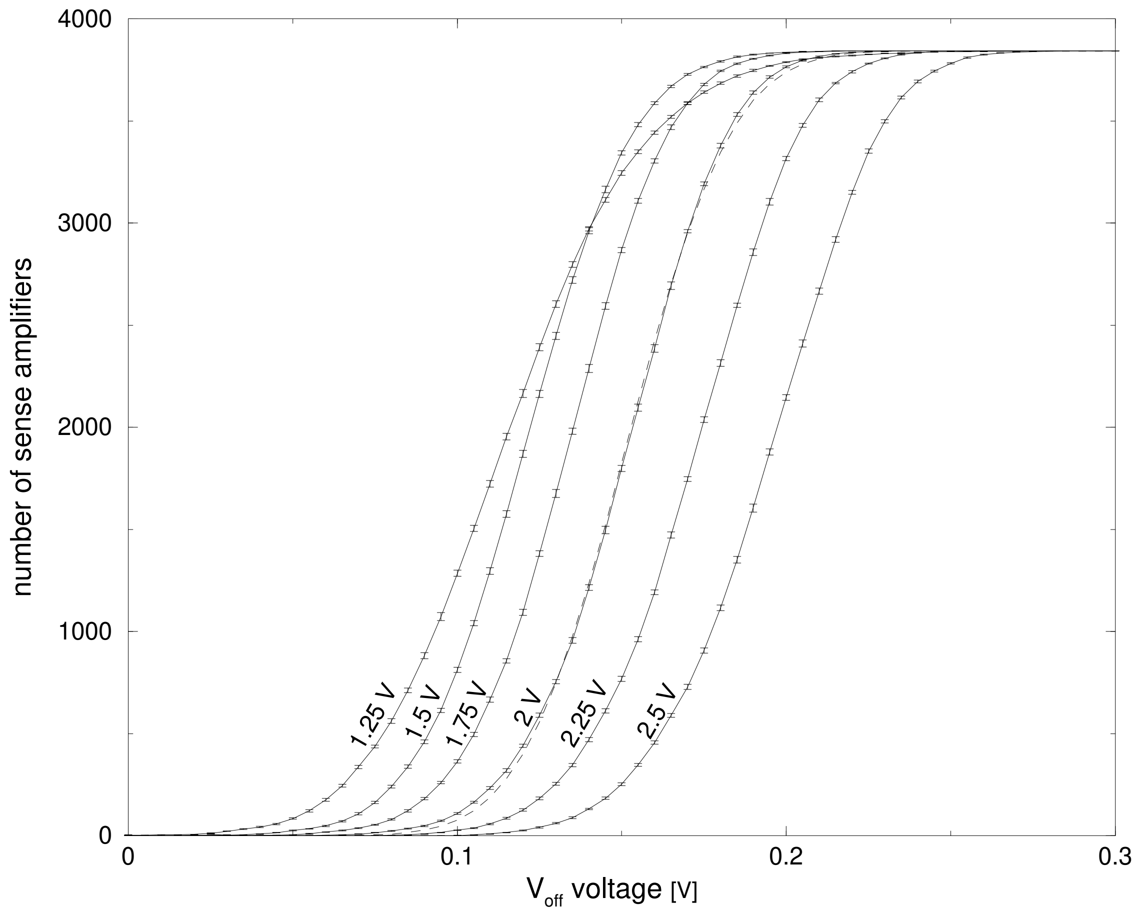



**Figure 4.10:** Number of sense amplifiers that have switched to their high active side versus the $V_{\text{off}}$ voltage. The dashed line shows the Error function fitted to the measured data. The common-mode voltages corresponding to the different curves are depicted next to them. The differential input voltage was 30 mV in all cases.

- Accumulation mode capacitances of the negatively biased NMOS transistors' gate-drain/source capacitances (those transistors that are switched off by clk1, clk2 and clk3) increase with the common-mode voltage [VIT98, BEH92] whereas the accumulation capacitance of the PMOS transistors switched off by $\overline{\text{clr}}$ and $\overline{\text{clk2}}$ decreases, but to a lesser extend since their negative bias is stronger in the common-mode voltage range measured.

This measurement has been verified at a common-mode voltage of 2 V in two ways:

1. It was repeated with a sequencer clock of 40 and 60 MHz. The $\lambda_{\text{off}}$ deviation was 1.3 % and 1.5 % respectively whereas the $\sigma$ difference was below 0.2 %.

2. The chess board pattern was reversed. Thus the other half of the sense amplifiers was examined. The $\lambda_{\text{off}}$ deviation was 0.12 % and the $\sigma$ difference 0.7 %.

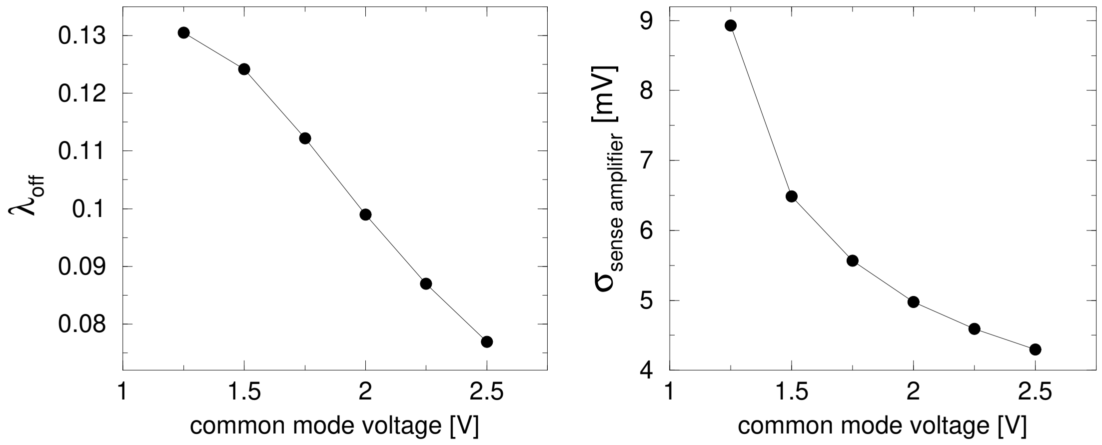



**Figure 4.11:** $\lambda_{\text{off}}$ and the width $\sigma$ of the sense amplifier offset distribution for different common-mode voltages.

## 4.3 Performance of the Resistive Fuse Network

The previous sections showed how the individual parts of the array perform, especially their dynamic range and their fixed pattern noise have been evaluated. This section covers the dynamic performance of the resistive fuse network, i.e. the interplay of all the components. The sequencer programs are the same as those used for the edge detection operation.

### 4.3.1 Dynamic Range and Precharge Voltage Selection

After a certain clock frequency has been selected three parameters remain that must be adjusted: the minimum and maximum input voltages and the precharge voltage. If they are set the only value that will be changed while the EDDA chip is used is the $V_{\text{off}}$ voltage that determines the edge level. Figure 4.12 shows the test picture that is used to determine good settings for the three parameters by visual inspection of the edge image. It is divided in three independent parts, arranged from left to right. On the left the bottom end of the input voltage range is evaluated. The black bar lies inside a graylevel ramp. With each row it decreases by 0.4 % until it reaches 0. To mark this point the gray levels below are set to 100 %. In the middle of the test image the same pattern is repeated for a gray bar (50 %) and on the right a white bar is used. The quality of the edge detection is determined by the minimum contrast the network is able to detect. In an ideal case it would detect the boundary between the black bar and the background all the way down close to zero if the the edge threshold is just above the 0.4 % per row. On the right of figure 4.12 the result of a simulation using the test image as input is shown. There are edges between the grayscale ramps and the constant regions down to a minimum contrast of about 1 %. The real chip does not get so far.

To adjust the parameters for the EDDA chip they are changed until the minimum contrast level detected is the same in all the three regions and as low as possible with respect to the noise that starts to show up at small $V_{\text{off}}$ levels. Also the noise level should be the same in the three parts. Figure 4.13 shows different edge images to illustrate this process. At the top row a good setting is

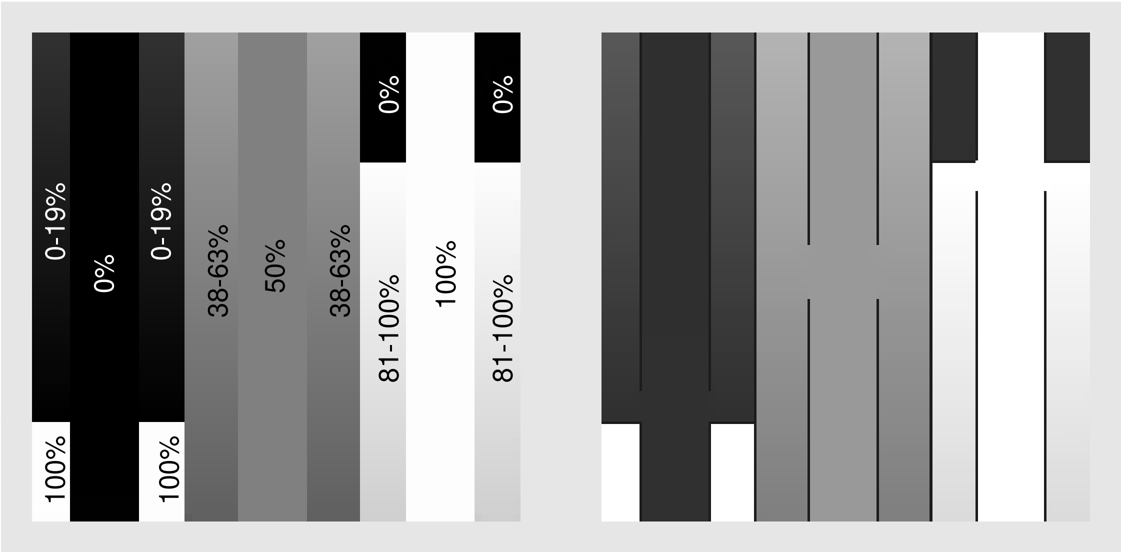



**Figure 4.12:** Left: Test image to adjust the precharge voltage and the dynamic range of the input signal. The gray levels are given in percent of the input voltage range. Right: Simulated node voltages and edges for the test image.

shown that is also used through the following sections. $V_{\text{off}}$ is decreased from left to right and it can be seen that the minimum detectable contrast gets lower but the picture also becomes noisier. The second row shows how the precharge voltage $V_{\text{precharge}}$ differently affects the noise levels in the dark and bright regions of the test picture. The reason for this is the nonlinearity of $\lambda_{\text{off}}$ measured in the previous section. To reduce the effect of this nonlinearity the dynamic range can be reduced. This has been done for the images of the bottom row. The input voltage range has been reduced to 1 V from 2.5 V. It can be observed that in fact the differences between the three regions are less than before. Otherwise this is accompanied by an increase in the overall noise level.

### 4.3.2 Fixed Pattern Noise of the Resistive Fuse Network

The fixed pattern noise (fpn) sets an upper limit to the signal-to-noise ratio of the resistive fuse network. It can not be measured by using a chess board pattern and reducing $V_{\text{off}}$ like it has been done in the previous section since the network operation introduces correlations between neighboring nodes. If no edge is present the smoothing effect of the network would change the differences between adjacent nodes from their desired values given by the chess board pattern. Therefore the input pattern has been omitted at all to measure the fixed pattern noise. If all the pixels are set to the same value and $V_{\text{off}}$ is reduced gradually the number of edges will increase from zero to the total number of sense amplifiers. This is caused by the fixed pattern noise of the fuse circuits. The non-uniformities of the analog storage and the switches inside the pixel are strongly reduced by the averaging effect of the resistive network. Figure 4.14 shows the resulting curve for an input voltage of 1.75 V. The clock frequency was 10 MHz. $V_{\text{off}}$ was decreased in 1 mV steps. The edges are read out 25 times with 50 network cycles between each readout cycle


**Figure 4.13:** Edge images generated by the resistive fuse network using the test image shown in the previous figure as input. In each row one parameter, depicted below the edge image, is varied while the others are held constant. First row (from left to right): with decreasing $V_{\rm off}$ the edges extend further along the different graylevel ramps but the noise also increases. Second row: at different precharge voltages the nonlinearity of $\lambda_{\rm off}$ causes different noise levels in each of the three image parts. Third row: by reducing the input voltage range the effect of the nonlinearity decreases at the expense of an overall increase in the noise level.

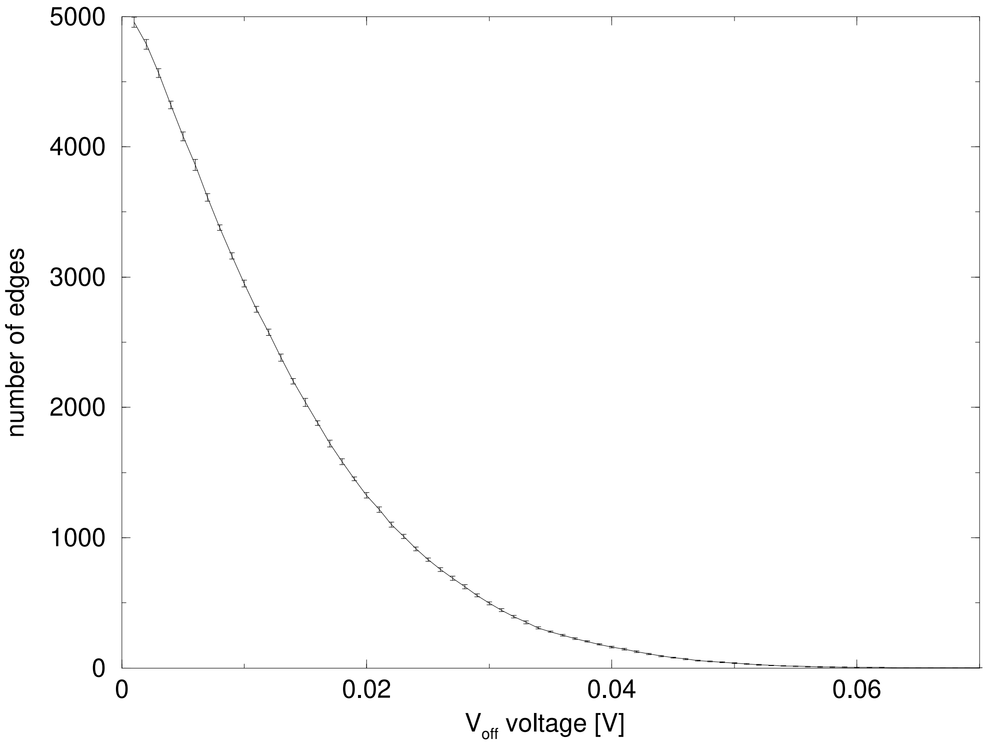



**Figure 4.14:** Number of edges versus $V_{\text{off}}$ voltage that are detected with a constant input voltage of 1.75 V.

(the burst readout was used). The error bars represent the standard deviation of the 25 data points. The precharge voltage was 1.53 V.

For low $V_{\text{off}}$ values the progression of the number-of-edges curve resembles the Error function. If $V_{\text{off}}$ crosses the threshold set by the offsets of the sense amplifier and $C_{\text{v}}$ capacitors of one individual fuse circuit this edge will be counted for every lower $V_{\text{off}}$ voltage as well, i.e. the error distribution is integrated over $V_{\text{off}}$ resulting in the Error function if a Gaussian distribution is assumed. This is no longer true if the number of edges reaches a level where the fuses that are not yet blown are influenced by their neighbors that already did. The method chosen to get the $\sigma$ of the offset distribution is therefore to fit the Error function to only a part of the data limited by the edge number. A value of 10 % has been chosen, these are $0.1 \cdot 2 \cdot 60^2 = 720$ edges. The resulting $\sigma$ is 16.6 mV. This value must be compared to the total dynamic range of the input signal to get the signal-to-fpn ratio.

This has been done by applying a pattern with a defined signal-to-background ratio to the resistive fuse network and changing $V_{\text{off}}$ while recording the number of edges. If the voltage difference on the sense amplifier inputs caused by $V_{\text{off}}$ is sufficiently higher than that produced by the input pattern no fuses are blown, i.e. no edges are detected. If $V_{\text{off}}$ is small the contrary situation will be present: all fuses along the input pattern are blown and the boundary of the pattern is shown in the edge image. Figure 4.15 shows the square used as the input pattern. The length of its edge is 17 pixels. The background voltage is 0.31 V lower than the square itself. This is 1/8th of the total dynamic range of 2.5 V used in this measurement. It is important that the shape is totally

| clk [MHz] | $\sigma_{\text{fpn}}$ [mV] | $V_{\text{off}}$ | mean value [mV] | sdev [mV] | signal-to-fpn ratio [dB] |
|---|---|---|---|---|---|
| 10 | 16.6 | down | 144 | 6.5 | 36.8 |
| " | " | up | 232 | 11.0 | 41.0 |
| 60 | 33.4 | down | 339 | 7.1 | 38.2 |
| " | " | up | 390 | 0.6 | 39.4 |

**Table 4.3:** Signal-to-fixed-pattern-noise ratio for two different clock frequencies.

enclosed by edges in the case of a low $V_{\text{off}}$ voltage to make sure that no smoothing takes place between the figure and its background. In figure 4.15 the cases with low and high $V_{\text{off}}$ voltages are shown on the left and the right. Below them the measured node voltages along a cut through the middle of the squares are shown. An example for an intermediate $V_{\text{off}}$ level is depicted by the image in the center of the top row. About half of the fuses are blown, located mainly in two edge segments. This is the effect of the spatial correlation of the resistive fuse network. If the first fuse blows in an otherwise intact edge the voltage difference across the remaining edge segments is reduced for the neighbors of the blown fuse. Therefore the next fuse that blows will be likely one of these neighbors.

