Helsinki University of Technology, Electronic Circuit Design Laboratory
Report 38, Espoo 2003

Mixed-Mode Cellular Array Processor Realization for Analyzing Brain Electrical Activity in Epilepsy

Mika Laiho

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Electrical and Communications Engineering for public examination and debate in Auditorium S4 at Helsinki University of Technology (Espoo, Finland) on the 28th of June, 2003, at 12 o'clock.

Helsinki University of Technology
Department of Electrical and Communications Engineering
Electronic Circuit Design Laboratory

Teknillinen korkeakoulu
Sähkö- ja tietoliikennetekniikan osasto
Piiritekniikan laboratorio

Distribution:
Helsinki University of Technology
Department of Electrical and Communications Engineering
Electronic Circuit Design Laboratory
P.O.Box 3000
FIN-02015 HUT
Finland
Tel. +358 9 4512271
Fax: +358 9 4512269

ISBN 951-22-6597-4
ISSN 1455-8440

Otamedia Oy
Espoo 2003

Abstract

This thesis deals with the realization of hardware that is capable of computing algorithms that can be described using the theory of polynomial cellular neural/nonlinear networks (CNNs). The goal is to meet the requirements of an algorithm for predicting the onset of an epileptic seizure. The analysis associated with this application requires extensive computation of data that consists of segments of brain electrical activity. Different types of computer architectures are overviewed. Since the algorithm requires operations in which data is manipulated locally, special emphasis is put on assessing different parallel architectures. An array computer is potentially able to perform local computational tasks effectively and rapidly.

Based on the requirements of the algorithm, a mixed-mode CNN is proposed. A mixed-mode CNN combines analog and digital processing so that the couplings and the polynomial terms are implemented with analog blocks, whereas the integrator is digital. A/D and D/A converters are used to interface between the analog blocks and the integrator. Based on the mixed-mode CNN architecture a cellular array processor is realized. In the realized array processor the processing units are coupled with programmable polynomial (linear, quadratic and cubic) first neighborhood feedback terms. A 10mm^2, 1.027 million transistor cellular array processor, with 2×72 processing units and 36 layers of memory in each is manufactured using a 0.25μm digital CMOS process. The array processor can perform gray-scale Heun's integration of spatial convolutions with linear, quadratic and cubic activation functions for 72×72 data while keeping all I/O operations during processing local. One complete Heun's iteration round takes 166.4μs, while the power consumption during processing is 192mW. Experimental results of statistical variations in the multipliers and polynomial circuits are shown. Descriptions regarding improvements in the design are also explained. The results of this thesis can be used to assess the suitability of the mixed-mode approach for implementing an implantable system for predicting epileptic seizures. The results can also be used to assess the suitability of the approach for implementing other

applications.

Keywords: Mixed-Mode Integrated Circuits, Analog Arithmetic Circuits, (Polynomial) Cellular Neural/Nonlinear Networks, CMOS, Discrete-Time Systems, Cellular Array Processors

Preface

This thesis is the result of work that was carried out in the Electronic Circuit Design Laboratory (ECDL) of Helsinki University of Technology between 1999 and 2003. The inspiring and relaxed university atmosphere has been a source of continuous challenges and learning opportunities. I want to thank professors Kari Halonen and Veikko Porra for recruiting me to the ECDL and for giving me an interesting research topic.

I have been privileged to work with very competent people and have received expert guidance at the course of the thesis work. I want to thank Prof. Halonen for his efforts as the supervisor of my thesis. The contribution of Prof. Ari Paasio in guiding this work with his versatile expertise has been invaluable. His devotion towards guiding the thesis persisted even after he started as a professor in the University of Turku. It has been a pleasure working with him. In addition to Prof. Paasio, teamwork with the rest of the "CNN team", namely Asko Kananen, Lauri Koskinen, Mikko Talonen and Jacek Flak has been both fruitful and fun. Furthermore, the whole crew of the ECDL deserves recognition due to their enthusiasm and fellowship. Specifically, advice from Dr. Mikko Waltari and Dr. Teemu Salo in various matters of circuit design has been valuable. Our secretary, Helena Yllö, has done excellent work in taking care of the practical matters. For their devoted and prompt work as pre-examiners of this thesis, I want to thank Prof. Peter Kinget (Columbia University, NY, USA) and Dr. Ricardo Carmona (Centro Nacional de Microelectronica, Sevilla, Spain). I also want to acknowledge the fruitful discussions with Prof. Ronald Tetzlaff and Dr. Roland Kunz from the Johann Wolfgang Goethe-University, Frankfurt, Germany.

This thesis was funded by the Academy of Finland (projects "Integrated Parallel Processors for Future Multimedia, Medical Imaging and Communication Systems" and "Integrated Parallel Processors for Future Data Processing and Analyzing Systems") and the Graduate School in Electronics, Telecommunications, and Automation (GETA). The work was also supported by the Nokia Foundation, the Foundation of Technology (Tekniikan edistämissäätiö), the Instrumentarium Science Foundation

and the Foundation of Electronics Engineers (Elektroniikkainsinöörien säätiö). The contribution of these in providing the resources for the research work is gratefully acknowledged.

During the thesis work, I have managed to spend active social life in various events and activities. For this I am grateful to my girlfriend Tanja and to my friends. Finally, my parents Marja-Leena and Olavi deserve my warmest thanks for their lifelong support.

Espoo, June 2003.

Mika Laiho

Contents

This page is intentionally left blank.

Symbols and Abbreviations

(ii_c, jj_c, kk)	data unit in kk^{th} block of ii_c^{th} row and jj_c^{th} column in data partitioned by chopping
(ii_f, jj_f, kk)	data unit in kk^{th} block of ii_f^{th} row and jj_f^{th} column in data partitioned by folding
$\Delta\Sigma$	Delta-Sigma
$\Delta x_{i,j}(n)$	time-discretized rate of change of state in a DTCNN
$\mathbf{A}$	space-independent convolution mask, feedback template
$\mathbf{A}^{(1)}$	space-independent linear feedback template in a polynomial CNN
$\mathbf{A}^{(2)}$	space-independent quadratic feedback template in a polynomial CNN
$\mathbf{A}^{(3)}$	space-independent cubic feedback template in a polynomial CNN
$\mathbf{A}_{i,j}$	space-dependent convolution mask, feedback template
$\mathbf{B}$	space-independent feedback template
$\mathbf{B}^{(1)}$	space-independent linear feedforward template in a polynomial CNN
$\mathbf{B}^{(2)}$	space-independent quadratic feedforward template in a polynomial CNN
$\mathbf{\Phi}_F$	bulk Fermi potential
τ	fixed delay in the method of delays
$\tau_a^{(1)}$	delay in linear feedback path
$\tau_a^{(2)}$	delay in second order polynomial feedback path
$\tau_a^{(3)}$	delay in third order polynomial feedback path
$a[L]$	SRAM address signals that determine which layer is being accessed, $L \in [1, 36]$

A_β	process dependent constant that is associated with current factor mismatch
$A_{k,l}$	interconnection strength (weight)
A_m	the area occupied by the state memory of a cell
A_o	the area of a cell excluding state memory
A_{Vt}	process dependent constant that is associated with threshold voltage mismatch
$act[kk]$	decoder activation signals that determine which data block is being processed, $kk \in [1, 36]$
ADC_D	ADC for sampling the PU input current after the bias current has been subtracted using ADC_Q
ADC_Q	ADC for subtracting the bias current from the PU input current
b_A	number of bits in $r_{i,j}(n)$
b_D	number of bits in DAC of mixed-mode CNN cell
C	constant integration term in CNN state equation
$C_{i,j}$	cell in i^{th} row and j^{th} column
$C_{ii,jj}$	cell in ii^{th} row and jj^{th} column in a partitioned mixed-mode CNN
$C_m(r)$	correlation integral
C_x	CNN state capacitor
$ct[6, 0]$	asynchronously generated ADC control signals
D^*	estimator of effective correlation dimension
$D_{2,max}$	maximum obtainable dimension
D_2^{eff}	effective correlation dimension
D_{CNN}	CNN dimension
$DAP1$	the part of the main unit of a PU that provides analog output currents that are linear, quadratic and cubic functions of $x[7, 0]$
$DAP2$	the part of the border unit of a PU that provides analog output currents that are linear, quadratic and cubic functions of $rgi[7, 0]$
$f_N(kk)$	top row mapping function in a partitioned mixed-mode CNN
f_s	sampling frequency
$f_S(kk)$	bottom row mapping function in a partitioned mixed-mode CNN

H	Heaviside (threshold) function
h	integration constant
I	constant biasing term in CNN state equation
I/V	current-to-voltage
$I_a^{(1)}$	linear self-feedback current
$I_a^{(2)}$	quadratic self-feedback current
$I_a^{(3)}$	cubic self-feedback current
I_{max}	absolute value of the maximum of I_{sum}
I_{OC}	sum of cell input currents excluding self-feedback currents
$I_{sum,i,j}(t)$	sum of input currents of cell $C_{i,j}$ at time t
I_{sum}	sum of cell input currents
I_s	specific current proportional to slope factor
I_{unit}	unit current
$Ib_{k,l}$	output current of $MB9$
$Ic_{k,l}$	output current of $MC18$
$k1$	auxiliary term in Runge-Kutta integration method
$k2$	auxiliary term in Runge-Kutta integration method
$k3$	auxiliary term in Runge-Kutta integration method
$k4$	auxiliary term in Runge-Kutta integration method
k_A	product of k_A and I_{unit} bounds the maximum of $r_{i,j}(n)$
k_D	scaling parameter that affects the integration step
k_t	number of changes of template orientation during an iteration round
L	length of the channel of a MOSFET
lsb	least significant bit
M	number of rows in data
$MB9$	multiplier block in border unit that contains nine multipliers
$MC18$	multiplier block in main unit that contains eighteen multipliers
MN	number of samples that D_2^{eff} is determined from
msb	most significant bit

N	number of columns in data
n	time index
N_r	number of different values that radius of influence in correlation integral is assigned
n_s	slope factor
nri	number of iterations
P_{alg}	power consumption of analog parts of a cell
$P_{c,inst}(R_b)$	the instantaneous power consumption of a cell in a partitioned network
P_{conv}	power consumption of data converters of a cell
P_{logic}	power consumption of digital logic of a cell
P_{logic}	power consumption of digital memory of a cell in a full size network
R	constant dissipative term in CNN state equation
r	radius of influence in correlation integral
$R_a(R_b)$	ratio of the area of a partitioned network and a full size network
R_b	number of data blocks in a partitioned data
$r_{i,j}(n)$	shifted digital rate of change of state of a mixed-mode CNN cell $C_{i,j}$
$r^*_{i,j}(n+1)$	rate of change of state at predicted state in Heun's integration method
r_l	lower bound of radius of influence in correlation integral
R_n	radius of neighborhood
$R_{p,inst}(R_b)$	ratio of the instantaneous power consumption of a partitioned network and a full size network
r_u	upper bound of radius of influence in correlation integral
R_x	number of horizontal data blocks in a partitioned data
R_y	number of vertical data blocks in a partitioned data
REG_1	36×8 bit register in the integrator
REG_2	36×8 bit register in the integrator
REG_CMB	8 bit register in the integrator
REG_Y	36×8 bit register in the integrator

rg_b	integrator output, controls data bus $rgo[7,0]$
rg_c	integrator output, controls data bus $x[7,0]$
$rgi[7,0]$	data bus, realizes input register connection of a PU
$rgo[7,0]$	data bus, realizes output register connection of a PU
$ri_{i,j}(n)$	digital rate of change of state of a mixed-mode CNN cell $C_{i,j}$
s	number of steps in the integration method
$T_f(R_b)$	processing time of data as a function of the number of blocks
t_{lt}	template load time
t_l	time of logic operations
t_{rd}	global read time
t_{sett}	analog settling time
t_s	sampling time
t_t	time it takes to change the orientation of a template
T_{unit}	unit transconductance
t_{wd}	global write time
$u_{i,j}$	input of a CNN cell in i^{th} row and j^{th} column
U_{in}	number of identical (unit) transistors in parallel at the input of a current mirror
U_{out}	number of identical (unit) transistors in parallel at the output of a current mirror
U_T	thermal voltage
V/I	voltage-to-current
$V_{i,j}(n)$	time-discretized CNN state voltage of cell $C_{i,j}$
$V_{i,j}(t)$	CNN state voltage of cell $C_{i,j}$ at time t
V_p	pinch-off voltage
V_{unit}	unit voltage
W	width of the channel of a MOSFET
$x[7,0]$	data bus that is controlled by integrator output rg_c
$x_{i,j}$	state of a CNN cell in i^{th} row and j^{th} column
$x_{i,j}^{(k1)}(n+1)$	auxiliary state in Runge-Kutta integration method

$x_{i,j}^{(k2)}(n+1)$	auxiliary state in Runge-Kutta integration method
$x_{i,j}^{(k3)}(n+1)$	auxiliary state in Runge-Kutta integration method
$x_{i,j}^{*}(n+1)$	predicted state in Heun's integration method
$x_{ii,jj,kk}$	state of kk^{th} layer of a cell in ii_f^{th} row and jj_f^{th} column in a partitioned mixed-mode CNN
$X_m(i)$	a vector formed from samples of EEG
$y_{i,j}$	output of a CNN cell in i^{th} row and j^{th} column
1D	one dimensional
2D	two dimensional
3D	three dimensional
lsb	least significant bit
msb	most significant bit
AμP	Analog Microprocessor
A/D	Analog-to-Digital
ADC	Analog-to-Digital Converter
ALU	Arithmetic Logic Unit
AND	logic AND operation
ANN	Artificial Neural Network
ARM8	RISC microprocessor core
ASIC	Application Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
B	Border Unit of a PU
C	Main Unit of a PU
C•RAM	Computational RAM
CAM	Content Addressable Memory
CNN	Cellular Neural/Nonlinear Network
CNNUM	CNN Universal Machine
D/A	Digital-to-Analog
DAC	Digital-to-Analog Converter
DCT	Discrete Cosine Transform

DIBL	Drain-Induced Barrier Lowering
DRAM	Dynamic RAM
DSM	Deep Sub-Micron
DSP	Digital Signal Processor
DTCNN	Discrete-Time CNN
EEG	Electroencephalogram
EKV	Enz-Krummenacher-Vittoz transistor model
ELDO	circuit simulation software
FSR	Full Signal Range
GAPU	Global Analogic Programming Unit
H.26L	a video coding standard
HI	Logic high signal level
HSPICE	circuit simulation software
I/O	Input/Output
IA-64	64 bit Intel Architecture
ITRS	International Technology Roadmap for Semiconductors
LO	Logic low signal level
MIMD	Multiple Instruction Multiple Data
MOS	Metal-Oxide-Semiconductor
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
MPU	Microprocessor Unit
NoC	Network on Chip
NSIP	Near Sensor Image Processing
OR	logic OR operation
PC	Personal Computer
PCB	Printed Circuit Board
PE	Processing Element
PPM	Pulse Phase Modulation
PU	Processing Unit

PWL	Piecewise Linear
PWM	Pulse Width Modulation
RAM	Random Access Memory
RC	Resistor Capacitor
RISC	Reduced Instruction Set Computer
SAR	successive approximation register
SIMD	Single Instruction Multiple Data
TOF	Time of Flight
W/L	Width over Length

Chapter 1

Introduction

1.1 Motivation

The minimum feature sizes of advanced silicon processes have decreased in a predictable way for decades. According to the ITRS, this development is predicted to continue for several years. For example, it is estimated that the printed gate length in microprocessor-unit (MPU) products will be 35nm in 2007 [1]. This scaling down affects the design of digital processors in several ways: the number of transistors that can be fitted into a die increases, the maximum clock rate increases, and the dynamic power consumption decreases. Another way of improving processor performance is by moving towards parallel architectures. Due to architectural improvements in serial data processors, several instructions can be processed during one clock cycle.

In some applications, the algorithm is either inherently (for example image processing tasks), or can be converted into a form that is suitable for computation in an array processor. In an array processor, the processing units and memory are distributed in a regular parallel formation and local computation is performed. The regularity of the array makes the prediction of delays easier and reduces memory access times. This allows the full potential of advanced silicon processes to be utilized more easily. Array processors can be realized using digital or analog computational blocks or by combining both. The parallel computing power of an array processor may act as an enabling technology for new applications.