Figure 4.16 shows how the number of edges change with $V_{\text{off}}$. $V_{\text{off}}$ was incremented or decremented in 1 mV steps, respectively. This is depicted by the arrows above and below the according curves. Each point is the mean value of 25 measurements at the same $V_{\text{off}}$ value and their standard deviation is represented by the error bars (only shown for the solid curve). The background input voltage was 1.6 V and the rest of the setup has not been changed from that used in the previous section. The different curves (solid, dashed and gray) represent three different measurements where the position of the square in the array was changed. The slope of the curves is caused by the fixed pattern noise. In an ideal network each curve would resemble a step function. The irregularity of the curves is caused by the spatial correlations between the network elements. To get each point of inflection a fit to the Error function was used and the mean value of the results of these fits were calculated for increasing and decreasing $V_{\text{off}}$. The results are shown in table 4.3. The signal-to-fpn ratio[3] for the case of increasing $V_{\text{off}}$ is the one that compares the unaltered input voltage to the fixed pattern noise - it is therefore the maximum signal-to-noise ratio the EDDA chip will achieve if an input signal is applied that covers the total dynamic range of the input voltages. Table 4.3 contains also the results of a second set of measurements that have been carried out with a clock frequency of 60 MHz. It can be seen that all the $V_{\text{off}}$ voltages are shifted to higher values compared with the measurements at 10 MHz. The reason for that is the on-resistance of the comparator voltage switch (see section 2.4.1). It is too high to charge the $C_{\text{v}}$ capacitors to their new value in the 17 ns a clock period lasts at 60 MHz. If an additional wait state is inserted between $\phi_1$ and $\phi_2$ the voltages observed are close to those seen at 10 MHz.

[3]The signal-to-fpn ratio ($\text{SNR}_{\text{fpn}}$) is calculated by the following formula: $\text{SNR}_{\text{fpn}} = 20 \log \frac{8\, V_{\text{fit mean value}}}{\sigma_{\text{fpn}}}$ with the mean values from table 4.3 as $V_{\text{fit mean value}}$ and $\sigma_{\text{fpn}}$ as described in the previous paragraph.

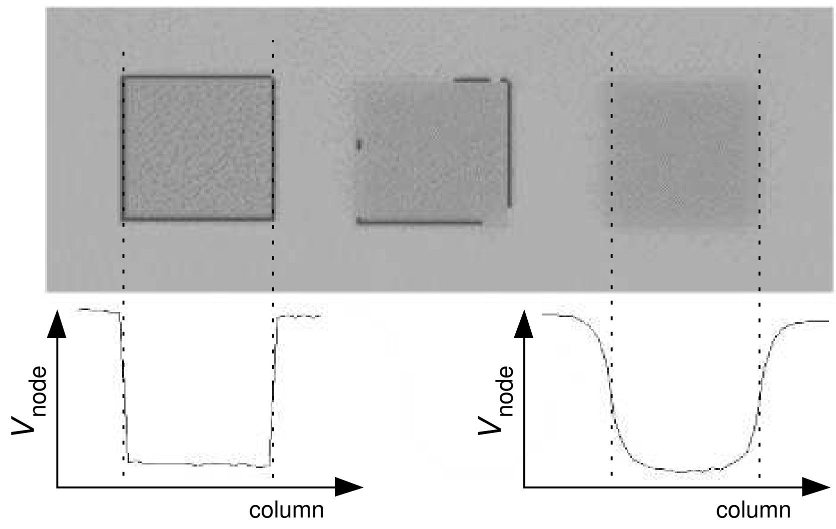



**Figure 4.15:** Gray scale image of the node voltages combined with the edge image for three different $V_{\rm off}$ voltages (top) and horizontal cuts through the middle of the left and the right image (bottom). The input pattern was a square with $17 \times 17$ pixel.

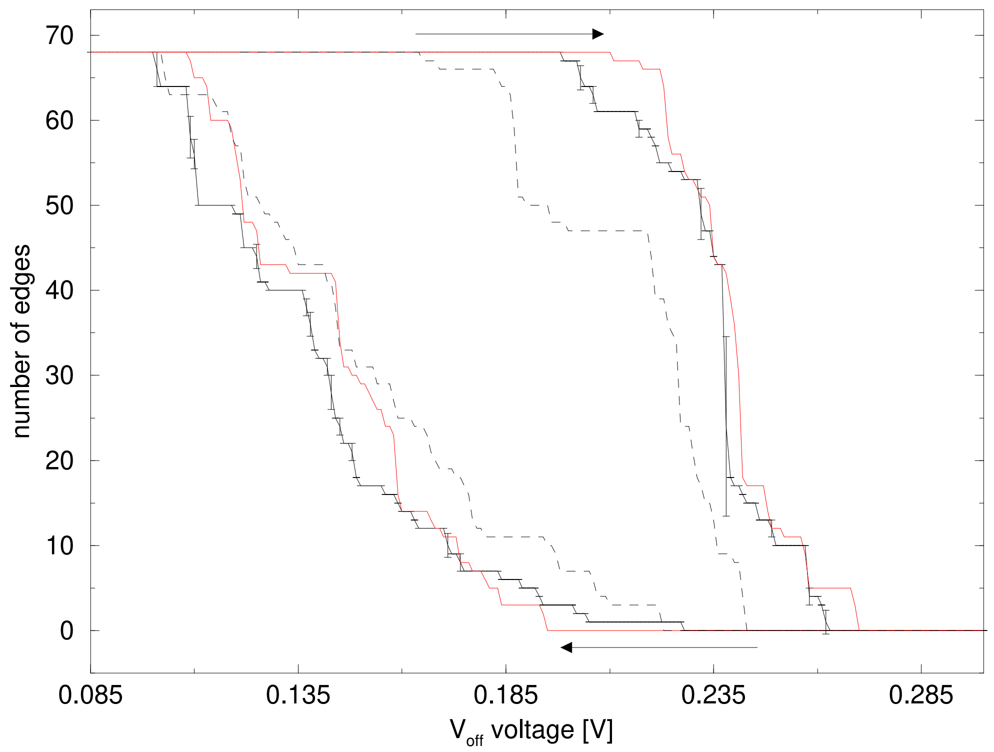



**Figure 4.16:** Number of edges versus $V_{\rm off}$ voltage. The input pattern is the same as in the previous figure. The different curves (solid, dashed, gray) represent three different locations of the square in the input image. The arrows depict the direction of the $V_{\rm off}$ steps.

### 4.3.3 Network Operation

**Detecting differently sized structures**

In section 1.2.4 the two dimensional network has been simulated to show its edge detection properties. An important aspect is that the voltage reduction caused by the smoothing process depends on the spatial size of the according structure. This was verified for the EDDA chip as well. The input image was similar to that used in section 1.2.4. It is shown in figure 4.17. Since the size of the image is constrained to $64 \times 64$ pixels some provisions must be made to avoid edge effects. The line in the middle does not reach the edges of the array. For the rectangle on the left only the leftmost part touches the edge. The response of the network to that image is shown in figure 4.18. It shows the node readout voltages along the cut marked by the arrows in figure 4.17. The fluctuations are mostly caused by the fixed-pattern-noise of the node readout source followers (see section 4.2.2). Both curves have been measured with the same $V_{\text{off}}$ value. For the solid curve $V_{\text{off}}$ has been decreased from a larger value to its final value while for the dashed one it has been increased from zero. If, in case of the dashed curve, $V_{\text{off}}$ had been decreased further the fuses at the edge of the rectangle would have blown first since there is the largest difference between neighboring fuses. The next one would have been the line and the single point is detected at last.

An example of the influence of the $V_{\text{off}}$ progression on the edge image is illustrated in figure 4.19. The source image is a cutout of a photograph showing the metal framework beneath a bridge in front of the sky. The $V_{\text{off}}$ curve is either increasing from zero to $V_{\text{end}}$ or decreasing from $2 \cdot V_{\text{end}}$ to $V_{\text{end}}$. In both cases $V_{\text{end}}$ has been 170 mV. The other EDDA settings are the same as in the previous section. It can be seen that the edge image shown in the upper right corner is full of edges at those regions of the image where small sized structures and parts of the sky are alternating. These fuses are blown from the beginning and $V_{\text{end}}$ is not high enough to overcome the node voltage differences in these parts of the image. If $V_{\text{off}}$ is decreased from a higher level towards $V_{\text{end}}$ these small structures are smoothed by the resistive network operation before $V_{\text{off}}$ reaches $V_{\text{end}}$. Like the three patterns in figure 4.18 this means a stronger reduction in the node voltage differences for the smaller than for the larger structures of the metal framework of the bridge. Therefore if $V_{\text{off}}$ reaches $V_{\text{end}}$ it will be small enough to make the fuses at the larger structures blow, but not those at the smaller ones. If $V_{\text{off}}$ is set to a constant level - $V_{\text{const}}$ - and the network is activated after several milliseconds of inactivity, it will be the same effect as if $V_{\text{off}}$ is reduced suddenly from a high level to $V_{\text{const}}$ since the subthreshold currents of the fuse resistor switches have equalized the node voltage differences. Therefore none of the fuses is blown during the first clock cycles.

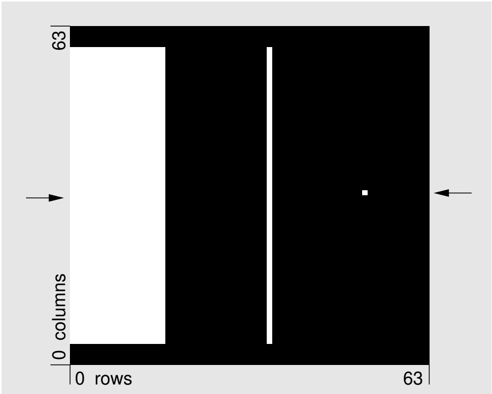



**Figure 4.17:** Test image used to measure the response of the resistive fuse network to three differently sized structures. A cut along the line depicted by the arrows is shown in the next figure.

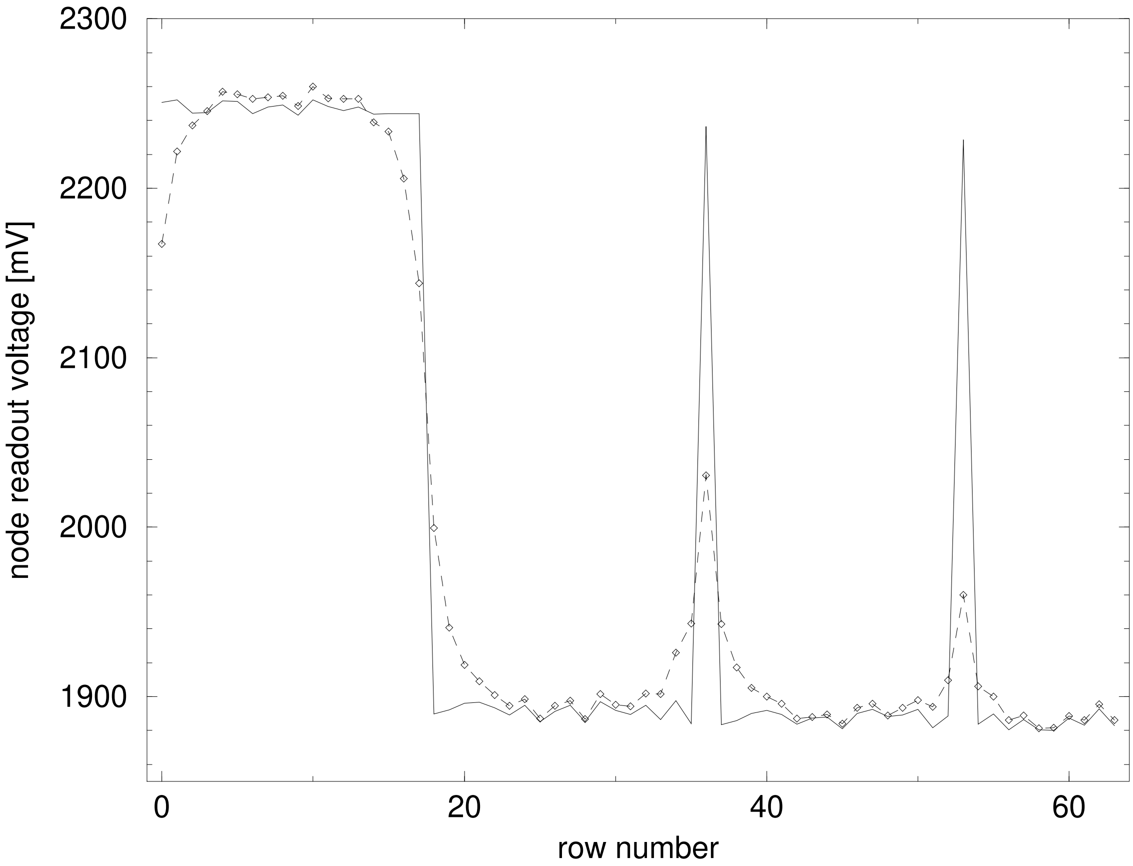



**Figure 4.18:** Example of the network operation using the image from the previous figure. The dotted line shows the measured node readout voltage while all fuses are ok. The solid one shows the case where the fuses at the edges of the structures are blown.

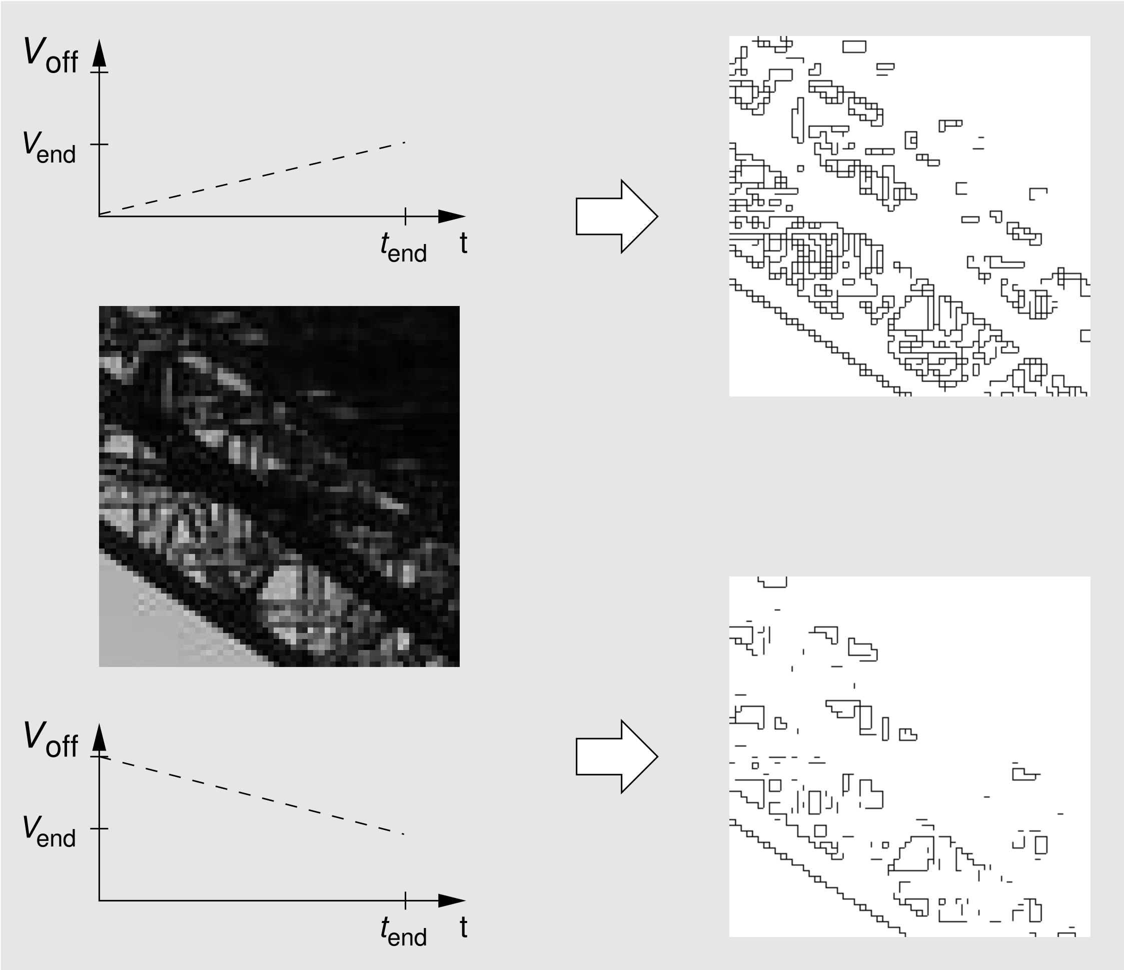



**Figure 4.19:** Influence of the $V_{\mathrm{off}}$ progression on the resulting edge image. The input image is shown on the left. Above and below it the $V_{\mathrm{off}}$ curve is depicted. For increasing $V_{\mathrm{off}}$ the edge image is shown in the upper right corner, for decreasing in the lower right, respectively.

### Characteristic length

To measure the characteristic length of the array an input image containing only one bright pixel in front of a uniform background was used. $V_{\text{off}}$ was adjusted in a way that all the fuses are ok and the array acts like an uninterrupted resistive network. In section 1.2.3 it was shown that the resulting node voltages resemble the modified Bessel function of order zero: $K_0$. Since the dynamic range of $V_{\text{off}}$ will not be high enough to keep the fuses intact if an input pattern utilizes the full dynamic range between adjacent pixels the ratio of the single test pixel to the background was limited to 1.2 V (about one half of the total dynamic range). Since the smoothing of this input signal leads to a node voltage reduction of a factor of 5 to 10, depending on the characteristic length, and the node voltage is further reduced by $\lambda_{\text{sa}}$ (see eq. 1.24) the resulting node voltage range is only about 100 mV. The fixed pattern noise of the readout source followers gets clearly visible in this region. Therefore the readout voltages have been recorded 50 times with and without the source pixel. The mean values of both data sets were subtracted from each other. Figure 4.20 shows the resulting node voltages along a horizontal cut through the array. The node voltages are normalized to their maximum value. The left case shows the standard network timing. The curve on the right uses a modified timing that increases the characteristic length by omitting clk2p from every second network cycle (for both phases). The input voltage was 2.5 V in this case.