Cellular neural/nonlinear network (CNN) theory can readily describe convolution-based local operations; it also proposes an architecture for realizing a parallel computing device. The state of a cell in a CNN is updated by using initial data, control terms and continuous-time spatio-temporal convolutions that are controlled by a

nonlinear activation function (conventionally a PWL function) of the cell state. The use of a CNN has been proposed for various applications in which the capability of manipulating data locally at a high speed is required. Examples of potential CNN applications are low-bit rate video coding [2], [3], visual computers with combined sensing and processing [4] and digital error correction [5]. In a polynomial CNN, the activation function is a polynom. The use of polynomial CNNs has been proposed for displacement vector estimation [6], for 3D modeling of soil [7] and for analysis of brain electrical activity in epilepsy [8]. The realization of the polynomial terms introduces new challenges to the implementation. In this thesis, realization of a cellular array processor for computing algorithms that can be described using the theory of polynomial CNNs is studied. The realization seeks to meet the needs of the algorithm for analyzing brain electrical activity in epilepsy. The algorithm has been developed by Prof. Tetzlaffs group in Johann Wolfgang Goethe-University in Frankfurt. A long-term goal is to implement an implantable system that can predict the onset of an epileptic seizure and possibly prevent it by using automatic pharmacological means. An array computer may be suitable for doing the computational tasks of the algorithm effectively and rapidly. Algorithm developers can use the results of this thesis to assess the applicability of the realized cellular array processor.

1.2 Research Contribution

The research in this thesis focuses on the hardware realization of a cellular array processor for analyzing brain electrical activity in epilepsy. Since the weights (couplings between processors) are programmable, a similar type of design can be applied to other applications that are modeled with the aid of a polynomial CNN. The work towards completion of this thesis proceeded so that, first, the special features of the epilepsy prediction algorithm were highlighted and potential realization alternatives were assessed and, second, a realization architecture was chosen and a cellular array processor was designed and manufactured. Finally, the performance of the individual circuit blocks along with the system-level performance was measured. Most of the measurements were carried out using a PCB measurement board. All the work towards this thesis was carried out independently by the author under the guidance of Prof. Ari Paasio and supervision of Prof. Kari Halonen.

The main research contribution of this thesis is the introduction of a mixed-mode CNN that can be used to realize an autonomous polynomial CNN. The mixed-mode CNN stores data robustly in digital domain and uses analog arithmetic circuits to perform the multiplication and to generate the polynomial terms. In a mixed-mode

CNN, the polynomial terms can be realized using one-quadrant circuits, while a four-quadrant operation can be achieved by digitally steering the current into a positive or negative sum node. Another research contribution of this thesis is its description of how a mixed-mode CNN can be realized so that global interaction is possible even when processing partitioned data. Some building blocks of analog array computers were described in Reference [9] and an analog polynomial CNN was investigated in References [10] and [11]. The theory of polynomial mixed-mode CNNs was described in References [12]- [16]. Selected circuit blocks and measurement results of the implemented cellular array processor were reported in References [17]- [19]. A large portion of the implementation and measurement results of the cellular array processor (Chapters 5 and 6) has not been reported outside this thesis. However, with regard to this, two papers have been accepted for publication (References [20] and [21]) and one has been submitted (Reference [22]). The work towards this thesis has also contributed to the publications of References [3] and [23]- [31].

1.3 Organization of the Thesis

This thesis is organized into seven chapters as follows. Chapter 1 serves as an introduction to the topic, while Chapter 2 reviews selected processor architectures and demonstrates a trend towards parallel architectures.

Chapter 3 describes how analysis of brain electrical activity in epilepsy can be used to predict an epileptic seizure, and shows how such a seizure can be predicted using a polynomial CNN. The requirements of hardware that could be used to realize the algorithm are identified.

Chapter 4 depicts the mixed-mode CNN architecture. It describes how the architecture can cope with the requirements of the epilepsy application. The choice of the integration method and the resolutions of the ADCs and DACs are studied. This chapter also shows how a mixed-mode CNN can be designed so that partitioned data can interact globally.

The network level design of the implemented cellular array processor is shown in Chapter 5, which also describes the design of the digital parts and the layout design. The design of the analog arithmetic circuits and the data converters is also covered, and measurement results are shown.

Chapter 6 describes how the performance of the mixed-mode CNN could be improved when more advanced silicon processes are used. Finally, Chapter 7 concludes the thesis.

References

[1] International Technology Roadmap for Semicionductors, 2002 update.

[2] A. Stoffels, T. Roska, L. O. Chua, 'Object-Oriented Image Analysis for Very-Low-Bitrate Video-Coding Systems using the CNN Universal Machine', *International Journal of Circuit Theory and Applications*, Wiley, Vol. 25, 235-258, 1997.

[3] A. Kananen, A. Paasio, M. Laiho, K. Halonen, 'CNN Applications from the Hardware Point of View: Video Sequence Segmentation', *International Journal of Circuit Theory and Applications*, Wiley, Vol. 30, pp. 117-137, 2002.

[4] T. Roska, A. Rodriquez-Vazquez, *Towards the Visual Microprocessor*, Wiley, NY, 2000.

[5] A. Kananen, A. Paasio, S. Lindfors, K. Halonen, 'A Cellular Nonlinear Network for Digital Error Correction', *IEEE International Symposium on Circuits and Systems*, Monterey, Vol. 3, pp. 255-258, 1998.

[6] D. Feiden, R. Tetzlaff, "Displacement Vector Estimation with Cellular Neural Networks", *International Joint Conference on Neural Networks*, Honolulu, Vol. III, pp. 2049 -2052, 2002

[7] P. Lopez, et al., "CNN-Based 3D Thermal Modeling of the Soil for Antipersonnel Mine Detection", *Proceedings of the International Workshop on Cellular Neural Networks and their Applications*, Frankfurt, pp. 307-314, 2002.

[8] R. Tetzlaff, R. Kunz, C. Ames, D. Wolf, 'Analysis of Brain Electrical Activity in Epilepsy with Cellular Neural Networks (CNN)', *Proceedings of the European Conference on Circuit Theory and Design (ECCTD'99)*, Stresa, Italy, pp. 1007-1010, 1999.

[9] M. Laiho, A. Paasio, K. Halonen, 'Building Blocks for Large Annealed Compact Neural Networks', *IEEE International Symposium on Circuits and Systems*, Geneva, pp. 415-418, 2000.

[10] M. Laiho, A. Paasio, K. Halonen, 'Structure of a CNN with Linear and Second Order Polynomial Feedback Terms', *6th International Workshop on Cellular Neural Networks and their Applications*, Catania, pp. 401-405, 2000.

[11] M. Laiho, A. Paasio, K. Halonen, 'CNN-based Brain Electrical Activity Monitoring System', *Baltic Electronic Conference*, Tallinn, pp. 191-192, 2000.

[12] M. Laiho, A. Paasio, A. Kananen, K. Halonen, 'A Mixed-Mode Polynomial Type CNN for Analyzing Brain Electrical Activity in Epilepsy', *International Journal of Circuit Theory and Applications*, Vol. 30, pp. 165-180, 2002.

[13] M. Laiho, A. Paasio, A. Kananen, K. Halonen, 'Discrete Time Analog Polynomial Type CNN with Digital State', *IEEE International Symposium on Circuits and Systems*, Sydney, pp. Vol. 3, pp. 497-500, 2001.

[14] M. Laiho, A. Paasio, A. Kananen, K. Halonen, 'Effects of Partitioning in a Mixed-Mode CNN', *European Conference on Circuit Theory and Design*, Espoo, Vol. 3, pp. 277-280, 2001.

[15] M. Laiho, A. Paasio, A. Kananen, K. Halonen, 'Border Connection Schemes in Partitioned Mixed-Mode CNNs', *European Conference on Circuit Theory and Design*, Espoo, Vol. 3, pp. 265-268, 2001.

[16] M. Laiho, A. Paasio, A. Kananen, K. Halonen, 'Cell and Network Level Design of a Mixed-Mode CNN', *IEEE International Symposium on Circuits and Systems*, Scottsdale, Vol. 1, pp. 621-624, 2002.

[17] M. Laiho, A. Paasio, A. Kananen, K. Halonen, 'Realization of Couplings in a Polynomial Type Mixed-Mode CNN', *7th International Workshop on Cellular Neural Networks and their Applications*, Frankfurt, pp. 422-429, 2002.

[18] M. Laiho, A. Paasio, A. Kananen, K. Halonen, 'A/D and D/A Converters in a Mixed-Mode CNN', *7th International Workshop on Cellular Neural Networks and their Applications*, Frankfurt, pp. 436-443, 2002.

[19] M. Laiho, A. Paasio, A. Kananen, K. Halonen, 'A Mixed-Mode Array Processor with Polynomial-Type Couplings', *28th European Solid-State Circuit Conference*, Florence, pp. 695-698, 2002.

[20] M. Laiho, A. Paasio, A. Kananen, K. Halonen, 'A Mixed-Mode Polynomial Cellular Array Processor Hardware Realization', accepted to *IEEE Transactions on Circuits and Systems-I*.

[21] M. Laiho, A. Kananen, A. Paasio, K. Halonen, 'High Speed Cellular Array Computer Realizations for Low Power Applications', accepted for publication in *International Joint Conference on Neural Networks*, Portland, 2003.

[22] M. Laiho, A. Paasio, A. Kananen, K. Halonen, 'Realization of Couplings in a Polynomial Type Mixed-Mode Cellular Neural Network', submitted to *International Journal of Neural Systems*.

[23] A. Paasio, A. Kananen, M. Laiho, K. Halonen, 'A Compact Computational Core for Image Processing', *European Conference on Circuit Theory and Design*, Espoo, Vol. 1, pp. 337-339, 2001.

[24] L. Koskinen, A. Paasio, M. Laiho, K. Halonen, 'Effect of CNN Shape Segmentation on MPEG-4 Shape Bit-Rate', *IEEE International Symposium on Circuits and Systems*, Scottsdale, 2002, Vol. 4, pp. 552-555.

[25] A. Paasio, M. Laiho, A. Kananen, K. Halonen, 'An Analog Array Processor Hardware Realization with Multiple New Features', *International Joint Conference on Neural Networks*, Honolulu, pp. 1952-1955, 2002.

[26] L. Koskinen, M. Laiho, A. Paasio, K. Halonen, 'MPEG-4 Based Modifications to a CNN Segmentation Chip', *7th International Workshop on Cellular Neural Networks and their Applications*, Frankfurt, pp. 71-77, 2002.

[27] A. Paasio, M. Laiho, A. Kananen, K. Halonen, 'Different Implementation Approaches for Analogue and Mixed-Signal Array Processing', *Proceedings of U.R.S.I. Kleinheubacher Tagung* (Kleinheubacher Berichte), Band 45, Kleinheubach, Germany, pp. 135-138, 2001.

[28] A. Paasio, M. Laiho, J. Poikonen, A. Kananen, K. Halonen, 'A 32x32 Cellular Test Chip Targeting New Functionalities', *IEEE International Symposium on Circuits and Systems*, Bangkok, Vol. III, pp. 506-509, 2003.

[29] M. Talonen, M. Laiho, A. Paasio, K. Halonen, '0.18um CMOS Image Sensor for Cellular Computers', accepted to *European Conference on Circuit Theory and Design*, Krakow, 2003.

[30] M. Laiho, A. Paasio, A. Kananen, 'A Parallel Processor Network, a Processor and a Method for Solving Differential Equations in a Parallel Processor Network', patent application PCT/FI02/00370, priority date 30.4.2001.

[31] A. Paasio, M. Laiho, 'An Analog-to-Digital Converter, a Current Scaler and a Method for Controlling Function of the Current Scaler', patent application PCT/FI03/00115, priority date 18.2.2002.

Chapter 2

Towards Parallel Architectures

There are basically two ways of increasing the computing speed of a processing device: either an individual operation is executed faster or several operations are executed simultaneously (in parallel). Naturally these two methods can be also combined: simultaneous improvements in architecture and technology have yielded great improvements in processing power. This chapter overviews different processor architectures that utilize parallel computing. The purpose is to illustrate, on the basis of the brief descriptions, the diversity of the different approaches. The processor types described in this chapter are meant to do different tasks and they handle different types of data. Therefore, comparisons between the approaches are not made.

Section 2.1.2 shows that general-purpose processors also utilize parallel computing. However, increasing the number of operations that are executed in parallel is rather involved. In applications in which the data is manipulated so that only local operations are performed on the data, processing in parallel is easier. In this case, the processors can be arranged in an array and the memory can be distributed among the processors. Sections 2.3-2.6 describe different architectures that are highly parallel and in which most available operations are limited so that data is manipulated only locally. In applications in which data can be processed with a massively parallel system, and in which the accuracy requirements are moderate, it may be beneficial to utilize analog parallel processing [1]. Sections 2.4 and 2.5 show examples of processor architectures that are designed for analog and/or mixed-mode computing. Section 2.6 overviews the cellular neural/nonlinear network (CNN) paradigm and CNN realizations. CNN theory can be used to describe many local processing tasks but it also proposes an architecture for realizing an array processor.

2.1 Computing Speed of General Purpose Processors

This section describes some of the efforts that designers are taking to improve the processing speed of general purpose processors (e.g. microprocessors and DSPs). The development of silicon processes and the introduction of new logic gates are key factors in improving the execution time of a single operation. Architectural improvements are essential when parallelism of computing is increased. In Subsection 2.1.3, a new concept called network on chip (NoC) is described. An NoC is a network of processing resources and memory that communicate asynchronously using a standardized protocol.

2.1.1 Computing Speed due to Development of Technology

The scaling down of minimum transistor sizes that has been ongoing for decades has enabled the use of higher clock frequencies, due to, for example, reduced parasitic capacitances. Consequently, the execution time per operation has shortened. Together with architectural improvements, this has almost guaranteed great improvements in processing performance. Today, the maximum clock speeds allowed by processes are difficult to achieve. Processor designers are devoting a lot of effort to planning a clock distribution that guarantees synchronous operation [2]. De-skew circuits, for example, are used locally to adjust the skew to cope with processing or temperature variations [3], while on-chip RC-filters are used to reduce clock jitter [4]. As ASIC designers may not have the means or resources with which to plan the clock distribution so precisely, it is getting more and more difficult to fully utilize the promise of the improving silicon processes.

2.1.2 Parallel Operation in General Purpose Processors

General purpose processors are traditionally based on serial data processing. However, several different means are used to make the inherently serial processing devices more parallel. An important concept utilized is the design of instruction sets towards high instruction level parallelism. In such a design method, the appearance of control operations and data accesses are predicted and speculated and executed in parallel. In IA-64, six instructions per clock can be run in parallel [5]. If the instruction level parallelism is low, chip multiprocessing can be used. In Reference [6], a single-chip multiprocessing system with eight processor cores was described. A protocol-based interconnect was used to connect the parts of the multiprocessing system. In Reference [7], a two-processor SIMD-type chip utilizing chip multiprocessing was shown

with a shared dual-port cache. In the chip multiprocessor systems, a lot of effort has to be put into memory sharing. In supercomputers, a very large number of processors is used in parallel. In Reference [8], a supercomputer is composed of two-processor SoCs that are assembled in a three-dimensional formation with six bi-directional links between each computing node. The architecture targets processor counts of 10000-100000 per system. In inherently serial general-purpose processors, the applications and instruction sets must be coded so that full use can be made of the permitted parallel operations.

2.1.3 Network on Chip

2.1.3.1 Motivation

Global communication on chip is getting more expensive in terms of processing speed and power consumption. This is because the delays of global interconnects are becoming more and more significant relative to gate or local interconnect delay when proceeding towards smaller gate dimensions [9]. The design of synchronous systems with high global clock speeds is also becoming increasingly difficult because the TOF of a signal is upper bounded [10]. The global interconnects can be made faster if repeaters are used but this increases the power consumption. Therefore, it would be beneficial to use parallel processing units that avoid communicating globally.

2.1.3.2 Network on Chip Platforms

Network on chip (NoC) is a platform-based design method for building systems on chips [11]. It is an attempt to bring the potential of the development of technology and the promise of massive parallel processing more easily available. In a platform based NoC design different abstraction levels (layers) are used to describe the communication among computing resources. Once a platform has been constructed, the designer can rely on the pre-determined layers. Compared to an ad hoc approach, the design task is greatly simplified. The processing resources and communication resources can be built on a mesh, for example, leading to a scalable architecture and predictable electrical properties. In [12] a packet switched platform is used for the communication between computing resources. In this design a communication switch is connected to a computing resource and four neighbor switches. Different network topologies and memory arrangements are under active study and performance models are being developed to compare, for example, different memory module organizations [13]. The concept of NoC is still quite a new research topic, but, in addition to

many conceptual propositions, hardware realizations also exist. In Reference [14], for example, an NoC with an ARM8 processor and 21 parallel satellite processors was realized on a Pleiades platform [15]. The processors are locally synchronous and global communication is asynchronous. The communication between the resources is based on a protocol. It is likely that, in the DSM era, large ASICs will not be built from scratch but on a platform that makes the utilization of silicon resources and design of data transfer easier for the designer.

2.2 Bit-Serial Processing

In bit-serial processing, words of different lengths are manipulated serially. The advantages of bit-serial processing are that the word length is not fixed and that it only requires hardware to compute bit operations with one-bit output. Therefore, compared to an ALU of full word length the amount of hardware is greatly reduced. However, the hardware is reduced at the cost of processing speed. The speed reduces with the word length because processing words with a length larger than one bit takes several clock cycles. Also, even if the range of available operations is broad, the more complex operations take many more clock cycles to execute. Multiplication, for example, takes a lot longer than addition.