To determine the characteristic length $l$ the Bessel function $K_0$ was fitted to the data except of the peak value, where $K_0$ diverges. The fit results are shown as dashed curves in the graphs. Eq. 1.12 was used to calculate $l$. For the left case $l = 1.2$ and for the right $l = 2.0$. This is smaller than the $l$ expected from the ratio of the pixel to fuse resistors if the layout values of the capacitances $C_{\text{p}}$ and $C_{\text{a}}/C_{\text{b}}$ are considered (see section 2.2). The relative increase of the effective $C_{\text{p}}$ caused by parasitic capacitances is much larger than for $C_{\text{a}}/C_{\text{b}}$ since $C_{\text{p}}$ is very small ($\approx$ 70 pF). This decreases the value of the SC pixel resistor and thereby the characteristic length. The right curve proves that $l$ can be changed by periodically omitting the clock for the pixel resistor. The higher noise level is due to the fact that the maximum node voltage in the right graph was only 60% of that in the left one. The signal reduction caused by the smoothing is 3.2 times stronger than in the right case.

### Power consumption

The power consumption was measured while executing standard network cycles using the gray scale test image as input (see section 4.3.1) in an endless loop. The digital and analog power supplies were measured separately. Tables 4.4 and 4.5 summarize the results for the different clock frequencies and bias voltage settings. The analog supply current was independent of the clock frequency. For the digital supply current the PLL multiplier and the bus frequency are stated. The sequencer clock is the product of both.

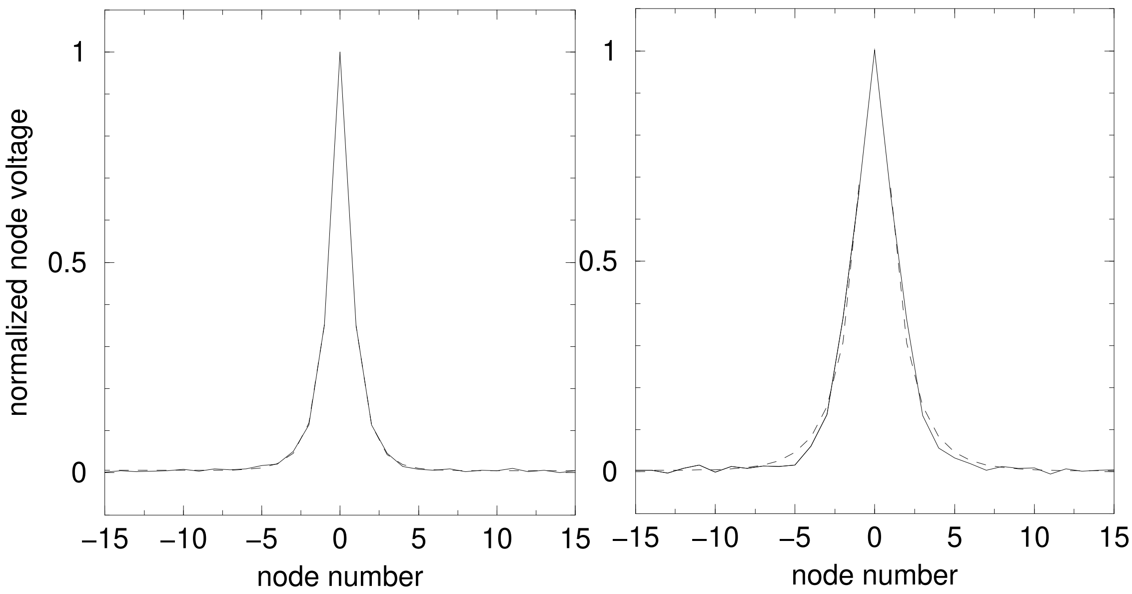



**Figure 4.20:** Smoothing of a single pixel's signal by the resistive fuse network shown for two different network timings. On the left the standard network timing has been used while on the right the clk2p signal has been omitted from every second network cycle. The dashed curves show a fit of the Bessel function $K_0$ to the measured data excluding the interval from -1 to 1.

| bus clock [MHz] | PLL mul. | current [mA] |
|---|---|---|
| 10 | 1 | 9 |
| 20 | 1 | 12 |
| 10 | 3 | 12 |
| 15 | 3 | 14 |
| 15 | 4 | 15 |
| 22 | 4 | 19 |

**Table 4.4:** Digital supply currents.

| pixel bias[V] | readout bias[V] | current [mA] |
|---|---|---|
| 1 | 3.96 | 7.7 |
| 0 | 3.96 | 2.7 |
| 0 | 5 | 2.1 |
| 0.96 | 3.96 | 4.5 |
| 1.04 | 3.96 | 8.6 |
| 1.08 | 3.96 | 10.9 |
| 1.12 | 3.96 | 13.7 |

**Table 4.5:** Analog supply currents.

| parameter | value | parameter | value |
|---|---|---|---|
| bus clock | 15 MHz | pixel bias | 1 V |
| PLL mul. | 4 | readout bias | 3.96 V |
| input data rate | 3 MHz | input voltage range | 0.5 - 3 V |
| analog readout rate | 0.6 MHz | precharge voltage | 1.5 V |
| edge readout rate | 15 MByte/s | network cycles | 10 + 1 readout |

**Table 4.6:** EDDA parameters as used in the figures 4.21, 4.22 and 4.23.

### 4.3.4 Example Pictures

Figure 4.21 shows an example image[4] and the results of the processing by the EDDA chip. Table 4.6 summarizes the electrical parameters used. The standard network timing as described in section 2.2.3 was used. After 10 network cycles one burst readout cycle is performed to get the edge image before the node voltages are read out. The total processing time per image was 1.7 $\mu$s. The different $V_{\text{off}}$ voltages used are depicted below the edge images. $V_{\text{off}}$ was held constant throughout the image processing.

#### Noise suppression

Due to the selective smoothing in the regions between the blown fuses the resistive fuse network is insensitive to noise as long as it exhibits no spatial correlations. Figures 4.22 and 4.23 show the response of EDDA to different noise levels in the input image. The example picture from the previous section was used again. Gaussian distributed noise was added. Three different noise levels were used: 2.5, 5 and 10 %. These percentages stand for the ratio of the width $\sigma$ of the distribution to the total dynamic range of the input signal. Since the parameters are also the same as in the previous section the $\sigma$ values of the noise expressed in volts are: 63 mV, 125 mV and 250 mV. The different $V_{\text{off}}$ values used are stated below the according images.

It can be seen that the edge images at lower $V_{\text{off}}$ values are more sensitive to the added noise than those at higher values. At higher $V_{\text{off}}$ voltages a smaller part of the noise induced pixel-to-pixel differences falls below the edge threshold, i.e. more of them are smoothed by the resistive network operation. The node voltage images show that the EDDA chip preserves the edges while suppressing the noise in-between. For the 10 % noise the source image has been also processed by a $5 \times 5$ Gaussian filter matrix [JAE93-2]. While its smoothing effect is comparable to that of the resistive network the edges are totally blurred while the EDDA chips preserves at least the strong ones.

[4] The image shows a part of a picture taken from [HBV99].

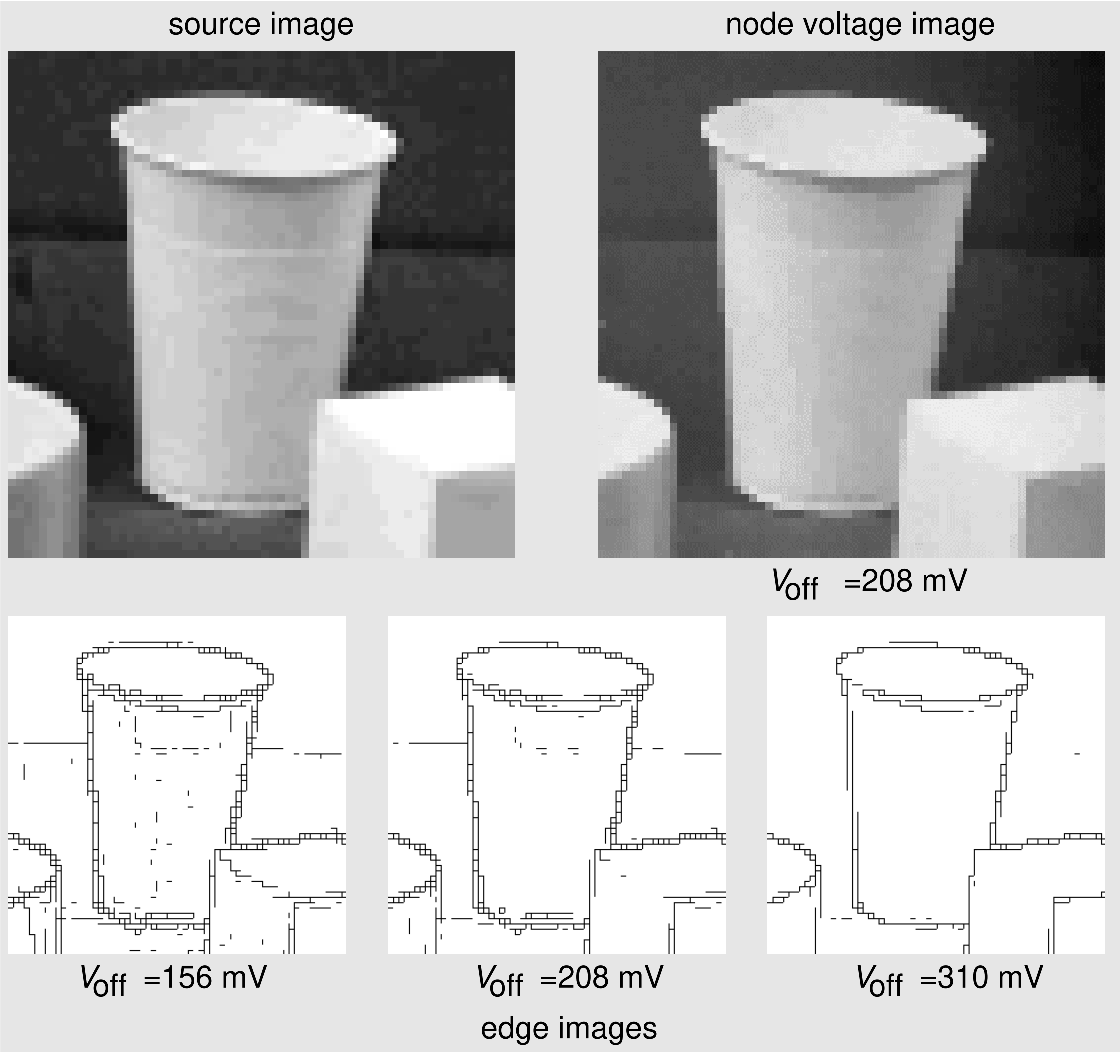



**Figure 4.21:** Example image processed by the EDDA chip.

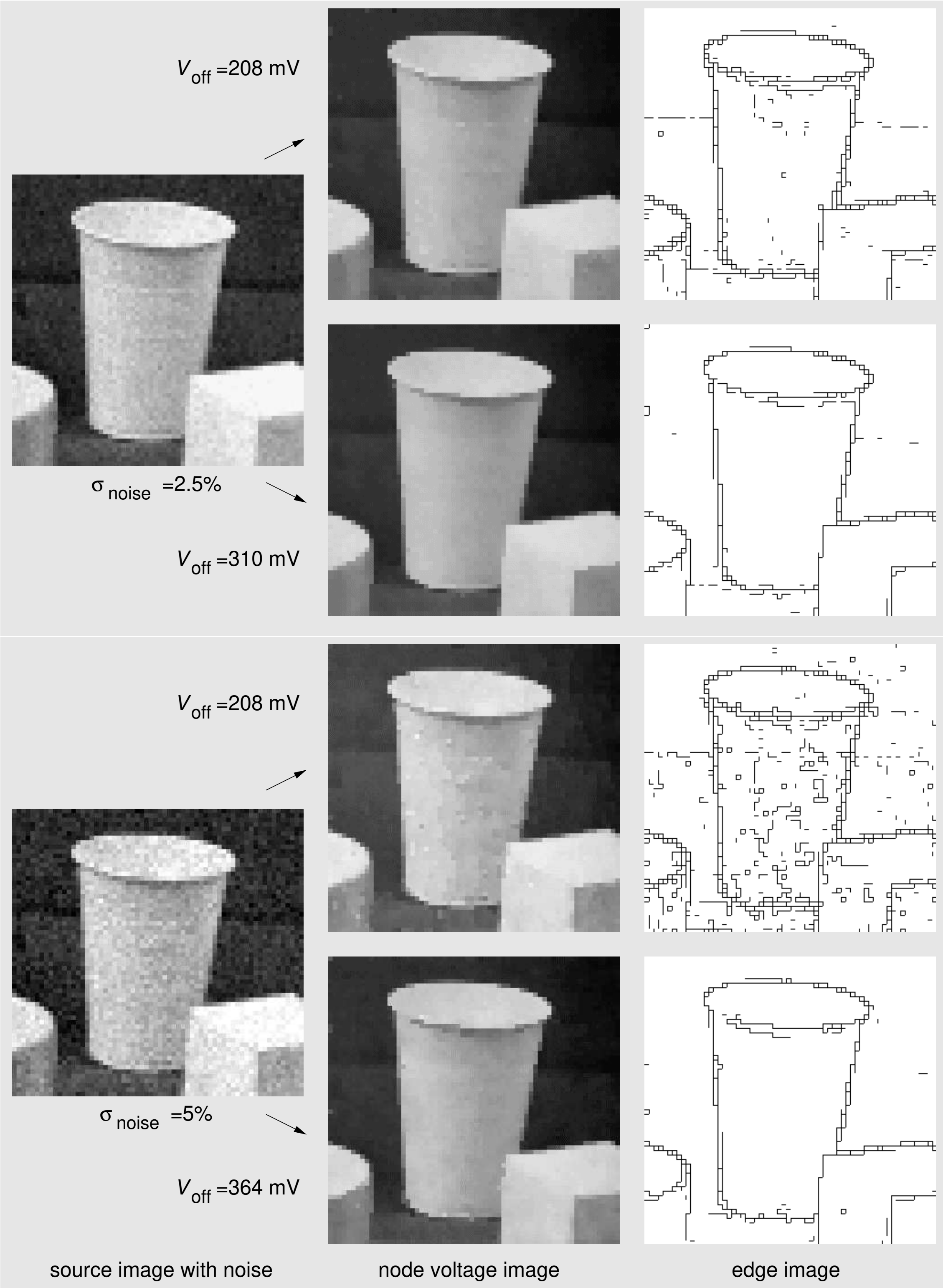



**Figure 4.22:** Noise suppression by the EDDA chip. Gaussian distributed noise was added to the input image. The width of the noise distribution is 2.5 % of the total dynamic range of the upper source image and 5 % of the lower one.

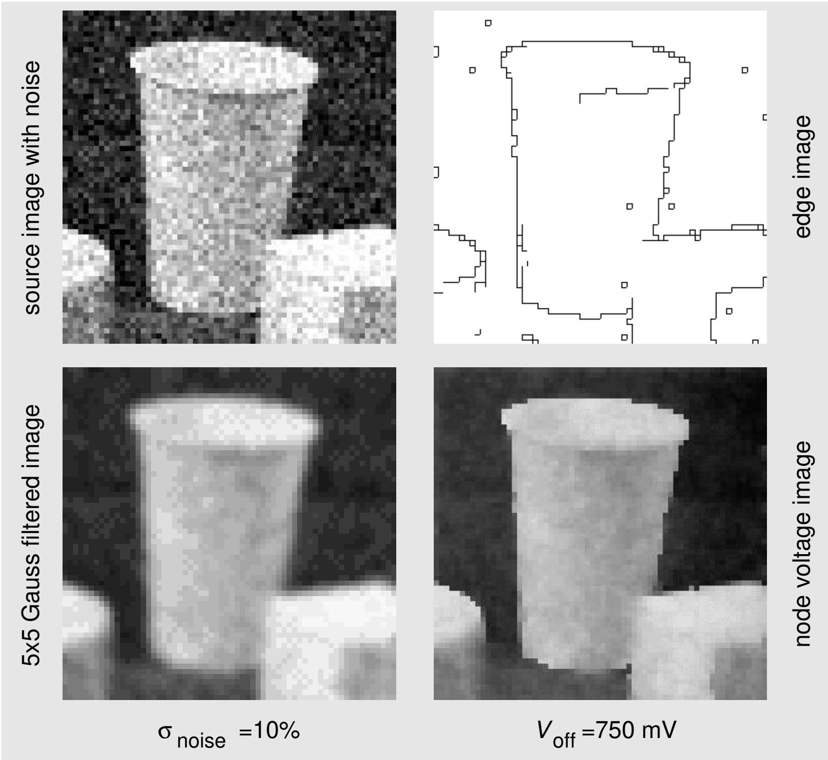



**Figure 4.23:** Noise suppression by the EDDA chip. The width of the noise distribution was 10 % of the total dynamic range of the input image. The noisy input image was also processed by a $5 \times 5$ Gaussian filter matrix for comparison.