2.2.1 Computational RAM

In a computational RAM (C•RAM)[1], the processing elements are distributed within memory [16]. The basic idea is to utilize the high internal bandwidth of a digital memory by inserting processing elements to one or more columns of memory. This leads to a small area penalty compared to a conventional memory. This way, a compact SIMD-type computer can be constructed. Figure 2.1 shows a bit-serial processing unit in a C•RAM 2.1. Each memory can access a certain address space, in this case a column of memory. The ALU is made of a multiplexer in which an eight-bit global instruction is multiplexed to a one-bit output using the contents of registers X, Y and the output of the memory. The output of the ALU can be written to write-enable register Z, to memory and/or it can be shifted left or right. The write-enable register is used to write the memory conditionally and the shift left and right operations enable communication with neighboring processing elements. The applications of a C•RAM are in, for example, DSP, image processing and computer graphics. Currently, more

[1] Also known as DSP-RAM

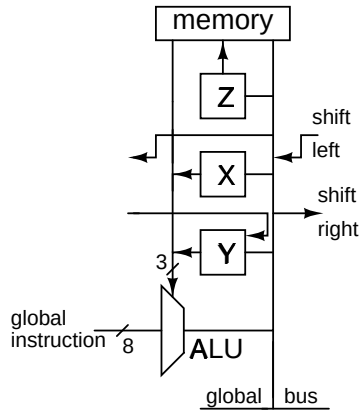

Figure 2.1 Bit-serial processing unit in a C•RAM [16].

computing power in a more compact area is being sought by considering 3D C•RAMs in which individual C•RAMs are stacked and connected with 3D vias [17].

2.2.2 Near Sensor Bit-Serial Image Processing

Near sensor image processing (NSIP) [18] aims to perform computationally intensive processing operations next to the sensor. This way only the significant image data is conveyed out of the chip as feature vectors and the I/O bandwidth is greatly reduced. Also in NSIP, the image acquisition can be locally controlled by the logic, i.e. adaptive exposition times are possible. In the NSIP image processor of [18], each sensor is accompanied by a bit-serial processing unit. The core of the bit-serial NSIP is the sensor, the output of which is compared to a threshold value and one bit of information is conveyed to the logic. Since the readout is nondestructive, consecutive binary readouts are possible. Consecutive samples of binary data yield gray-scale information. Each processing element (PE) is 4-connected to its nearest neighbors[2]. Masked logic AND operations can be performed for the neighborhood. The logic allows for fast intra-PE asynchronous propagation. In addition to this, each PE has a bit-serial processor and digital memory. Using the combination of the sensor and PE, various image processing operations are possible. Available operations include gray scale morphology, histogram equalization, global OR, pixel-level A/D conversion and adaptive exposure time.

[2] See Section 2.6 for definitions.

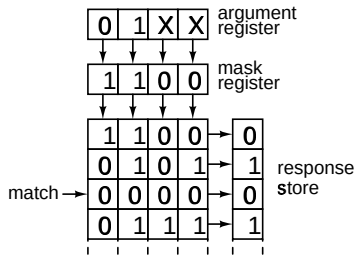

Figure 2.2 Search operation in a content addressable memory.

2.3 Content Addressable Memory

In a conventional memory data is accessed on the basis of address, independently from the data stored in the memory. A content addressable memory (CAM) works so that given a data word as input, a CAM provides one or more addresses in which a fully or partially matching data is located. The search operation of a CAM, explained, for example, in [19], is illustrated in Figure 2.2. The argument register contains the data to be matched with contents of the memory, while the mask register is used to identify which bits take part in the comparison. The contents of the memory are matched and the responses are stored in a register. If there are multiple hits, a resolver decides which hit is chosen. The matching can be performed fully parallel for each data word, serially, or, by software. Naturally, the fully parallel realization is the fastest. A CAM is a convenient way of realizing, for example, lookup tables. Also, they have been designed for high speed switching/routing in communications applications. In [20], a fully parallel 2.5Mb CAM, for example, was realized for use in a 2.5Gb/s ATM. In a content addressable memory, parallel write functions can also be included to complement the CAM search functions. These features extend the range of applications of these associative processors to different image processing applications [21]. DTCNN (see Section 2.6) and morphological operations, for example, can be performed by combining search, read, write, data transfer and hit-flag operations.

2.4 Artificial Neural Networks

A neural network is a combination of simple processing units in parallel that can store experiential data (obtained through learning) in terms of strengths of connections (weights) between the processing units [22]. Artificial neural networks (ANNs) are man-made realizations of neural networks. Most commonly ANNs are realized with software that is run in conventional computers, but dedicated hardware can also

be used. The definition of neural networks shown above is rather general and applies to many different parallel processing architectures, such as those in Section 2.6 (CNN), for example, if the weights are selected by learning. As the number of synaptic connections on chip increases, so does the number of weights that have to be learned and stored. This impedes analog implementations especially, because of the lack of compact analog memories with fast write operations. Mixed-mode implementations are potential alternatives to digital realizations. In [23], the control as well as the long-term storage of weights is digital. The computation is analog and the analog weights are periodically refreshed using D/A converters. The learning of weights can be performed either outside the chip or on chip. In [24], for example, the back propagation algorithm was realized on chip. ANNs are a powerful tool in, for example, the analysis of outputs of sensory arrays. Due to their ability to generalize, real-time classification of multi-channel data is possible.

2.5 Analog/Mixed-Signal Array Processing

This section overviews some analog/mixed signal array processor types that have been proposed for real time processing. In an array processor a multiple of processing units is arranged into an array. Usually the array is two dimensional. A large spectrum of different approaches has been proposed. The programmability, for example, and the type of data that can be processed, varies. This makes a direct comparison of these array computers difficult.

2.5.1 SIMD Array Composed of Analog Microprocessors

An analog microprocessor (AμP) attempts to combine digital programmability with analog arithmetic. The elementary instructions are as in a digital μP, but they are realized using analog means. In Reference [25], analog registers, analog ALU and analog comparison operations were realized using analog switched-current memories and processing elements. A SIMD-type processor can be constructed by arranging AμP units in an array. References [26] and [27] describe a 21×21 SIMD array composed of AμPs. An individual AμP can transfer data to/from its nearest neighbors. Also, a digitally programmable multiplier based on binary weighted currents is included in each AμP. Therefore, convolution-type operations are possible. The SIMD array also features easy digital programmability and an image sensor is included in every AμP. Consequently, the AμP SIMD array is another candidate for near sensor image processing.

2.5.2 Analog Multiple Instruction Multiple Data Processing

An example of analog multiple instruction, multiple data (MIMD) image processor is given in [28]. In this design an 80×78 sensor array is partitioned into four sub-arrays of 40×39. Each sub-array of sensors can be read simultaneously, each sensor has three outputs. One processing unit is allocated for each sub-array. Convolution operations with 3×3 masks are computed by shifting an active pixel area over the sub-array. Selected currents of the pixels in the active area are summed in row and column directions. The processing unit performs simultaneously five different operations to the input currents that are provided by the sub-array. Large convolution masks can be processed using a combination of 3×3 convolution masks. However, the elements of the convolution masks cannot be independently programmed. Independent of the mask size, the programmability is limited to two different sub-fields of coefficients.

2.5.3 Mixed-Mode Array Processor Using A/D/A Multipliers

Reference [29] depicts a mixed-mode array processor in which each processing unit is 4-connected to its nearest neighbors with an analog bus. A processing unit resembles that of the AμP. An ALU, a sample and hold unit and switches are included. What is different in this design is that the ALU performs multiplication and division with a combination of A/D and D/A converters. By changing the reference voltages of the ADC and DAC, programmable multiplication and division are possible. The fact that the ALU in this design only has one multiplier/divider slows down the convolution operations.

2.5.4 Mixed-Mode Array Processor for Vector Matrix Multiplication

A mixed-mode array processor for performing vector-matrix multiplication was shown in [30]. In this design, the data is stored in DRAM cells in a bit-parallel form, i.e. one row of data in the matrix is represented by the word length of rows of the DRAM. In this design, the data is stored digitally, analog one-bit multiplications and charge summations are performed and the results are combined with an ADC. The vector is unary coded. The computation is based on accumulating the charge that results from a multiplication of two one-bit binary numbers. The charge on each row is accumulated as a result of a multiplication with a unary vector. A delta-sigma ($\Delta\Sigma$) modulator is used to combine the partial results of the unary vectors. The final result is obtained by combining the results of the $\Delta\Sigma$ ADCs.

2.5.5 Mixed-Mode Nonlinear Oscillator Networks

Reference [31] proposed a nonlinear oscillator network hardware for performing resistive fuse computing to extract regions of image. The 8-connected [3] nonlinear oscillator cells are arranged into a grid, each corresponding to a pixel of image. A region of image is identified by coherent firing of the oscillators. Each oscillator cell is provided with nonlinearly modulated current sources. Using pulse width and phase modulation techniques to switch the nonlinearly modulated current sources ON and OFF, an input voltage is nonlinearly mapped into an output voltage. Reference [31] also suggests the use of pulse modulation techniques to realize the programmable weights. Reference [32] describes a nonlinear oscillator network realization with 50×50 oscillators in parallel. In the realized circuit, the weights were realized using digital hardware.

2.5.6 Dedicated Analog Computing Arrays for Image Processing

Reference [33] suggests that an array processor can be built of processing units with several dedicated computing cores. The processing tasks are separated into different categories and dedicated circuitry is optimized for each category. According to [33], using this approach is more likely to lead to more compact hardware than using a general-purpose analog computational core, since different processing tasks may require different qualities from the hardware. Convolution operations with bipolar I/O, for example, require less accurate realizations than circuits for computing gray-scale convolutions. When using separate cores, the accuracy of the core for processing bipolar data could be relaxed and a higher programming and computing speed could be obtainable. Circuit realizations could also be optimized by reducing programmability. Even, for example, if the programmability of a convolution mask were to be reduced, many important operations could still be computed. In Reference [33], different computing cores were suggested for bipolar I/O convolutions, linear spatial filters, ranked order spatial filters and nonlinear fuzzy-type processing (for example, gradient and tent map). If all these computing cores were included in a processing unit, the area would grow considerebly. Therefore, the type and quantity of computing cores should be defined by the requirements of the particular application to be used. In Reference [34], processing units composed of dedicated computing cores were suggested for performing video sequence segmentation. Dedicated processing cores can be made of simple analog primitives. Consequently, low operating voltages can be used.

[3] See Section 2.6 for definition.

2.6 Cellular Neural/Nonlinear Networks

The cellular neural/nonlinear network [35] paradigm is a hardware-oriented theory that can very conveniently describe many operations in which data is manipulated locally. CNNs are composed of locally connected processing units (cells) arranged in a regular grid. The definition of a neural network given in Section 2.4 applies to CNNs if the weights are chosen by learning methods. Otherwise it is customary to refer to cellular nonlinear networks. This section gives some basic definitions used in array computing, followed by CNN theory and realizations.

2.6.1 Definitions

In this section, some general definitions relating to array processing are given. The definitions can be used in the context of CNNs, but they can be applied to other types of array computers as well. An individual processor that the processor grid is composed of can be called, for example, a cell, a processing unit or an elementary processing unit, depending on the realization. In this section, it is denoted a cell.

2.6.1.1 Array Dimension and Type

The cells in array processors are usually arranged in one- or two-dimensional grids because planar silicon processes are not well suited for grids of higher dimensions. Cells in two-dimensional arrays can be arranged in grids of different typologies, such as rectangular and hexagonal. Out of these, it is most common to arrange the cells in rectangular grids. In this thesis, only rectangular grids are considered. Therefore, an individual cell among an $M \times N$ network of cells can be identified by $C_{i,j}$ where $i \in [1,M]$; $j \in [1,N]$. M is the number of rows and N is the number of columns.

2.6.1.2 Neighborhood and Connections Between Cells

The cells in an array processor can usually interact directly only with selected cells in their close proximity, i.e. neighborhood cells. The neighborhood of cell $C_{i,j}$ is a collection of cells that are within a sphere of influence of cell $C_{i,j}$, the radius of influence being R_n. The R_n-neighborhood of cell $C_{i,j}$ consists of cells $C_{m,p}$ where $m = i + k - (R_n + 1)$, $p = j + l - (R_n + 1)$, $k \in [1, 2R_n + 1]$ and $l \in [1, 2R_n + 1]$. If cell $C_{i,j}$ is directly connected to all cells in the neighborhood, it is 8-connected. In a 4-connected network cell $C_{i,j}$ is connected to those neighborhood cells that are in the same row and/or column as cell $C_{i,j}$. Other connectivity patterns are also possible.

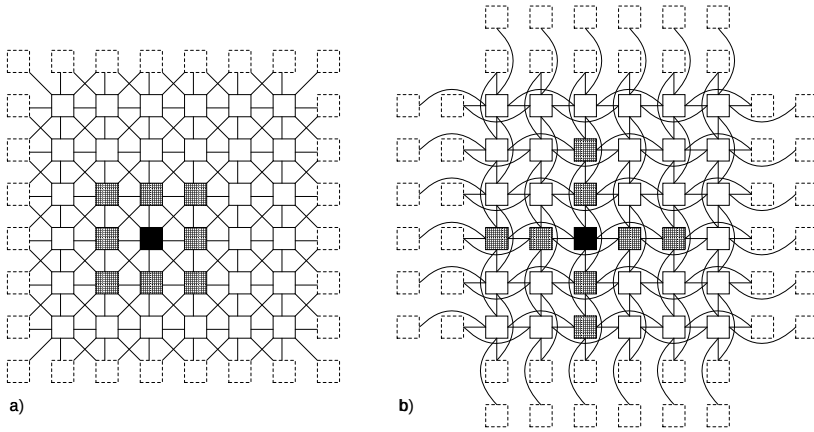

Figure 2.3 Figure 2.3a) shows a first-neighborhood, 8-connected network. Figure 2.3b) shows a network in which the cells are 4-connected to cells in a second neighborhood. The gray cells are directly connected to the black cell. The dashed rectangles are border cells.

Figure 2.3a) shows a network in which the cells are 8-connected to the first neighborhood. The black cell, for example, is directly connected to the gray cells. Figure 2.3b) depicts a 4-connected, second-neighborhood network. Again, the gray cells are directly connected to the black cells. Looking at Figure 2.3b) it can be seen that direct connections to a large neighborhood lead to problems in the wiring. In addition to the $M \times N$ active cells, conditions at the border of the cell grid have to be determined. The dashed rectangles in Figures 2.3a) and b) denote the border cells (virtual cells). Boundary cells are active grid cells that are directly connected to border cells, while edge cells are boundary cells that are located next to border cells. Active-grid cells that are not boundary cells are denoted regular cells. The border cells are used to create a correct boundary condition for edge cells. Cells can also have an indirect effect on each other. This occurs in operations in which information propagates through the directly connected cells. Global interaction is possible through propagation of local events that can proceed in the processor grid.

2.6.1.3 Convolution-Type Operations

Many array computers can perform convolution-type operations in which the size of the convolution mask is defined by the direct connections to the neighborhood. The convolution mask (template) for an 8-connected, first-neighborhood network is

$$\mathbf{A}_{i,j} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix}, \tag{2.1}$$

where $A_{k,l}$ are the interconnection strengths (weights). The weights can be programmable or fixed. If $\mathbf{A}_{i,j}$ is the same for all i, j, the convolution mask is said to be space-independent. In hardware realizations, space-independency of the coefficients is desirable because, in that case, collective programming of weights is possible.

2.6.2 Theory of Cellular Neural/Nonlinear Networks

2.6.2.1 Continuous-Time CNN (Original Model)

A cellular neural network (CNN) [35] can be described as a collection of regularly arranged identical analog processing elements (cells). The cells are directly connected to selected cells in the neighborhood. Each cell $C_{i,j}$ has a state $x_{i,j}$ and a constant input $u_{i,j}$. These, together with directly connected neighborhood cells, define the output of a cell, namely $y_{i,j}$. Assuming first neighborhood and that the templates are programmed space-independently, the dependence of the cell state on the outputs and inputs of neighborhood cells can be expressed by the cell state equation [35]

$$\begin{aligned} C\frac{dx_{i,j}(t)}{dt} &= -\frac{1}{R}x_{i,j}(t) + \sum_{k=1}^{3}\sum_{l=1}^{3} A_{k,l}y_{m,p}(t) \\ &\quad + \sum_{k=1}^{3}\sum_{l=1}^{3} B_{k,l}u_{m,p}(t) + I \end{aligned}, \tag{2.2}$$

where $m = i + k - 2$, $p = j + l - 2$. Terms $A_{k,l}y_{m,p}(t)$ and $B_{k,l}y_{m,p}(t)$ describe the strength by which cell $C_{m,p}$ affects cell $C_{i,j}$. $\mathbf{A}$ and $\mathbf{B}$ are space-independent feedback and feedforward templates. If the feedforward template is zero, the CNN is called autonomous. Also shown in Equation 2.2 are constant terms C and R and a constant biasing term I. In [35], the activation function (relationship between $x_{i,j}$ and $y_{i,j}$) is a piecewise linear function

$$y_{ij}(t) = f(x_{i,j}(t)) = \frac{1}{2}\left[\left|x_{ij}(t)+1\right| - \left|x_{ij}(t)-1\right|\right]. \tag{2.3}$$