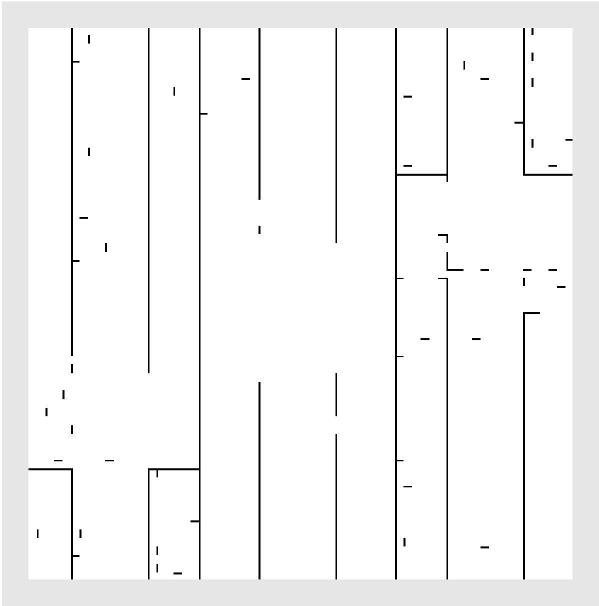


**Figure 4.24:** Edge image in response to the gray scale test image shown in figure 4.12.

## 4.4 Results From a Second Chip

A second die has been examined. Figure 4.24 shows its response to the gray scale calibration image shown in figure 4.12. The clock frequency was 15 MHz and 20 network cycles have been executed. The input voltage range was still 0.5 to 3 V. The optimum precharge voltage was 1.36 V. This is about 100 mV lower than it has been for the first chip with these parameters. The fixed pattern noise of the standard network operation was measured for 10 and 60 MHz by the method described in section 4.3.2. The results are 39.4 and 40.3 dB. This is in good accordance to the values of the first chip: 41.0 and 39.4 dB.

A series of example pictures has been processed by the second chip. It is shown in the figures 4.25 and 4.26. The clock frequency has been 15 MHz and the precharge voltage 1.43 V. $V_{\mathrm{off}}$ has been individually adjusted for each of the images.

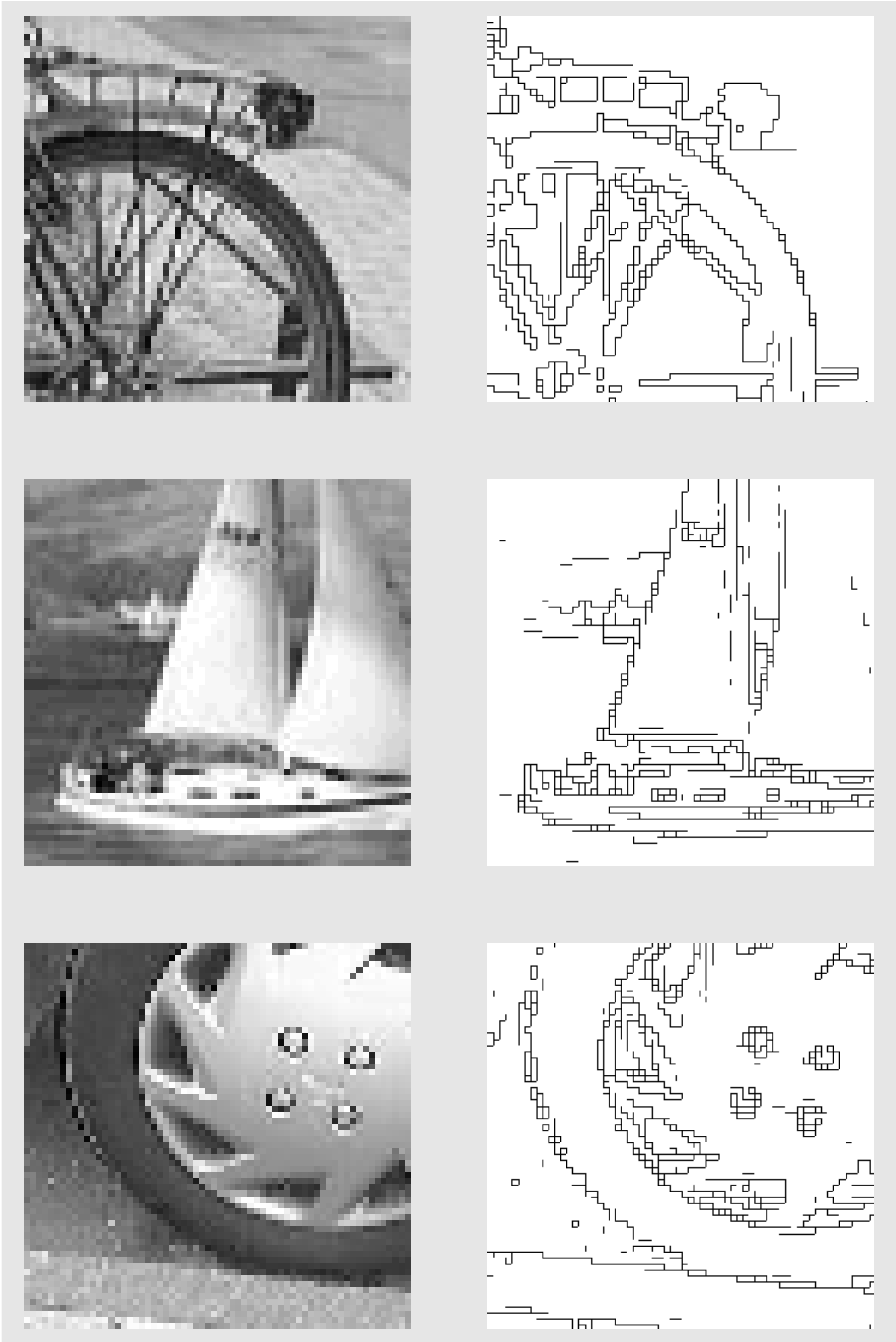


**Figure 4.25:** Example pictures. The gray scale input image is shown on the left, the edge image on the right. From top to bottom: rear wheel of a bicycle, sailing boat, rim and tire of a car.

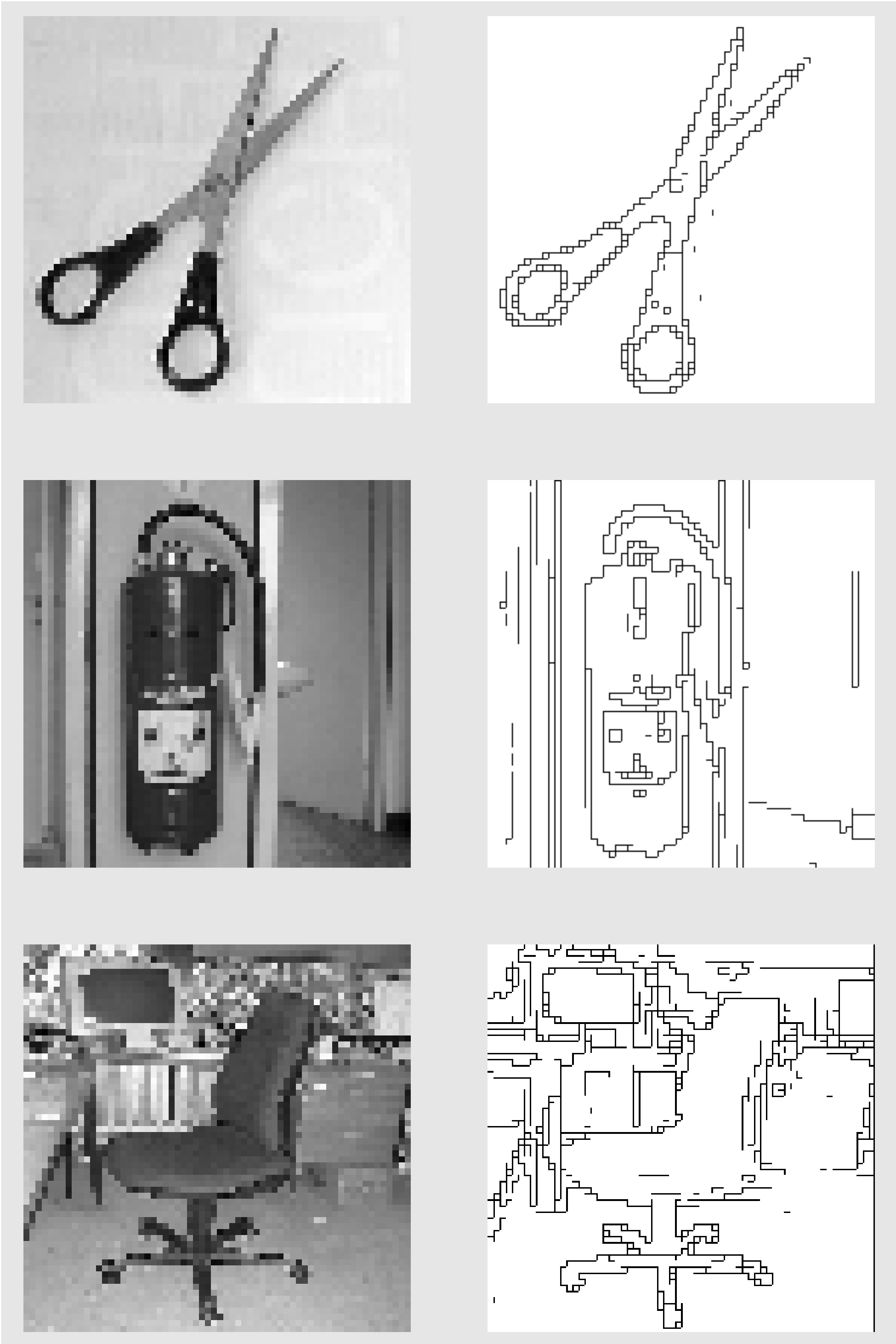


**Figure 4.26:** Example pictures continued. From top to bottom: pair of scissors, fire extinguisher, chair in front of desk.

# Discussion

Before the final EDDA chip was tested it was not clear if the fixed pattern noise would be low enough for real world applications. With a measured rms of about 7 bit it certainly can be used but a digital implementation will perform better. The advantages of the analog system are its high speed due to the fully parallel network operation and its low power consumption. A specialized digital solution at the same speed would use much more silicon area, especially if it was to be implemented in a comparable process technology. An FPGA implementation of this algorithm would be possible but it is very unlikely to be an improvement in terms of area or speed compared to a digital ASIC[5].

If the image processing system is build around a powerful CPU the edge detection can be carried out in software. Intel, for example, states an execution time of 15.8 clocks/pixel for a $5 \times 5$ Laplacian [JAE93-3] with hand-optimized assembler code using the multimedia extensions (MMX) on a Pentium II CPU [INT99]. This amounts to 300 $\mu$s for a $66 \times 66$ pixels 8 bit gray scale image with a 233 MHz CPU. The floating point version of the Laplacian needs 1.3 ms, the same time as the integer algorithm without MMX. The 'C' implementation of the resistive fuse algorithm (see section 1.4) is rather slow compared to these numbers: 160 ms for 10 network cycles. This implementation uses `float` as the data type and is not specially optimized for speed. The power consumption of a Pentium II CPU makes its usage in a portable system difficult (especially if a battery life of more than a few hours is desired).

One alternative is the StrongARM processor [SAR99]. This chip consumes only 240 mW while delivering 150 *Dhrystone 2.1 Mips*[6] in its latest revision (the SA1110). A Pentium II with 233 MHz achieves about 660 Dhrystone 2.1 Mips. An optimized Laplacian on the StrongARM CPU can be expected to last about 6 ms, 10 cycles of the EDDA simulation (translated to integer arithmetic) about 700 ms. Compared to the EDDA chip, which consumes about 125 mW at 60 MHz and needs 1.6 $\mu$s for 10 cycles, the analog solution is still $4 \cdot 10^3$ up to $4 \cdot 10^5$ times faster. The StrongArm microprocessor is manufactured in a 0.35 $\mu$m technology and contains $2.5 \cdot 10^6$ transistors while EDDA uses about 200000 transistors in a 0.6 $\mu$m process. It is possible to implement the EDDA circuit with a tenfold increase in the number of the array elements. It would also take advantage of the smaller feature size of a 0.35 $\mu$m technology since the transistors used as switches can be scaled down and the capacitance per unit area of the gate oxide is larger due to its reduced thickness. This would increase the speed advantage at least tenfold. Since the higher transconductance of the transistors allows a higher clock frequency it could be even more.

The actual version of the logarithmic CMOS camera (see introduction) delivers $384 \times 288$ pixels, 25 times the EDDA resolution. A microprocessor solution based on the StrongARM could

[5] Application Specific Integrated Circuit

[6] The Dhrystone Mips are defined as the result of the Dhrystone benchmark program divided by the VAX 11/780's Dhrystone value of 1757. The most recent version number of the Dhrystone benchmark is 2.1 [DHR99].

process a maximum of 6.6 images/s (based on the estimation of the Laplacian's execution time). This means that the CPU subsystem is totally occupied by the edge detection in a real time application. So even at a moderate resolution a low-power CPU is not fast enough to do the edge detection in software together with the other tasks that the main CPU of a portable image processing system should perform, for example the further processing of the edge data and the formatting of the results for an output device. A dedicated image processing front end is needed in any case. This makes the analog solution very attractive, since it is fast enough, does not add much to the power budget (since this is most likely dominated by the system CPU) and could make an A/D converter redundant.

To use the speed advantage of the EDDA chip the image I/O must be able to cope with the speed of the resistive fuse array. This is not the case for the prototype presented in this thesis. The input data rate is limited to about 10 MHz and the edge readout rate to 60 MHz (if a sufficiently fast FPGA is used for the EDDA bus interface). With these limitations - and considering the overhead for multiplexing the larger camera image onto the resistive fuse array by assuming an overlap between the image parts of 5 pixels - the actual EDDA chip can process 70 images/s. The I/O overhead is nearly 30000 %. There were mainly two reasons why no faster I/O interface has been integrated: first, it seemed not sensible to invest time and money in those circuits as long as it was not known if the analog core would work reasonably. Second, an I/O interface should be adapted to the circuits it has to communicate with. Those were not specified at the time the EDDA prototype was developed. Only the camera design was available, and its analog output rate is limited to 8 MHz. Therefore the EDDA core has been designed with high speed in mind (fast digital readout, high nominal clock speed, parallel I/O) while the external interface was not optimized for speed, but rather for simplicity and testability.

With the results of the measurements at 60 MHz in mind (see section 4.3.4) the EDDA chip seems to be better suited for very high speed applications (more than 1000 frames/s) than for the medium speed computer vision tasks (around 50 frames/s) it has been originally designed for. This turned out during the design process and was considered as a very interesting operational area for analog VLSI systems, since it is occupied by high-end (and high power consumption) multi-chip digital systems at the present time. There are two possibilities to create an edge detection system able to run at frame rates in the range of 100 kHz with the EDDA core:

- The transfer of the analog input data is done from both sides of the array at the same time by several double buffered high-speed multiplexers. With four inputs at 100 MHz the time to load an analog image would be decreased to 10 $\mu$s.
- Active photo sensors in CMOS technology are directly integrated into the array. The actually implemented analog storage could be reused to do an offset calibration for the individual image sensors. At sufficiently high illumination levels frame rates of 1 MHz should be possible. Obviously this solution is more difficult to implement than the one mentioned above. Also the resolution is limited by the resolution of the EDDA array while the usage of a high speed camera with freely addressable pixels still allows to map different parts of a larger image onto the resistive fuse array.

The output could be enhanced much easier. Since the digital edge data is stored in the row readout circuits in parallel any fast digital interface can be put next to the row readout. The most simple solution would be to widen the peripheral bus of EDDA to 32 or 64 bits without changing anything else. Readout speeds of 3 $\mu$s are possible with a sequencer clock frequency of 60 MHz. Another interesting approach is to put a certain degree of logic next to the row readout to process the edge data on-chip. The detection of movements or the vectorization of the edge image are possible applications. The evaluation of the EDDA chip has shown that the analog SC circuits tolerate the noise generated by the high speed digital circuits. Therefore the integration of a digital image processing logic or more complex bus interfaces (for example high-speed serial or PCI) should be possible.

At the time the EDDA chip was designed, the noise suppression capability was not realized. Therefore the analog node voltage readout was only implemented for evaluation of the chip. It is slow (about 2 MHz maximum) compared to the analog input or the edge readout and adds fixed pattern noise at a level of about -45 dB to the analog signal. It is possible to implement a correlated double sampling technique by connecting the source follower input to a reference potential (for example the pixel bias voltage) before the node capacitor is read out. The difference between both measurements is then used as the output signal, thus suppressing the offset variations. Also the speed can be improved by the usage of row-wise readout amplifiers and two banks of sample-and-hold stages. This would increase the analog output data rate by a factor of 64 (the number of rows to be read out) without changing the individual readout current since all the source followers of one column are used in parallel.

The example pictures of the previous chapter have been processed at a constant $V_{\mathrm{off}}$ level[7]. In this case the resistive fuse network reaches an equilibrium state which does not change anymore by further network cycles. It is possible that a time-varying $V_{\mathrm{off}}$ progression - where the network does not reach an equilibrium - improves the edge detection. This may be worthwhile to examine with a heuristic method like a genetic algorithm. The EDDA chip is a good candidate for the optimization of hardware parameters by such a method since a certain setup can be quickly evaluated, allowing a large number of iterations of the algorithm per second. Also the analysis of image sequences without clearing the node capacitors between the individual images may reveal interesting results, since the edge pattern of the previous image influences the actual one.

The analog precision will be improved if the common mode voltage dependence of the fuse's $V_{\mathrm{off}}$ input can be reduced. This should be investigated in further revisions of the EDDA chip. By combining the edge data at different $V_{\mathrm{off}}$ levels it should be also possible to enhance the quality of the edge information.

The EDDA chip shows that it is possible to do highly parallel analog computation in a way it can compete with digital solutions in terms of power consumption and speed. It is important that the methods used in the analog circuits make use of the advantages of modern sub-micron process technologies. Therefore it benefits in the same way from smaller feature sizes than digital circuits do. This is not the case for the majority of the continuous time analog implementations that rely more on the transistors' characteristic curves. Obviously the EDDA chip is not as accurate as a digital implementation might be and there are only limited possibilities for trading speed against precision, which can be easily done in a digital circuit. Most promising for future work seems to be the further optimization of the EDDA chip for high-speed, low-power applications since it is unlikely that a digital concept will perform better in this area. It may be a useful building block in an active vision system for mobile robots or - with integrated photo sensors - for fast positional measurement systems.

[7]The node capacitor voltages have been equalized by the subthreshold currents before.