In analog circuit realizations, the output nonlinearity resembles a unity gain sigmoid

$$y_{ij}(t) = f(x_{i,j}(t)) = \frac{2}{1+e^{-4x_{i,j}(t)}} - 1, \tag{2.4}$$

which is a continuous function that approximates Equation 2.3. Commonly used border conditions for an $M \times N$ network with $i \in [1, M]$, $j \in [1, N]$ are defined as follows:

If the states, inputs and outputs of the border cells are fixed, the border condition is denoted fixed (Dirichlet) boundary condition. The border-cell states in a cyclic (toroidal) boundary condition are defined as $x_{i,0} = x_{i,N}$, $x_{i,N+1} = x_{i,1}$, $x_{0,j} = x_{M,j}$ and $x_{M+1,j} = x_{1,j}$. The inputs and outputs of the border cells are defined similarly. In zero-flux (Neumann) boundary condition the states of border cells are $x_{i,0} = x_{i,1}$, $x_{i,N+1} = x_{i,N}$, $x_{0,j} = x_{1,j}$ and $x_{M+1,j} = x_{M,j}$. Again, the inputs and outputs of the border cells are defined similarly; for example, $y_{i,0} = y_{i,1}$.

2.6.2.2 CNN Universal Machine

In a CNN universal machine (CNNUM) [36] the capabilities of the original CNN are improved by adding new functionalities to the cell and by introducing a global analogic programming unit (GAPU). Digital memory, Boolean logic and control logic are added to the cell. Also, analog memories and the means to perform functions to the contents of the analog memories are included in the cell. The GAPU contains means to store different templates and logic programs. It also stores different switch configurations and contains a unit that controls all global operations. The GAPU can be programmed to run sets of templates and logic operations. Therefore, a CNNUM can be used to run complete CNN algorithms.

2.6.2.3 Polynomial CNN

In a polynomial CNN, the couplings between cells can also be polynomial functions of cell input and output [37]. For example, the state of a polynomial CNN with first- and second-order polynomial terms of activation function $f(x_{ij})$ is defined as

$$\begin{aligned} C\frac{dx_{ij}(t)}{dt} &= -\tfrac{1}{R}x_{ij}(t) + \textstyle\sum_{k=1}^{3}\sum_{l=1}^{3}\left[A_{k,l}^{(1)}y_{m,p}(t) + A_{k,l}^{(2)}y_{m,p}^{2}(t)\right] \\ &\quad + \textstyle\sum_{k=1}^{3}\sum_{l=1}^{3}\left[B_{k,l}^{(1)}y_{m,p}(t) + B_{k,l}^{(2)}y_{m,p}^{2}(t)\right] + I \end{aligned} , \tag{2.5}$$

where $\mathbf{A}^{(1)}$, $\mathbf{B}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{B}^{(2)}$ are the first and second order polynomial feedback and feedforward templates.

2.6.2.4 Full Signal Range CNN

Reference [38] proposed a CNN to be realized so that both the state and output of a cell are truncated between -1 and 1. The use of this full signal range (FSR) CNN somewhat changes the CNN dynamics. The FSR model has a couple of important benefits when it comes to the realization of CNNs. Firstly, the dissipative term $-\frac{1}{R}x_{i,j}(t)$ in Equation 2.2 can be realized by subtracting the self-feedback term $A_{2,2}$ by unity. Secondly, since

the state and output are equal, there is only one time constant in the cell. Furthermore, the whole state swing can be reserved for states between -1 and 1.

2.6.2.5 Discrete Time CNN

Reference [39] proposed a discrete-time CNN (DTCNN). It was defined as having continuously valued inputs and weights and a bipolar output. A CNN that performs discrete-time integration can be defined also in a more general way. The time-discretized rate of change state of a first-neighborhood FSR DTCNN cell can be defined as

$$\Delta x_{i,j}(n) = \sum_{k=1}^{3}\sum_{l=1}^{3}[A_{k,l}x_{m,p}(n) + B_{k,l}u_{m,p}(n)] + I. \tag{2.6}$$

The state $x(n+1)$ is determined by updating $x(n)$ with a fraction of $\Delta x_{i,j}(n)$. This can be realized in an analog discrete-time CNN or in a digital CNN. In a digital DTCNN the state can be updated using several different digital integration methods such as Euler's method, Heun's method or Runge-Kutta method[4].

2.6.3 CNN Realizations

2.6.3.1 Analog CNN

CNNs have been realized on the basis of both the original model and the FSR model. Reference [40] describes a 20×20 analog CNN that is realized using the original CNN model. The first-neighborhood, 4-connected cells interact via differential transconductance multipliers. However, the largest realizations so far are based on the FSR model. Reference [41] describes a CNN realization with 128×128 cells. One transistor, four-quadrant multipliers are used to realize the couplings between cells. Image sensors are also included in the cells and therefore the chip is capable of NSIP-type computing. There are also more exotic CNN realizations. An analog CNN realization that has two layers of active cells is described in Reference [42]. The layers are coupled and the time-constant of the other layer can be programmed. The combination of the two coupled layers can model complex spatio-temporal dynamics.

2.6.3.2 Digital Emulated CNN

A CNN can also be realized digitally. In digital realizations, the cells are less compact, and therefore the number of cells is smaller than in analog realizations. The benefits of digital realizations are their suitability for mainstream digital technologies, good

[4] See Section 4.1.2 for more details on these integration methods.

accuracy and robust digital storage. An example of a digital CNN realization is shown in Reference [43]. It emulates the FSR CNN by integrating spatial convolutions with Euler's method. Twenty four cells are included on chip. Some on-chip memory is included in the emulated CNN so that the I/O bottleneck between the memory chip and CNN chip is eased. Due to variable precision, the computing speed can be traded for precision.

2.6.3.3 Positive Range High Gain CNN

A positive range high-gain CNN uses a positive range high-gain sigmoid (threshold function) as the activation function. Cells in this kind of CNN have bipolar I/O, but the templates are continuously programmable. It can compute threshold logic operations for neighborhood cells. If there is no need for gray-scale operations, having the threshold function as the activation function greatly simplifies the analog design . A multiplier, for example, can be composed of a current source and switches [44].

References

[1] E. A. Vittoz, 'Analog VLSI Signal Processing: Why, Where and How?', *Journal of VLSI Signal Processing*, vol. 8, pp. 27-44, 1994.

[2] P. J. Restle, et al., 'A Clock Distribution Network for Microprocessors', *IEEE Journal of Solid-State Circuits*, Vol. 36, pp. 792-799, 2001.

[3] S. Rusu, S. Tam, 'Clock Generation and Distributions for the First IA-64 Microprocessor', *Proc. of the IEEE Solid-State Circuits Conference*, San Francisco, USA, 2000, pp. 176-177, 448.

[4] N. A. Kurd, et al., 'Multi-GHz Clocking Scheme for Intel(R) Pentium(R) 4 Microprocessor', *Proc. of the IEEE Solid-State Circuits Conference*, San Francisco, USA, 2001, pp. 404-405.

[5] G. Singer, S. Rusu, 'The First IA-64 microprocessor: a Design for Highly-Parallel Execution', *Proc. of the IEEE Solid-State Circuits Conference*, San Francisco, USA, 2000, pp. 422-423.

[6] L. A. Barroso, et al. 'Piranha: a Scalable Architecture Based of Single-Chip Multiprocessing', *Proc. of the* 27^{th} *International Symposium on Computer Architecture*, Vancouver, Canada, 2000, 282-293.

[7] A. Kowalczyk, et al. ' The First MAJC Microprocessor: A Dual CPU System-on-a-Chip', *IEEE Journal of Solid-State Circuits*, Vol. 36, pp. 1609-1616, 2001.

[8] G. Almasi, et al., 'Cellular Supercomputing with System-On-A-Chip', *Proc. of the IEEE Solid-State Circuits Conference*, San Francisco, USA, 2002, pp. 196-197.

[9] R. Yung, S. Rusu, K. Shoemaker, 'Future Trend of Microprocessor Design', *Proc. of the* 28^{th} *European Solid-State Circuits Conference*, Florence, Italy, Sept. 2002, pp. 43-46.