# Appendix A

# RAM

The static RAM that stores the sequencer program consists of the following functional blocks, shown in figure A.1:

- The bit cell array that contains 1344 cells, organized as 42 columns of 32 bits each.
- The address decoder to select one of the 42 columns
- The read/write driver circuit that combines the global read and write signals with the output of the address decoder to activate the read or write line of the selected column.
- The input/output multiplexers and drivers array that contains multiplexers to reduce the output bus width from 32 to 16 bits and to route the 16 bit input data to the appropriate bit lines of the array.

The total area occupied by these circuits is 0.33 mm$^2$. The RAM is optimized for a minimum address-to-data delay. It must be less than 2 ns to ensure the correct sequencer operation at 100 MHz. To achieve a minimum column to output delay each bit cell contains an own bit line driver. Figure A.2 shows the schematic diagram of one bit cell. The storage latch is build by the inverters I1 and I2. I3 is the readout driver, enabled by the column read signals (the inverted read signal is not shown in the block diagram). A single NMOS pass transistor M1 serves as the write gate. The transistor sizes of I2 are selected in a way that its NMOS is particularly weak. This will be necessary if a logical high should be stored in the cell. M1 must be stronger, i.e. its on-resistance must be lower, than the NMOS of I2. But while the gate-source voltage of I2 is equal to $V_{\mathrm{DD}}$ unless I1 switches it gets reduced for M1 with the increasing voltage at the input of I1. The gate-source voltage of M1 is reduced by 50 %, increasing its on-resistance approximately by a factor of 4. This is reflected by the W/L ratios of both transistors to make sure that a high bit can be reliably written to the cell. The on-resistance of M1 is less than 20 % of that of I2's NMOS transistor. This also includes a security margin to compensate for the transistor mismatch.

I3 is implemented as clocked inverter. It is able to drive the bit line and the buffer in the output multiplexer in less than 1 ns. The output multiplexer consists of two clocked inverters, selecting the even or odd Q output of the ram array according to address bit 0. Both outputs are connected to a common driver, implemented as an inverter with fourfold drive strength. Thus the RAM output can be directly routed into the standard cell logic without any additional buffers. The write multiplexer uses transmission gates to connect the input data bus to the even or odd input lines of the bit cell array.

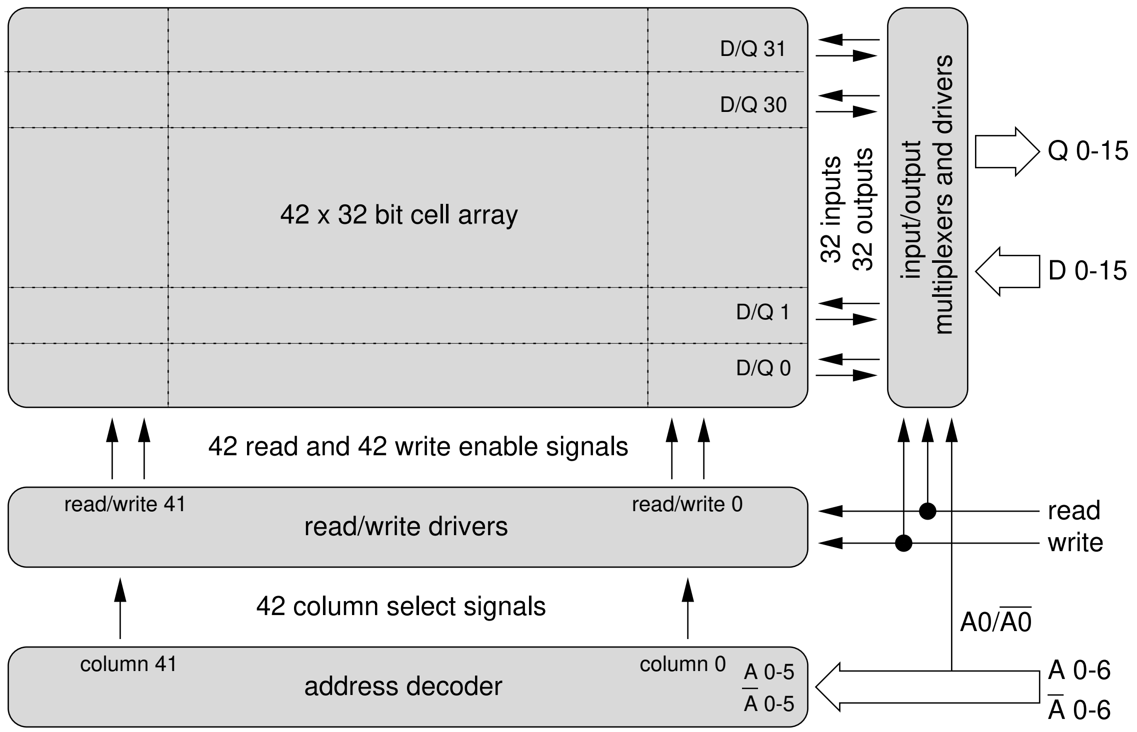



**Figure A.1:** Block diagram of the RAM.

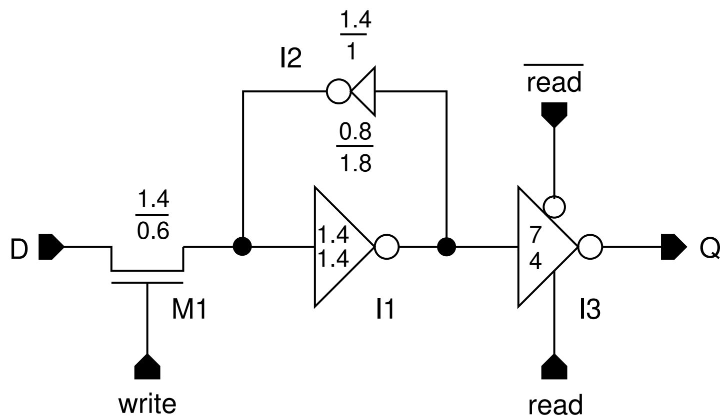



**Figure A.2:** Schematic diagram of the bit cell. For I2 the W/L ratios of both its PMOS (above) and NMOS (below) transistors are given.

Since the date hold inverter I2 is very weak the bit cell flips if M1 is activated and the input data line is floating at the opposite voltage level. Therefore the input lines that are not used are connected to their respective outputs by transmission gates located in the input multiplexer during a write access.

The address decoder needs complementary address inputs. Each columns select consists of a six input NAND gate. Its inputs are connected to the combination of low and high-active address lines that corresponds to the column address. For example column 25 uses the pattern $\overline{A5}$-A4-A3-$\overline{A2}$-$\overline{A1}$-A0. The transistor widths are chosen in a way that the column decoder delay is nearly symmetric to ensures that the last column is as fast turned off as the new one is activated.

The layout plot of the RAM is shown in figure A.3. An enlarged cutout of this plot is displayed in figure A.4. One bit cell is marked by the dashed rectangle. To save area the odd rows and columns are mirrored along their X and Y-axis respectively. By this way some of the metal traces and the substrate and n-well contacts can be shared between neighboring cells[1]. The substrate contacts have a separate ground net to avoid noise injection into the substrate.

[1]The digital RAM circuit does not suffer from the increased device mismatch caused by this symmetry operations since the signals from one cell are not related to those from others. The device variations caused by rotating and mirroring are not large enough to change the logic level thresholds of the cells significantly.

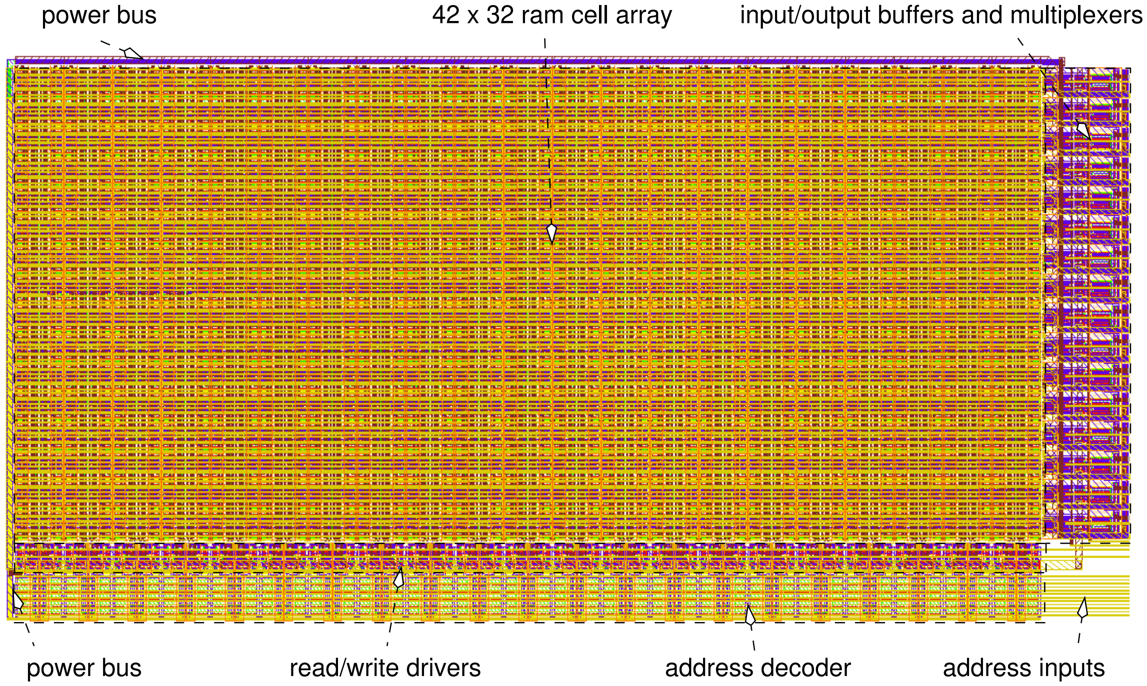



**Figure A.3:** Layout plot of the RAM. The area shown is $855 \times 385\ \mu\mathrm{m}^2$.

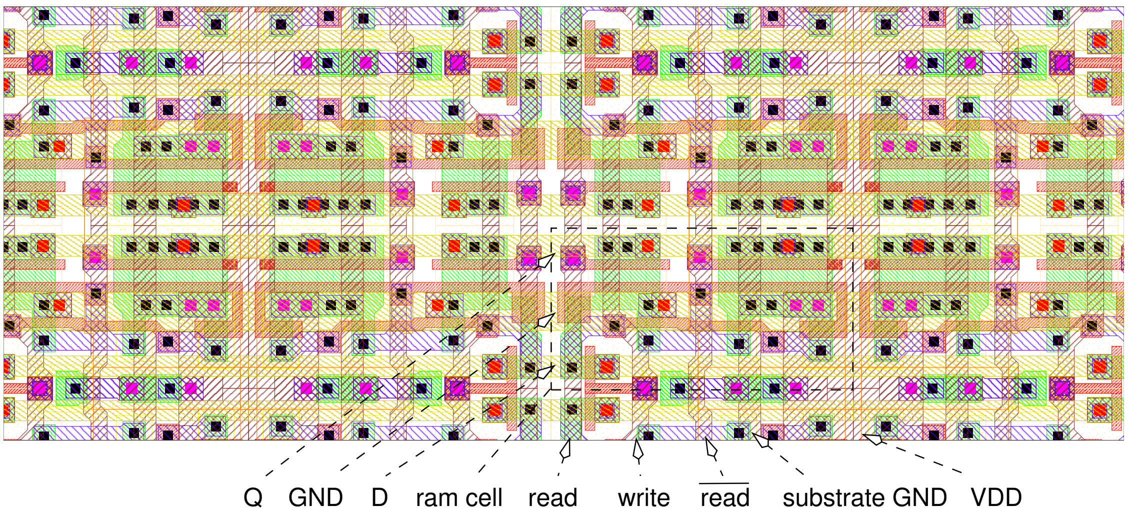



**Figure A.4:** Cutout of the bit cell array. One cell measures $18.6 \times 10\ \mu\mathrm{m}^2$. For a layer legend see appendix E.

# Appendix B

# Phase Locked Loop

This appendix covers the phase-locked loop (PLL) circuit that generates the sequencer clock. Its design is based on the 'classical digital PLL' as it is explained in [BES97]. The different building blocks have been redesigned to be suited for VLSI integration. The PLL needs no external components and operates from the same single 5 V supply as the rest of the EDDA chip does. For reasons of power supply noise suppression it also utilizes different bond pads for the analog and the digital circuits[1]. Figure B.1 shows a block diagram of the PLL. The arrows depict the flow of control information in the loop. Solid arrowheads are digital control signals, i.e. the information is coded in the frequency or the pulse width, while the hollow ones represent analog signals where the information is coded in the current or the voltage. The different parts of the control loop have the following functions:

**Voltage controlled oscillator (VCO)** The VCO generates the high frequency clock signal $f_{\text{osc}}$. The output frequency is controlled by the control voltage $V_{\text{osc}}$.

**VCO divider** $f_{\text{osc}}$ is divided by the clock multiplier of the PLL. It can be adjusted to 2, 3 or 4. The VCO divider also contains circuitry to ensure that the phase of the divided bus clock and the sequencer clock is equal.

**Phase-frequency detector (PFD)** The PFD compares the phase of the bus clock with the external clock and generates error signals, called *up* and *down* depending on the direction of the required frequency correction, if the phase of the bus clock lags or precedes the external clock. The length of the pulses is proportional to the phase difference.

**charge pump** The *up* and *down* signals are converted to current pulses of opposite direction by the charge pump.

**Loop filter** The current pulses of the charge pump are integrated by the loop filter to produce the $V_{\text{osc}}$ control voltage for the VCO. The filter capacitor and the filter resistor can be digitally adjusted (this is denoted by the diagonal arrows).

The clock buffers are not part of the PLL loop but they are integrated in the PLL circuitry. The *bypass* input switches between the PLL generated clock signals and the external clock input. The switches are built from two clocked inverters with triple drive strength while the main clock buffers

[1] It shares these bond pads with the analog and digital power supply nets of the other EDDA circuits.

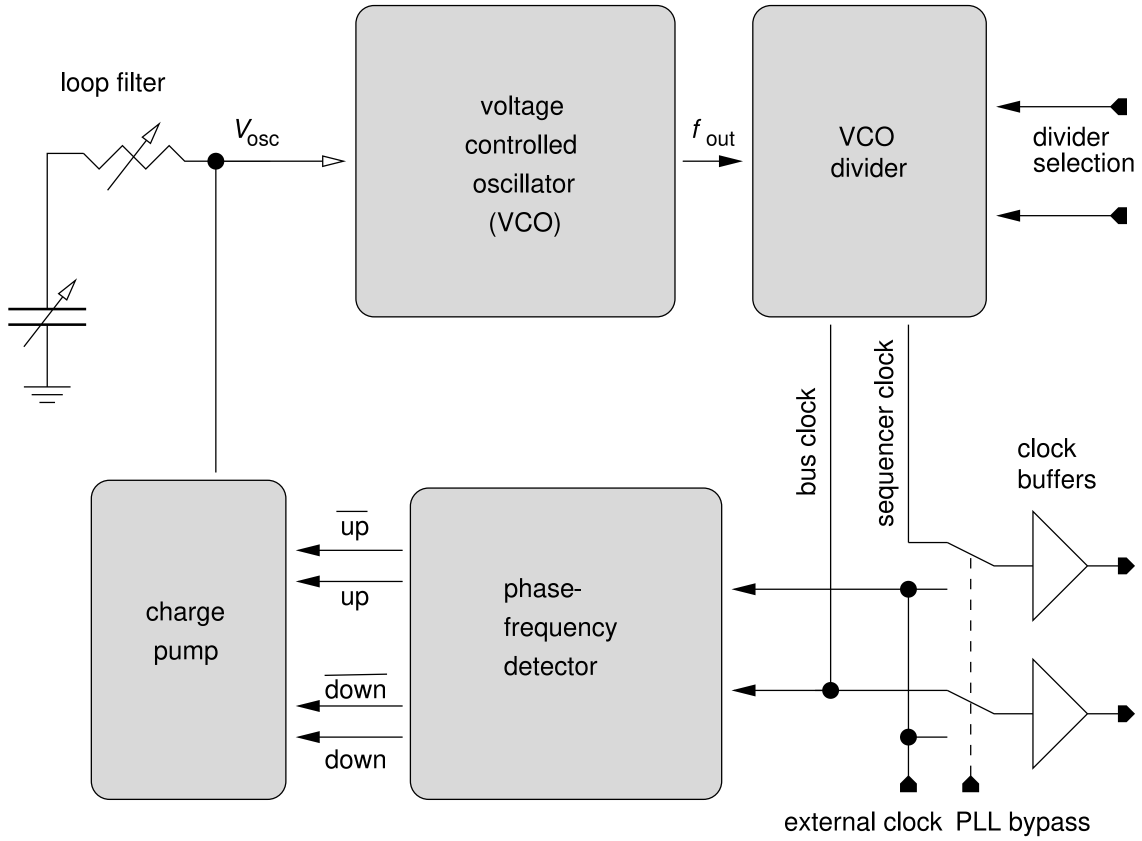



**Figure B.1:** Block diagram of the PLL.

are inverters with 24fold drive capability. They drive the global sequencer and bus clock lines. The digital circuits, i.e. the VCO divider and the PFD, are implemented with the AMS standard cells while the analog circuits are full custom designs.

## VCO

Figure B.2 shows the schematic diagram of the VCO. It consists of two parts: the voltage controlled oscillator itself (upper half) and the output buffer that converts the differential analog signal from the VCO to a digital one with a 50 % duty cycle (lower half). The VCO is a three stage ring oscillator. Each stage consists of an operational transconductance amplifier with its differential output connected to the input of the adjacent stage. Between stage 3 and stage 1 the signal is inverted by swapping the differential output signals of stage 3 before connecting them to stage 1. This makes the circuit oscillate because the output signal is fed back to the input with a 180° phase rotation. The speed of each stage depends only on the bias voltage of the current source transistor since the output load is equal to the constant input capacitance of the next stage. Figure B.3 shows a Spectre simulation of the ring oscillator at $V_{\mathrm{osc}} = 3.5$ V. It can be seen that the phase shift between the stages is 60°, compensating together the external shift of 180°. The period of the oscillation is 5.3 ns and the amplitude ranges from 1 to 1.2 V. At 100 MHz $V_{\mathrm{osc}}$ is 1.7 V and the amplitude is reduced to 400 mV. At this frequency each stage draws 70 $\mu$A. The lower limit

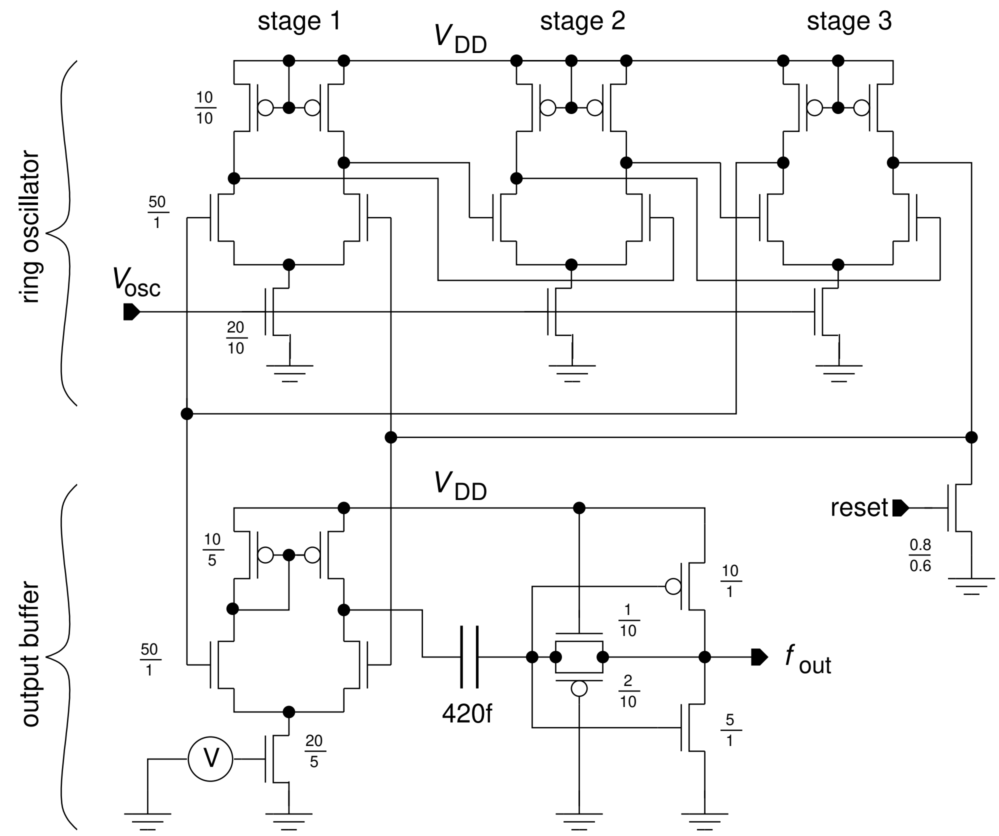



**Figure B.2:** Schematic diagram of the VCO and the VCO output buffer.

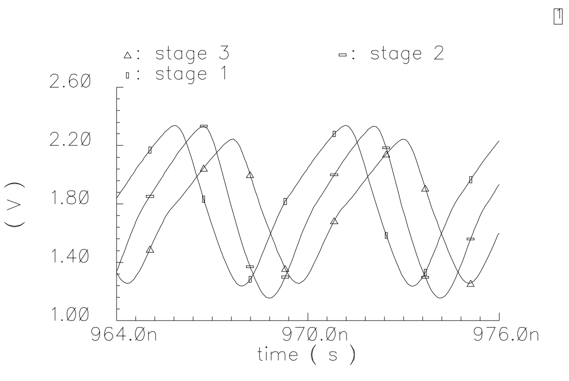



**Figure B.3:** Transient simulation of the outputs of the different stages of the ring oscillator at $V_{\text{osc}} = 3.5$ V.

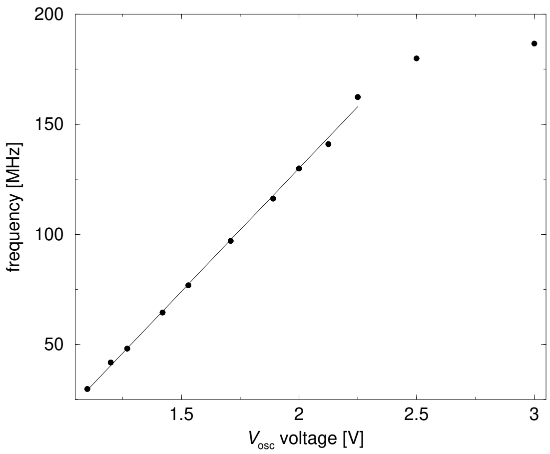



**Figure B.4:** Simulation of the voltage-frequency dependence of the oscillator.

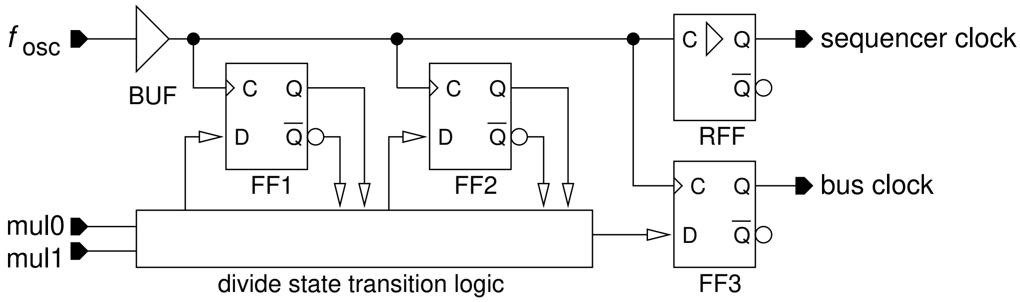



**Figure B.5:** Simplified schematic diagram of the VCO divider.

of the oscillator is around 30 MHz. The amplitude is below 100 mV at this frequency. If $V_{osc}$ is reduced further the output buffer will not be able to restore the oscillator signal to the full 5 V level any more. The frequency is proportional to the $V_{off}$ voltage until the upper frequency limit is reached. Figure B.4 shows the voltage-frequency curve taken from simulation data at different $V_{osc}$ levels. The result of a linear regression is shown as solid line for the data points below 2.5 V. The frequency gain of the oscillator is 112 MHz/V. The reset transistor was necessary for the simulation since the algorithm used starts the transient analysis from the dc operating point values. This is just the unstable equilibrium point of the oscillator circuit. Since any amount of noise destroys the equilibrium condition this problem is not present in the real circuit but the reset transistor was kept for testing purposes.

The output buffer restores the low voltage differential output signal of the ring oscillator to the full voltage swing of 5 V necessary to drive the following digital stages. It consists of a two stage differential amplifier. The first stage is an ordinary differential input stage with a current-mirror load, converting the differential oscillator signal to a single-ended one. It is constantly biased because the ring oscillator has the advantageous property that its output amplitude rises with increasing frequency, thus compensating for the constant gain-bandwidth product of the output buffer. The second stage is a high gain inverting amplifier that is AC-coupled to the first stage to remove any common-mode components from the oscillator signal. It is biased by a high-ohmic feedback build from two long transistors used in their ohmic region if no input signal is applied to the amplifier. The width ratio of the PMOS to the NMOS is similar to that used in the digital standard cell library. Together with the feedback this ensures that the amplifier is symmetric and the output signal has a 50 % duty cycle across the whole frequency range of the oscillator. The coupling capacitor is built from a metal1-2-3 sandwich structure since it must be linear at small voltages. This can not be achieved by a gate-channel capacitance. The whole circuit uses only 800 $\mu$A at 100 MHz. This is an advantage of the ring oscillator compared to an inverter delay chain that uses the full voltage swing in each stage.

## VCO divider

The high frequency output signal of the VCO must be divided by the desired PLL clock multiplier to get the reference signal for the phase detector. In the VCO divider this is done with the synchronous state machine shown in figure B.5. The state bits are stored in FF1 and FF2. The state transition logic generates the different clock sequences depending on the state of the two input lines mul0 and mul1 which select the multiplier. It also produces the output signal for the bus clock flipflop

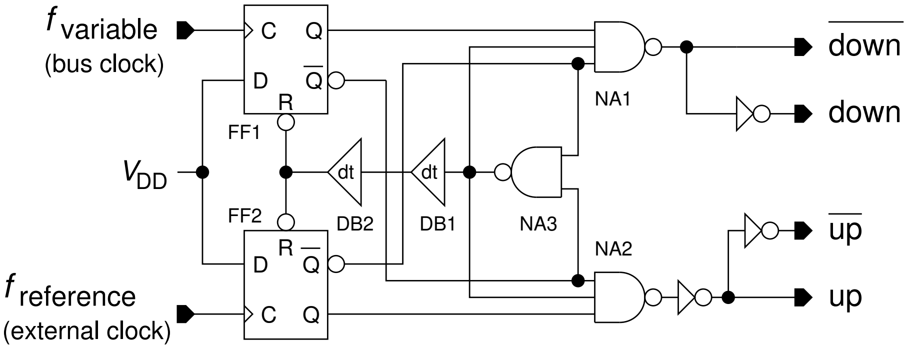



**Figure B.6:** Schematic diagram of the phase-frequency detector.

FF3. To make sure that the phase of the sequencer clock is equal to that of the bus clock a replica flipflop (RFF) has been built by modifying the flipflop standard cell used for FF3 in a way that the clock-to-output delay stays the same but the output signal mirrors the clock input. It has no data input any more. The simulated delay between the bus and the sequencer clock output was 28 ps for the rising edge (which is the active one) and 230 ps for the falling one. The purpose of the input buffer BUF is to increase the slope of $f_{\text{osc}}$ to minimize the clock jitter between FF3 and RFF and to reduce the power consumption of all the flipflops.

### Phase-frequency detector

The most commonly used phase detector of a digital PLL is the phase-frequency detector (PFD) shown in figure B.6. In contrast to the EXOR or the edge-triggered JK-flipflop the PFD does not only respond to the phase but also to the frequency error between its input signals. Therefore it offers an unlimited pull-in range [BES97-2]. This is very beneficial in the case of the EDDA chip where it should be possible to change the input frequency at any time[2]. The PFD has three different output states: inactive, up and down (inactive means that neither *up* nor *down* are active). If the rising edge of the variable clock input is seen and the outputs are both inactive, *down* is activated. If a rising edge occurs on the reference input and *down* is already active, the NAND gate NA3, connected to the $\overline{\text{Q}}$ outputs of FF1 and FF2, will reset both flipflops. If the reference signal is seen with *down* inactive, *up* is activated until the flipflops are reset by a rising edge of the variable clock input. The triple input NAND gates NA1 and NA2 ensure that either *up* or *down* are active, but not both.

The delay buffers DB1 and DB2 make sure that the duration of the reset pulse is long enough for the flipflops. The connection from the output of NA3 to NA1 and NA2 also exists for timing reasons only. It inhibits the *up* and *down* outputs for the delay of NA3 after the flipflops have been reseted. This avoids short spikes at an output if an edge and the reset pulse arrive at the according flipflop at the same time. To avoid glitches if both edges are synchronous the Q outputs, which are the slower ones, are used for activation and the $\overline{\text{Q}}$ outputs for inhibition. Also the three inputs of the NAND gate have slightly different input-to-output delays and the Q output is connected to

[2]Its drawback is that it does not allow missing edges in the reference signal like the EXOR or the JK-flipflop do. This forbids its usage in data recovery applications.

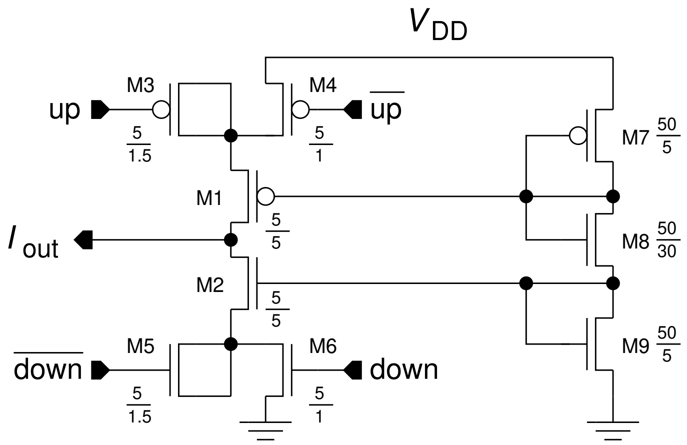



**Figure B.7:** Schematic diagram of the charge pump.

the slowest while $\overline{\mathrm{Q}}$ is connected to the fastest one. This ensures that the output of the PFD is glitch-free while maintaining a dead-time below ±300 ps[3] and using only standard cells[4].

## Charge pump and loop filter

The charge pump must transform the voltage pulses from the PFD into current ones. The minimum pulse length generated by the PFD can be below 200 ps. This leads to the problem that the charge transferred via the gate-drain capacitance of a transistor quickly becomes larger than that flowing through the channel while it is switched on. To compensate this charge injection dummy transistors are used. Figure B.7 shows the schematic diagram of the charge pump. The current source transistors M1 and M2 are enabled by M4 and M6. To avoid a high current pulse in the moment they are switched on the dummy transistors M3 and M5 are activated a little bit earlier. This is ensured in the PFD by always generating the signal for M4 and M6 by inverting that of M3 and M5. They are also 50 % wider. Since both their drain and source regions are connected to the sources of M3 and M5 respectively their charge injection is approximately three times higher. Figure B.8 shows an example of a short charge pulse at the output of the charge pump (connected to the RC loop filter). The delay between the reference and the variable clock was 500 ps. The charge transferred with this pulse is $Q = It = 6.5\mu\text{A}\cdot 300\text{ ps} = 2\cdot 10^{-15}\text{C}$ (using a rectangle approximation) compared to the charge injection of M4: $Q = VC = 5\text{V}\cdot 14\text{ fF} = 7\cdot 10^{-15}\text{C}$. This shows that the compensation scheme works very well. The current is controlled by the bias transistors M7 to M9. M7 and M1 as well as M2 and M9 form current mirrors. Therefore the current is the same for up and down pulses despite the fact that the NMOS and the PMOS transistors have equal sizes. This is necessary to make sure that the capacitances are the same for the up and down path to keep the charge pump symmetric. The nominal current is about 6 $\mu$A (with the 'typical mean' simulation parameters for the devices).

The loop filter is a simple RC filter as depicted in figure B.1. The effective size of the capacitor can be selected by digital switches between 0 and 49 pF in 7 pF steps. The capacitors are realized

[3] The maximum time interval between the two input signals that does not produce an output pulse.

[4] With the exception of the delay buffers. They are modified standard cell buffers made slower by lengthening the transistors channels.

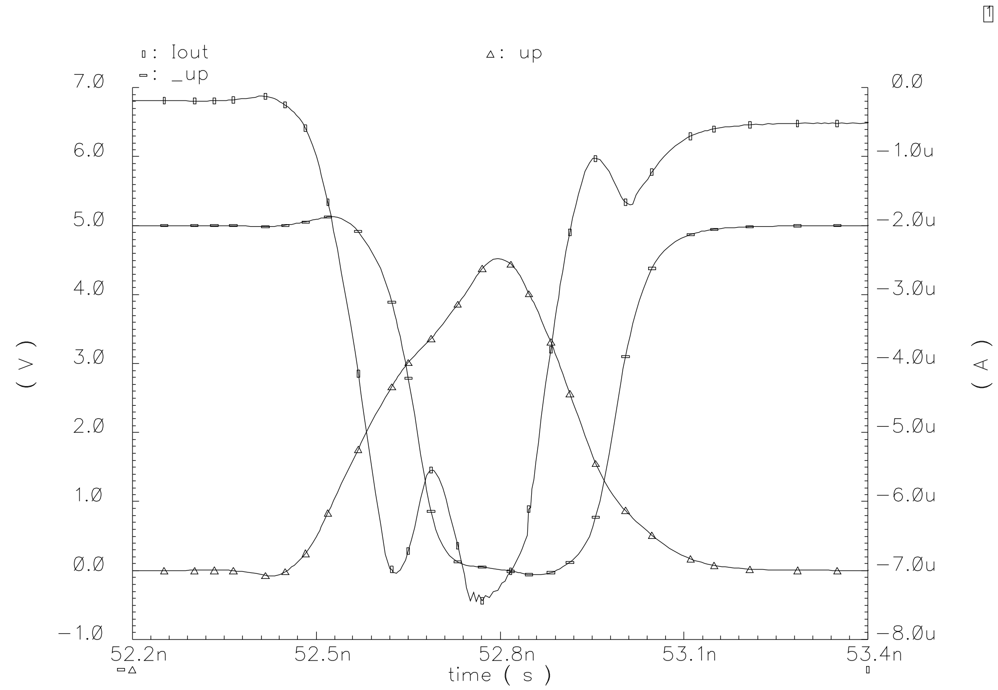



**Figure B.8:** Example charge pulse $I_{\text{out}}$ generated by the charge pump.

as PMOS transistors. The resistor is also digital adjustable. Its resistance ranges from 0 to 60 kΩ (in 4 kΩ steps). It is implemented by n-well resistors. The gate signals of the switches for the capacitors and resistors are buffered by inverters to suppress the digital noise that might be on the select lines coming from the digital control section of EDDA. By changing the RC time constant of the filter it is possible to compensate device variations, especially in the bias circuit of the charge pump.

## Layout

Figure B.9 shows a layout plot of the PLL circuit. The area used is only 0.07 mm$^2$. The digital part can be seen on the right side, enclosed by a wide guard ring. The loop filter capacitors occupy the largest area of the analog part. The metal sandwich capacitor between the stages of the VCO output buffer uses also a significant amount of the area (due to the small specific capacitance per area between the metal layers).

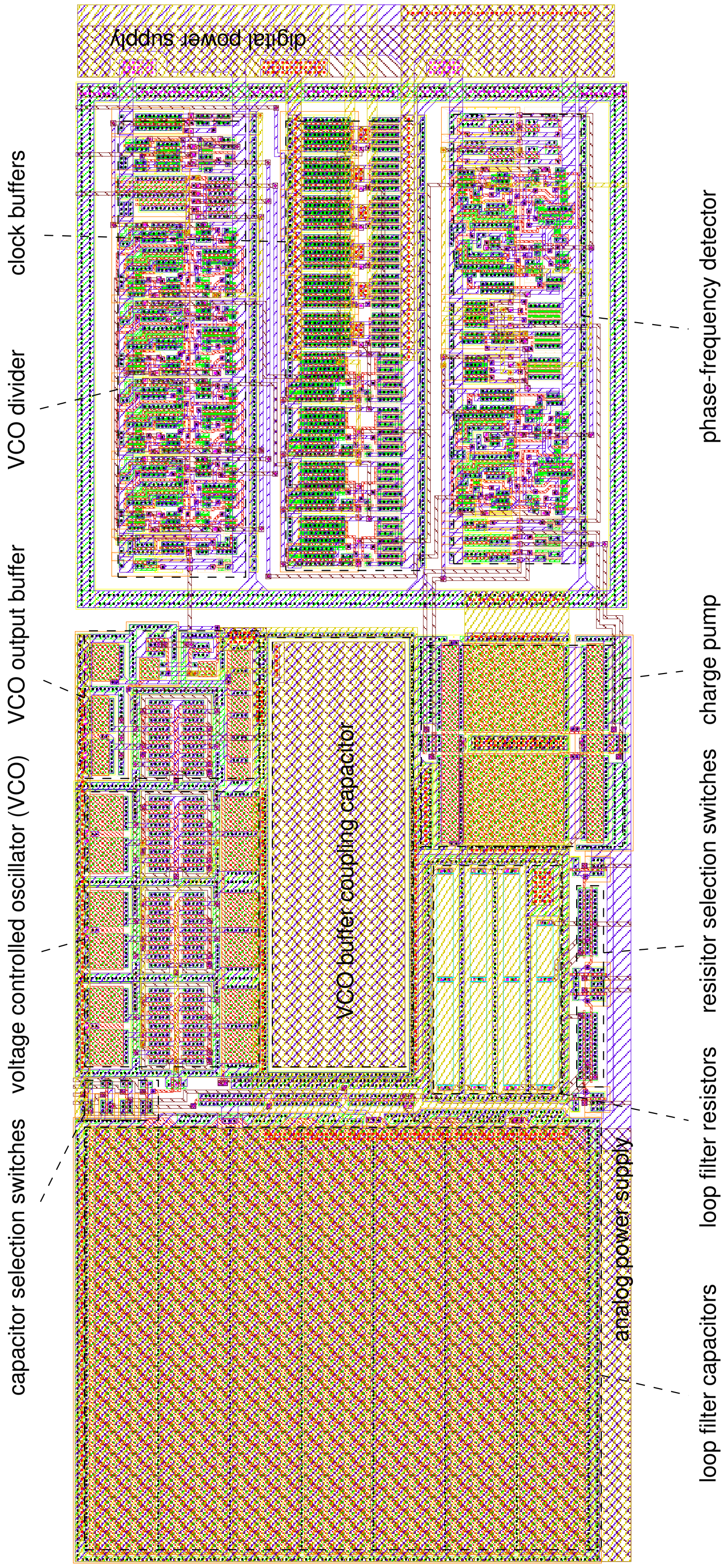



**Figure B.9:** Layout plot of the PLL. The area shown is $435 \times 168\ \mu m^2$.

# Appendix C

# Programming the EDDA-chip

## C.1 Registers

The bus interface unit of the EDDA chip decodes the following addresses:

| address(hex) | r/w | width | name | function |
|---:|---|---|---|---|
| 0 | rw | 5 | CR | control register |
| 1 | rw | 7 | PLLFILTER | PLL loop filter |
| 2 | rw | 4 | PLLCR | PLL control register |
| 3 | rw | 8 | DELAY | non-overlapped delay control |
| 4 | rw | 8 | ENCNT | edge readout cluster select |
| 5 | r | 8 | EDOUT | edge readout data |
| 6 | r | 7 | CNTOUT | loop counter 0 |
| 7 | r | 8 | RADR | sequencer address counter |
| 80 | rw | 16 | RAM0 | RAM word 0 |
| ... | | | | |
| D3 | rw | 16 | RAM83 | RAM word 83 |

Since the width of the RAM exceeds the 8 bit width of the peripheral bus of EDDA two subsequent data cycles are used for writing or reading a word of the RAM. Each address cycle resets the byte count to the low-byte. The first data cycle within the RAM address range sets it to the high-byte. The second data cycle sets it back to the low-byte. The ram address transferred by an address cycle is stored in the sequencer address counter register.

### Register definitions

**Control register** (CR, address 0):

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| - | - | - | strobe | sig | ainc | de | ae |

Reset value: 0

**ae** The address enable bit allows the sequencer instruction decoder to update the address counter. Usually this bit is used to start the sequencer.

**de** The data enable bit allows the sequencer instruction decoder to update any output register. A sequencer program can be tested by enabling ae without de. By enabling de without ae it is possible to directly write to an output register through the RAM by setting the sequencer address to any ram address and writing the desired command to that address. This immediately executes the command without changing the sequencer address.

**ainc** Automatically increments the address counter after any read/write access to it. Speeds up program loading by 50-%.

**sig** Handshake signal to the sequencer. Can be included in conditional jump instructions.

**strobe** The strobe bit can be only written. After a one has been written to it, the sig bit is inverted for one bus clock cycle. This allows strobed signals with both logic levels.

**PLL loop filter register** (PLLFILTER, address 1):

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| - | csel 2 | csel 1 | csel 0 | rsel 3 | rsel 2 | rsel 1 | rsel 0 |

Reset value: csel=4, rsel=8

**csel 2-0** Capacitance selection of the PLL loop filter. C=csel·7pF.

**rsel 2-0** Resistance selection of the PLL loop filter. R=rsel·4kΩ.

**PLL control register** (PLLCR, address 2):

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | pllres | pllbypass | pllmul 1 | pllmul 0 |

Reset value: pllres=0, pllbypass=1, pllmul=2

**pllmul 1-0** PLL multiplier. 1 and 2 both select a multiplier of 2, 3 equals to 3 and 0 selects 4.

**pllbypass** Disables PLL.

**pllres** PLL VCO reset.

Both PLL registers have a special reset mechanism. If one of the bus mode inputs of EDDA is held low while a reset signal is applied the registers are not set to their default values but maintain their current contents instead.

**Non-overlapped delay control register** (DELAY, address 3):

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| selaux1 1 | selaux1 0 | selaux0 1 | selaux0 0 | deld 3 | deld 2 | deld 1 | deld 0 |

Reset value: selaux1=0, selaux0=0, deld=4

**deld 3-0** Non-overlapped clock delay.

**selaux0 1-0** Auxiliary output 0 selection:

| selaux0 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| signal | off(0) | clk1 | bus clock | ss 7 |

**selaux1 1-0** Auxiliary output 1 selection:

| selaux1 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| signal | off(0) | clk2 | sequencer clock | ss 8 |

The auxiliary outputs can be used as programmable handshake outputs if they are set to output ss 7 or ss 8 respectively. These are bits in the strobe signal register (see next section). They can also be programmed to monitor the PLL outputs (bus clock and sequencer clock) or the non-overlapped array clock signals clk1 and clk2.

**Edge readout cluster select register** (ENCNT, address 4):

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| rsout | csout | ss 7 | ss 6 | encnt 3 | encnt 2 | encnt 1 | encnt 0 |

Reset value: encnt=8

**encnt 3-0** Output enable counter for the edge readout cluster selection. It can be read and written directly but is is also automatically updated after one byte of the edge data has been read out. In this case it automatically wraps around from 7 to 0. The values from 8 to 15 disable all the clusters.

**ss 7-6** Read only. They reflect the state of the strobe signal register bits 7 and 6.

**csout** Read only. Reflects the state of the column shift register output. If it is high the last column is selected.

**rsout** Read only. Reflects the state of the row shift register output. If it is high the last row is selected.

**Edge readout data** (EDOUT, address 5):

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| ed 7 | ed 6 | ed 5 | ed 4 | ed 3 | ed 2 | ed 1 | ed 0 |

**ed 7-0** Read only. Data from the actually selected cluster. Which one of the four sense amplifier outputs of each row circuit is selected depends on the bits 5 and 6 of the miscellaneous signals register (see next section).

**Loop register 0** (CNTOUT, address 6):

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| - | cnt0 6 | cnt0 5 | cnt0 4 | cnt0 3 | cnt0 2 | cnt0 1 | cnt0 0 |

**cnt0 6-0** Read only. Reflects the actual state of loop register 0. This register can be used to monitor a sequencer program.

**Sequencer address register** (RADR, address 7):

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| sndr | radr 6 | radr 5 | radr 4 | radr 3 | radr 2 | radr 1 | radr 0 |

**radr 6-0** Read only. Read back of the address counter of the sequencer.

**sndr** Read only. Reflects the output of the flipflop that synchronizes the 'next data ready handshake' input signal.

## C.2 Sequencer

The sequencer instructions of the EDDA chip are encoded in the upper 4 bits of the RAM word. The lower 12 bits contain the immediate data of the instruction. The only exception is the 'load fast clocks' command, which has an immediate data field of 14 bits. The following table lists the encoding of all the sequencer commands:

| mnemonic | encoding | name | function |
|---|---|---|---|
| LFC | 00xx | load fast clocks | loads non-overlapped clocks register |
| LMS | 0100 | load misc signals | loads miscellaneous signals register |
| LSS | 0101 | load strobe signals | loads strobe signals register |
| LLR | 0110 | load loop register | loads one of the four loop registers |
| JPZ | 1000 | jump zero or condition | jumps if loop register zero or condition true (without count) |
| JPN | 1001 | jump not zero | jumps if loop register not zero, if zero, waits for condition true |
| DLE | 1010 | delay with end condition | loads delay register and counts to zero, if zero, waits for condition true |
| DLL | 1011 | delay with loop condition | loads delay register and counts if condition true until zero |
| JPS | 1100 | jump shift register | jumps if shift register condition true |
| JPF | 1101 | jump flags | jumps if flag condition true |

If an instruction is encountered that is undefined, i.e. not listed in the table above, the sequencer stops execution. By changing the sequencer address counter it can be restarted.

### Sequencer load instructions

**Load fast clocks (LFC):**
The non-overlapped clock register is loaded with the lower 14 bits of the instruction word. All the bits are high active with respect to the non-overlapped logic. If one bit is changed from zero to one in this register the according output edge will be delayed by the non-overlapped delay, no matter whether the clock signal is high or low active. If the bit is cleared no delay is added to the signal.

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | vc | rclk2b | rclk1b | rclk2a | rclk1a | clr |

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| pcon | rpcon | clk3 | clk3read | clk2p | $\overline{\text{clk2}}$ | clk2 | clk1 |

| | |
|---|---|
| **vc** | State of the comparator voltage ($V_{\text{cmp}}$) switch (see section 2.4.1). This is the only exception from the rule stated above. If the vc bit is changed the non-overlapped delay is always applied to the activating signals of the $V_{\text{cmp}}$ switch. |
| **rclk2b, rclk1b, rclk2a, rclk1a** | Clock signals for the row readout sense amplifiers (see section 2.3.2). |
| **clr, clk3, clk3read, clk2p, $\overline{\text{clk2}}$, clk2, clk1** | Clock signals to the column clock buffers (see section 2.3.1). |
| **pcon, rpcon** | Precharge switch signals (see section 2.3.2). |

**Load misc signals (LMS):**
The miscellaneous signals register is loaded with the lower 7 bits of the instruction word.

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | - | - | - | - |

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| - | selhv | selab | en8 | bypass | gwritesa | gwrite | gread |

| | |
|---|---|
| **selhv, selab** | Row readout sense amplifier select for the edge readout. Selab selects bank a or b, selhv the horizontal or vertical edge data (see section 2.3.2). |
| **en8** | This bit enables the ninth readout cluster (the one that contains only 2 bits: row 0 and row 65). |
| **bypass, gwritesa** | Global write enable for the row storage amplifiers and activation of their bypass transistors (see section 2.3.2). |
| **gwrite, gread** | Global read and write enable signals for the column clock buffers (see section 2.3.1. |

**Load strobe signals (LSS):**
The strobe signals register is loaded with the lower 11 bits of the instruction word. Most of the bits in this register are strobe bits, i.e. they are not stored, but trigger some action.

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | - | strobe 8 | strobe 7 | ss 8 |

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| ss 7 | ss 6 | csclk | rsclk | csres | rsres | csin | rsin |

**strobe 8, strobe 7, ss 8, ss 7** Ss 7 and ss 8 are flag bits that can be used for handshaking between the sequencer and external hardware if the according auxiliary output is enabled. If strobe 7 or strobe 8 are set to one, ss 7 or ss 8 are cleared automatically with the sequencer clock, i.e. they are only high for one clock cycle. If strobe 7/8 is low, ss 7/8 is a normal register bit.

**ss 6** Ss 6 is for the edge readout handshake. It is set by a sequencer program after it has transferred edge data into the row readout sense amplifiers. This signals the bus interface that a read access to the edge readout data register can be accomplished without waits. After 8 reads ss 6 is automatically cleared. The next read access must wait until the next data is ready and ss 6 is set to one again by the sequencer. The sequencer itself can monitor the ss 6 bit if it wants to know if the row readout sense amplifiers have been read out.

**csclk, rsclk** Column/row shift clock. Strobe bits that generate one clock for the column or row shift register.

**csres, rsres** Column/row shift register reset. Strobe bits that clear the column or row shift register.

**csin, rsin** Column/row shift register serial in. Strobe bits that clock a one bit into the first shift register cell if written together with csclk or rsclk high.

**Load loop reg (LLR):**
The selected loop register is loaded with the lower 7 bits of the instruction word. The sequencer contains four independent loop register. They are 7 bits wide. Each loop register has a zero output that is included in the condition of the flow control instructions. If a test for a loop register is performed it will be automatically decremented by one until it has reached zero (except for JPZ, see below).

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | lr 1 | lr 0 | - | - |

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| - | lc 6 | lc 5 | lc 4 | lc 3 | lc 2 | lc 1 | lc 0 |

**lr 1-0** Number of the selected loop register.

**lc 6-0** Loop count that is loaded into the selected loop register.

### Sequencer flow control instructions

The flow control of the sequencer is based on various conditional jump and delay instructions. The delay instructions halt the address incrementation until their condition is fulfilled while the jump instructions can load a new address into the sequencer's address counter. Most of the flow control instructions use the following condition together with the loop registers zero signal:

seqcond = (mask 2 AND $\overline{\text{sndr}}$) OR (mask 1 AND sndr) OR (mask 0 AND sig)

**$\overline{\text{sndr}}$, sndr** Inverted and normal next data ready handshake input synchronized to the sequencer clock. Either polarity or edge of the handshake input can be used in the sequencer program. Edges can be detected by two consecutive flow control instructions.

**sig** The handshake signal bit from the control register.

**mask 2-0** Condition mask included in the instruction word. If both mask2 and mask1 are high the condition depends only on the zero signal, i.e. the handshake signals are turned off.

**Delay with end condition (DLE) and delay with loop condition (DLL):**

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0(DLE)/1(DLL) | lr 1 | lr 0 | mask 2 | mask 1 |

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| mask 0 | lc 6 | lc 5 | lc 4 | lc 3 | lc 2 | lc 1 | lc 0 |

**lr 1-0** Number of the selected loop register.

**mask 2-0** The mask for the handshake signals.

**lc 6-0** Loop count that is loaded into the selected loop register. The number of loops performed is lc+2.

The DLE instruction tests for 'zero[lr] AND seqcond' while the DLL instruction decrements the loop register only if seqcond is true. In both cases the program flow is delayed for at least lc+2 clocks.

**Jump zero or condition (JPZ) and jump not zero (JPN):**

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0(JPZ)/1(JPN) | lr 1 | lr 0 | mask 2 | mask 1 |

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| mask 0 | dadr 6 | dadr 5 | dadr 4 | dadr 3 | dadr 2 | dadr 1 | dadr 0 |

**lr 1-0** Number of the selected loop register.

**mask 2-0** The mask for the handshake signals.

**dadr 6-0** Destination address for the jump.

The JPN instruction will load dadr into the sequencer address counter if 'zero[lr]=true', if not, it will wait until seqcond=true to increment the address counter to point at the next instruction and decrement the selected loop register by one. With JPN a '`{···}while()`' loop can be constructed. JPZ is used as a kind of '`break`' statement, i.e. to exit from a loop. It carries out the jump if 'zero[lr] OR seqcond' and does not decrement the loop register at all.

**Jump shift register (JPS) and jump flags (JPF):**

| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0(JPS)/1(JPF) | mask 3 | mask 2 | mask 1 | mask 0 |

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| - | dadr 6 | dadr 5 | dadr 4 | dadr 3 | dadr 2 | dadr 1 | dadr 0 |

**mask 3-0** Mask for the jump condition.
JPS:'(mask 3 AND $\overline{\text{rsout}}$) OR (mask 2 AND rsout) OR (mask 1 AND $\overline{\text{csout}}$) OR (mask 0 AND csout)'
JPF:'(mask 3 AND $\overline{\text{ss 7}}$) OR (mask 2 AND ss 7) OR (mask 1 AND $\overline{\text{ss 6}}$) OR (mask 0 AND ss 6)'

**dadr 6-0** Destination address for the jump.

These instructions will load dadr into the address counter if their respective condition is true. If not, the address counter will be incremented to point at the next instruction.

# Appendix D

# Calculation of the node voltage $V_\mathrm{n}$ in case of a deactivated fuse

If the fuse is deactivated by omitting clk2/$\overline{\text{clk2}}$ during $\phi_2$ the characteristic length $l$ decreases. In this case the node is isolated and the pixel resistor charges the node capacitor $C_\mathrm{n}$ to the pixel value. $\lambda_\mathrm{sa}$ is ignored for the sake of simplicity. The first time clk2p is high without clk2/$\overline{\text{clk2}}$ changes $Q_\mathrm{n}$ from $Q_\mathrm{n0}$ to $Q_\mathrm{n1}$:

$$Q_\mathrm{n1} = \lambda_\mathrm{np} Q_\mathrm{p} + \lambda_\mathrm{np} Q_\mathrm{n0} \tag{D.1}$$

with $\lambda_\mathrm{np}$ defined as:

$$\lambda_\mathrm{np} = \frac{C_\mathrm{n}}{C_\mathrm{n} + C_\mathrm{p}} \tag{D.2}$$

This is also true for the step $i+1$:

$$Q_{\mathrm{n}i+1} = \lambda_\mathrm{np} Q_\mathrm{p} + \lambda_\mathrm{np} Q_{\mathrm{n}i} \tag{D.3}$$

Since $V_\mathrm{n}$ must converge towards $V_\mathrm{p}$ for large $i$ and be equal $V_\mathrm{n0}$ for $i = 0$ the following approach can be made:

$$V_\mathrm{n} = \frac{Q_\mathrm{n}}{C_\mathrm{n}} \overset{i \to \infty}{=} V_\mathrm{p} = \frac{Q_\mathrm{p}}{C_\mathrm{p}} \tag{D.4}$$

$$Q_{\mathrm{n}i} = \frac{C_\mathrm{n}}{C_\mathrm{p}} Q_\mathrm{p}(1 - \gamma^i) + \gamma^i Q_\mathrm{n0} \tag{D.5}$$

Substituting this into eq.D.3:

$$\frac{C_\mathrm{n}}{C_\mathrm{p}} Q_\mathrm{p}(1 - \gamma^{i+1}) + \gamma^{i+1} Q_\mathrm{n0} = \lambda_\mathrm{np} Q_\mathrm{p} + \lambda_\mathrm{np} \frac{C_\mathrm{n}}{C_\mathrm{p}} Q_\mathrm{p}(1 - \gamma^i) + \lambda_\mathrm{np} \gamma^i Q_\mathrm{n0} \tag{D.6}$$

This can be rewritten with corresponding terms above each other:

$$\begin{aligned} & \frac{C_\mathrm{n}}{C_\mathrm{p}} Q_\mathrm{p} && - && \gamma \frac{C_\mathrm{n}}{C_\mathrm{p}} Q_\mathrm{p} \gamma^i + \gamma \gamma^i Q_\mathrm{n0} \\ = & \left( \lambda_\mathrm{np} + \lambda_\mathrm{np} \frac{C_\mathrm{n}}{C_\mathrm{p}} \right) Q_\mathrm{p} && - && \lambda_\mathrm{np} \frac{C_\mathrm{n}}{C_\mathrm{p}} Q_\mathrm{p} \gamma^i + \lambda_\mathrm{np} \gamma^i Q_\mathrm{n0} \end{aligned} \tag{D.7}$$

Since $\lambda_\mathrm{np} + \lambda_\mathrm{np}\frac{C_\mathrm{n}}{C_\mathrm{p}} = \frac{C_\mathrm{n}}{C_\mathrm{p}}$ a comparison of the coefficients shows that $\gamma = \lambda_\mathrm{np}$ solves the equation.

$$Q_{\mathrm{n}i} = \frac{C_\mathrm{n}}{C_\mathrm{p}} Q_\mathrm{p}(1 - \lambda_\mathrm{np}^i) + \lambda_\mathrm{np}^i Q_{\mathrm{n}0} \tag{D.8}$$

Dividing by $C_\mathrm{n}$ gives the equivalent equation for the voltages:

$$V_{\mathrm{n}i} = V_\mathrm{p}(1 - \lambda_\mathrm{np}^i) + \lambda_\mathrm{np}^i V_{\mathrm{n}0} \tag{D.9}$$

# Appendix E

# Layer appearance

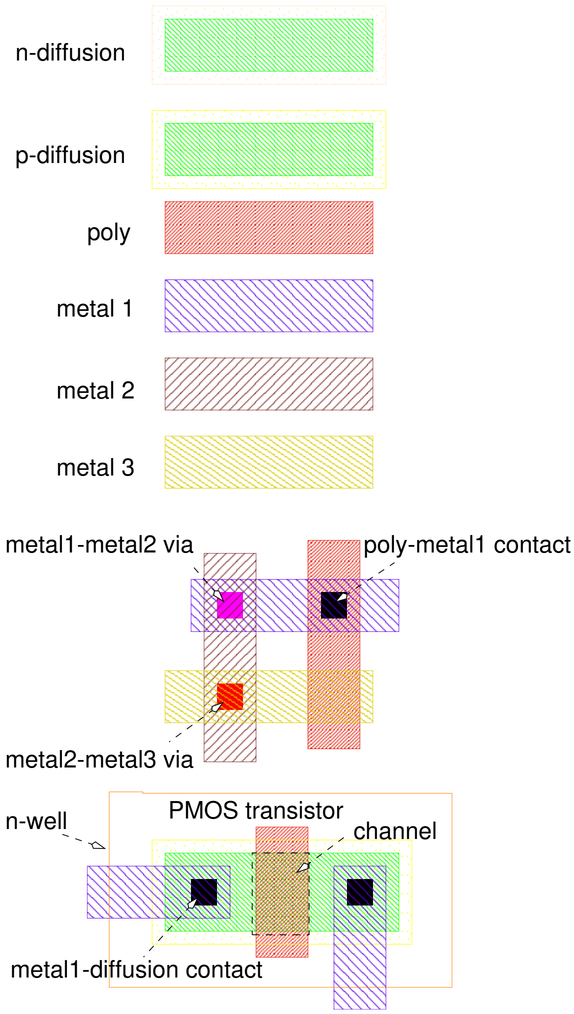



**Figure E.1:** Graphical appearance of the different process layers in the layout plots.

# Bibliography


[AMS94] C. Wong, G. Melcher, “Device Matching in CBQ”, Austria Mikro Systeme International AG, Unterpremstätten, Austria, 1994

[AMS98-1] “0.6 um CMOS Design Rules”, Austria Mikro Systeme International AG, Unterpremstätten, Austria, 1998

[AMS98-2] “0.6 um CMOS Joint Group Process Parameters”, Austria Mikro Systeme International AG, Unterpremstätten, Austria, 1998

[AMS98-3] “0.6 um CMOS Joint Group Process Parameters”, Austria Mikro Systeme International AG, Unterpremstätten, Austria, 1998, page 10

[AMS98-4] “0.6-Micron Standard Cell Databook”, Austria Mikro Systeme International AG, Unterpremstätten, Austria, 1997

[BEH92] A. Behr et al, *IEEE J. Solid-State Circuits*, vol. 27, october 1992, pp. 1470-1475

[BES97] R. Best, ”Phase-Locked Loops”, McGraw-Hill, ISBN 0-07-006051-7, 1997

[BES97-2] R. Best, ”Phase-Locked Loops”, McGraw-Hill, ISBN 0-07-006051-7, 1997, pp. 91-102

[CDS97-1] ”Cadence Online Library Openbook: Verilog-XL User Guide”, Cadence Design Systems, San Jose, USA, version 4.41, 1997

[CDS97-2] ”Cadence Online Library Openbook: Synergy Reference”, Cadence Design Systems, San Jose, USA, version 4.41, 1997

[CMC95] ”Draft Standard for a Common Mezzanine Card Family: CMC”, IEEE Std. P1386/Draft 2.0, Bus Architecture Standards Committee of the IEEE Computer Society, Institute of Electrical and Electronics Engineers, Inc, New York, USA, 1995

[CPP98] B. Stroustrup, “The C++ Programming Language”, Addison Wesley publishing company, ISBN 0-201-88954-4, 1998

[DHR99] “Dhrystone and MIPs performance of ARM processors”, `http://www.armltd.co.uk/DevSupp/Sales+Support/FAQ/Other/Index.html`, 1999

[DRO99] D. Droste, “Entwurf eines Wellenfrontsensors mit einem ASIC”, *Ph.D. thesis*, in german, Institute of Computer Engineering, University of Mannheim, 1999, chapter 6, page 111

[GEI90-1] R. Geiger, P. Allen, N. Strader, “VLSI Design Techniques for Analog and Digital Circuits”, McGraw-Hill, ISBN 0-07-023253-9, 1990, pp. 32-85

[GEI90-2] R. Geiger, P. Allen, N. Strader, “VLSI Design Techniques for Analog and Digital Circuits”, McGraw-Hill, ISBN 0-07-023253-9, 1990, pp. 302-318

[GEI90-3] R. Geiger, P. Allen, N. Strader, “VLSI Design Techniques for Analog and Digital Circuits”, McGraw-Hill, ISBN 0-07-023253-9, 1990, pp. 174-177

[GEI90-4] R. Geiger, P. Allen, N. Strader, “VLSI Design Techniques for Analog and Digital Circuits”, McGraw-Hill, ISBN 0-07-023253-9, 1990, pp. 511-515

[GEI90-5] R. Geiger, P. Allen, N. Strader, “VLSI Design Techniques for Analog and Digital Circuits”, McGraw-Hill, ISBN 0-07-023253-9, 1990, pp. 246-250

[GEI90-6] R. Geiger, P. Allen, N. Strader, “VLSI Design Techniques for Analog and Digital Circuits”, McGraw-Hill, ISBN 0-07-023253-9, 1990, pp. 466-469

[GEI90-7] R. Geiger, P. Allen, N. Strader, “VLSI Design Techniques for Analog and Digital Circuits”, McGraw-Hill, ISBN 0-07-023253-9, 1990, pp. 488-491

[GEI90-8] R. Geiger, P. Allen, N. Strader, “VLSI Design Techniques for Analog and Digital Circuits”, McGraw-Hill, ISBN 0-07-023253-9, 1990, pp. 805-808

[GRA93] P. Gray, R. Meyer, “Analysis and Design of Analog Integrated Circuits”, John Wiley & Sons Ltd, Chichester, England, ISBN 0-471-59984-0, 1993, pp. 464-466

[HAR89] J. Harris et al, “Resistive Fuses: Analog hardware for detecting discontinuities in early vision”, *Analog VLSI implementation of Neural Systems*, C. Mead and J. Ismail, Eds. , Kluwer, 1989

[HBV99] “Handbook of Computer Vision and Applications”, B. Jähne, ed., Academic Press, volume 2: Signal Processing and Pattern Recognition, ISBN 0 12 379772 1, 1999, p. 278

[INT99] “Image Processing Library Performance Specifications”, Intel, `http://developer.intel.com/vtune/perflibst/ipl/ipspecs.htm`, 1999

[JAE93-1] B. Jähne, “Digitale Bildverarbeitung”, in german, Springer Verlag, Heidelberg, 1993, pp. 96-100

[JAE93-2] B. Jähne, “Digitale Bildverarbeitung”, in german, Springer Verlag, Heidelberg, 1993, p. 108

[JAE93-3] B. Jähne, “Digitale Bildverarbeitung”, in german, Springer Verlag, Heidelberg, 1993, p. 103-105

[JOS96] N. Josuttis, “Die C++ Standardbibliothek: Eine detaillierte Einführung in die vollständige ANSI/ISO-Schnittstelle”, in german, Addison-Wesley-Longman, Bonn, Germany, ISBN 3-8273-1023-7, 1996

[KOB91] P. Kobayashi et al, “An Active Resistor network for Gaussian Filtering of Images”, *IEEE J. Solid-State Circuits*, vol. 26, may 1991, pp. 738-742

[KLE95-1] R. Klette, P. Zamperoni, “Handbuch der Operatoren für die Bildverarbeitung”, in german, Vieweg, Wiesbaden, Germany, ISBN 3-528-16431-X, 1995, pp. 210-214

[KLE95-2] R. Klette, P. Zamperoni, “Handbuch der Operatoren für die Bildverarbeitung”, in german, Vieweg, Wiesbaden, Germany, ISBN 3-528-16431-X, 1995, pp. 144-153

[LAK94-1] K. Laker, W. Sansen, “Design of Analog Integrated Circuits and Systems”, McGraw-Hill, ISBN 0-07-036060-X, 1994, pp. 587-591,630

[LAK94-2] K. Laker, W. Sansen, “Design of Analog Integrated Circuits and Systems”, McGraw-Hill, ISBN 0-07-036060-X, 1994, pp. 535-562

[LAN98] J. Langeheine, “Entwurf und Test eines optisch konfigurierbaren Widerstandsnetzwerks”, *diploma thesis*, in german, Institute of High Energy Physics, Heidelberg University, 1998, pp. 6-17

[LOO98] M. Loose, K. Meier, J. Schemmel, “CMOS image sensor with logarithmic response and self calibrating fixed pattern noise correction”, *Advanced Focal Plane Arrays and Electonic Cameras*, Thierry M. Bernard, ed., Proc. SPIE 3410 ISBN 0-8194-2862-0, 1998, pp. 117-127

[LOO99] M. Loose, K. Meier, J. Schemmel, “Self-calibrating logarithmic CMOS image sensor with single chip camera functionality”, IEEE CMOS & AIS workshop, Karuizawa, Japan, 1999, Proceedings p. R27

[LOV98] S. Lovett, M. Welten, et al, “Optimizing MOS Transistor Mismatch”, *IEEE J. Solid-State Circuits*, vol. 33, january 1998, pp. 147-152

[MEA89] C. Mead, “Analog VLSI and Neural Systems”, Addison Wesley publishing company, ISBN 0-201-05992-4, 1989, pp. 339-351

[MEI96] M. Loose, K. Meier, J. Schemmel, “A camera with adaptive photoreceptors for a tactile vision aid”,*Intelligent Robots and Computer Vision XV: Algorithms, Techniques, Active Vision and Materials Handling*, David P. Casaent, ed., Proc. SPIE 2904 ISBN 0-8194-2306-8, 1996, pp. 528-537

[MEN99] L. Schmidt-Mende, “Entwicklung eines optoelektronischen Sehhilfesystems”, *diploma thesis*, in german, Institute of High Energy Physics, Heidelberg University, 1999

[MIC99] ”Benutzerhandbuch microEnable”, in german, Silicon Software GmbH, M2, 16, D-68161 Mannheim, germany, 1999

[PCI95] ”PCI Local Bus Specification”, revision 2.1, PCI Special Interest Group, P.O.Box 14070, Portland, OR97214, USA, 1995

[PRI95] B. Prince, “Semiconductor Memories”, John Wiley & Sons Ltd, Chichester, England, ISBN 0-471-92465-2, 1995, pp. 162-174

[SAR99] “Intel StrongARM SA-1110 Microprocessor”, advance information brief datasheet, Intel Corporation, Order Number: 278241-002, July 1999

[SCH97] J. Schemmel, “Design und Test einer Single-Chip Kamera mit integrierter Analog/Digital-Wandlung auf der Basis adaptiver Photorezeptoren”, *diploma thesis*, in german, Institute of High Energy Physics, Heidelberg University, 1997

[SYN98] ”Synopsys Online Documentation”, Synopsys, Inc., Mountain View, CA, USA, v1998.08, 1998

[VIT98] E. Vittoz, W. Sansen, ”Advanced Analog IC Design”, Electronics Laboratories Advanced Engineering Course on CMOS & BiCMOS IC Design ’98, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 1998, pp. PA-3

[VHD97] ”VHDL Language Reference Manual”, IEEE Std. 1076.1, Design Automation Standards Committee of the IEEE Computer Society, Institute of Electrical and Electronics Engineers, Inc, New York, USA, 1997

[XIL97] ”XC4000E and XC4000X Series Field Programmable Gate Arrays”, databook, Xilinx, Inc, San Jose, USA, 1997

[YU92] P. Yu et al, “CMOS Resistive Fuses for Image Smoothing and Segmentation”, *IEEE J. Solid-State Circuits*, vol. 27, april 1992, pp. 545-552

# Danksagung

Während der Arbeit an dieser Dissertation gab es immer Menschen, die mir die notwendige fachliche und persönliche Unterstützung zukommen ließen. Ihnen möchte ich an dieser Stelle ganz herzlich danken:

- Herrn Prof. Dr. K. Meier, der mir überhaupt erst die Möglichkeit eröffnet hat, eine Arbeit mit so ungewissem Ausgang in Angriff nehmen zu dürfen.
- Herrn Prof. Dr. Bernd Jähne, der freundlicherweise das Zweitgutachten übernommen hat.
- Dr. Dirk Droste, Jörg Langeheine und Markus Loose, da die mit ihnen geführten Fachdiskussionen wesentlich zum Gelingen dieser Arbeit beigetragen haben.
- Michael Keller, der die so dringend benötigte CAD Software am Laufen hielt.
- Ralf Achenbach, für seine Hilfe beim Herstellen der Platinen sowie dem Organisieren wichtiger Messgeräte.
- Cornelius Schumacher, für seine vielen guten Ratschläge.
- Allen Mitarbeitern des Instituts für Hochenergiephysik, die sowohl für die nötige Infrastruktur als auch das gute Betriebsklima gesorgt haben.
- All den Programmierern, die so unentbehrliche Software wie LaTeX, XFIG, NEDIT und XMGR geschrieben haben und mir damit viele graue Haare (und dem Staat viel Geld) erspart haben.
- Barbara Ueberle, die mit ihren Englischkenntnissen sowie ihrer Bereitschaft, meinen Stress, meine Zweifel und meine Überstunden zu ertragen, diese Arbeit unterstüzt hat.
- Meinen Eltern, deren Hilfe vielleicht nicht so konkret, aber durch ihre bedingungslose Unterstützung meiner Ausbildung um so wichtiger war.