[10] D. Sylvester, K. Keutzer, 'A Global Wiring Paradigm for Deep Submicron Design', *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 19, pp. 242-252, 2000.

[11] L. Benini, G. De Micheli, 'Networks on Chips: a New SoC Paradigm', *Computer*, Vol. 35, pp. 70-78, 2002.

[12] S. Kumar, et al., 'A Network on Chip Architecture and Design Methodology', *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, Pittsburg, USA, pp. 105-112, 2002.

[13] M. Forsell, 'Scalable High-Performance Computing Solution for Networks on Chips', *IEEE Micro*, Vol. 22, pp. 46-55, 2002.

[14] H. Zhang, et al., 'A 1V Heterogeneous reconfigurable Processor IC for Baseband Wireless Applications', *Proc. of the IEEE Solid-State Circuits Conference*, San Francisco, USA, 2000, pp. 68-69, 448.

[15] M. Sgroi, et al., 'Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design', *Proceedings of the Design Automation Conference*, Las Vegas, USA, pp. 667-672, 2001.

[16] D. Elliott, et al. 'Computational RAM: implementing processors in memory' *IEEE Design & Test of Computers* , Vol. 16, pp. 32-41, 1999.

[17] J. Koob, et al., 'Design of a 3D Fully-Depleted SOI Computational RAM', *Proc. of the* 28^{th} *European Solid-State Circuits Conference*, Florence, Italy, Sept. 2002, pp. 135-138.

[18] J-E Eklund, C. Svensson, A. Åström, 'VLSI Implementation of a Focal Plane Image Processor - A Realization of the Near-Sensor Image Processing Concept',

IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 4, pp. 322-335, 1996.

[19] T. P. Haraszti, *CMOS Memory Circuits*, Kluwer: Boston, 2000, 54-61.

[20] F. Shafai, et.al, 'Fully Parallel 30-Mhz, 2.5-Mb CAM', *IEEE Journal of Solid-State Circuits*, Vol. 33, pp. 1690-1696, 1998.

[21] T. Ikenaga, T. Ogura, 'A Fully Parallel 1-Mb CAM LSI for Real-Time Pixel-Parallel Image Processing', *IEEE Journal of Solid-State Circuits*, Vol. 35, pp. 536-544, 2000.

[22] S. Haykin, *Neural Networks: a Comprehensive Foundation*, Prentice-Hall: Upper Saddle River, p. 2, 1999.

[23] J. Schemmel, et al., 'An Integrated Mixed-Mode Neural Network Architecture for Megasynapse ANNs', *Proc. of the International Joint Conference on Neural Networks*, Honolulu, USA, 2002, pp. 2704-2709.

[24] F. Stupmann, R. Rode, N. Schmidt, G. Geske, 'Implementation of Learning in Continuous Analog Circuitry', *Proc. of the International Joint Conference on Neural Networks*, Honolulu, USA, 2002, pp. 733-736.

[25] P. Dudek, P. Hicks, 'A CMOS General-Purpose Sampled-Data Analogue Microprocessor', *Proc. of the IEEE International Symposium on Circuits and Systems*, Geneva, Switzerland, Vol. 2, pp. 417-420, 2000.

[26] P. Dudek, P. Hicks, 'A CMOS General-Purpose Sampled-Data Analogue Microprocessor', *Proc. of the IEEE International Symposium on Circuits and Systems*, Sydney, Australia, Vol. 4, pp. 490-493, 2001.

[27] P. Dudek, P. Hicks, "A General-Purpose CMOS Vision Chip with a Processor-Per-Pixel SIMD Array", *Proc. of the* 27^{th} *European Solid-State Circuits Conference*, Willach, Austria, pp. 228-231, 1999.

[28] R. Etienne-Cummings, Z. Kevork Kalayjian, D. Cai, "A Programmable Focal-Plane MIMD Image Processor Chip", *IEEE Journal of Solid-State Circuits*, Vol. 36, No. 1, pp. 64-73, 2001.

[29] D. A. Martin, H. Lee, I. Masaki, 'A Mixed-Signal Array Processor with Early Vision Applications', *IEEE Journal of Solid-State Circuits*, Vol. 33, pp. 497-502, 1998.

[30] R. Genov et al., “A 5.9mW 6.5GMACS CID/DRAM Array Processor”, *Proc. of the* 28^{th} *European Solid-State Circuits Conference*, Florence, Italy, pp. 715-718, Sept. 2002.

[31] H. Ando, T. Morie, M. Nagata, A. Iwata, ’A Nonlinear Oscillator Network for Gray-Level Image Segmentation and PWM/PPM Circuits for Its VLSI Implementation’, *IEICE Trans. Fundamentals*, Vol. E83-A, pp. 329-336, 2000.

[32] H. Ando, T. Morie, M. Nagata, A. Iwata, “An Image Region Extraction LSI Based on a Merged/mixed-signal Nonlinear Oscillator Network Circuit”, *Proc. of the* 28^{th} *European Solid-State Circuits Conference*, Florence, Italy, pp. 703-706, Sept. 2002.

[33] A. Paasio, M. Laiho, A. Kananen, K. Halonen, ’An Analog Array Processor Hardware Realization with Multiple New Features’, *Proc. of the International Joint Conference on Neural Networks*, Honolulu, USA, 2002, pp. 1952-1955.

[34] A. Kananen, A. Paasio, M. Laiho, K. Halonen, ’CNN Applications From the Hardware Point of View: Video Sequence Segmentation’ , *International Journal of Circuit Theory and Applications*, Wiley, Vol. 30, 2002, pp. 117-137.

[35] L.O. Chua, L. Yang, “Cellular Neural Networks: Theory”, *IEEE Transactions on Circuits and Systems*, Vol. 35, pp. 1257-1272, 1988.

[36] T. Roska, L.O. Chua, ’The CNN Universal Machine: An Analogic Array Computer’, *IEEE Transactions on Circuits and Systems- II*, Vol. 40, pp. 163-173, 1993.

[37] R. Tetzlaff, R. Kunz, C. Ames, D. Wolf, “Analysis of Brain Electrical Activity in Epilepsy with Cellular Neural Networks (CNN)”, *Proc. of the European Conference on Circuit Theory and Design*, Stresa, Italy, pp. 1007-1010, 1999.

[38] S. Espejo, R. Carmona, R. Dominguez-Castro, A. Rodriguez-Vazquez , “A VLSI-oriented Continuous-time CNN Model”, *International Journal of Circuit Theory and Applications*, Vol. 24, no. 3, pp. 341-356, 1996.

[39] H. Harrer, J. A. Nossek, ’Discrete-Time Cellular Neural Networks’, *International Journal of Circuit Theory and Applications*, Vol 20, pp. 453-467, 1992.

[40] P. Kinget, M. Steyaert, ’An Analog Parallel Array Processor for Real-Time Sensor Signal Processing, *Proc. of the IEEE Solid-State Circuits Conference*, San Francisco, USA, pp. 92-93, 1996.

[41] G. Linan, et al., 'ACE16K: an Advanced Focal-Plane Analog Programmable Array Processor', *Proc. of the* 27^{th} *European Solid-State Circuits Conference*, Villach, Austria, pp. 216-219, 2001.

[42] R. Carmona, et al., 'A CMOS Parallel Array Processor Chip with Programmable Dynamics for Early Vision Tasks', *Proc. of the* 28^{th} *European Solid-State Circuits Conference*, Florence, Italy, pp. 371-374, 2002.

[43] P. Keresztes et al. 'An Emulated Digital CNN Implementation', *Journal of VLSI Signal Processing Systems*, Vol. 23, No. 2/3, 1999, pp. 291-303.

[44] A. Paasio, A. Kananen, K. Halonen, V. Porra, 'A QCIF Resolution Binary I/O CNN-UM Chip', *Journal of VLSI Signal Processing Systems*, Vol. 23, No. 2/3, 1999, pp. 281-290.

This page is intentionally left blank.

Chapter 3

Algorithm for Analyzing Brain Electrical Activity in Epilepsy

3.1 Background

Epilepsy is a chronic disease in which the patient suffers from seizures that weaken awareness and sensation and cause involuntary movements and spasms. About one percent of all people in the world have epilepsy [1]. The seizures are caused by temporary electrical disturbances in the brain. Epilepsy does not influence a person's intelligence; between seizures most people that have epilepsy can, in principle, live a normal life. However, surgical therapy and anti-epileptic drugs do not provide control over epilepsy for 25% of patients [1]. A sudden epileptic seizure while the patient is in a dangerous place or driving a car, for example, may lead to serious injury. If the onset of an epileptic seizure could be predicted minutes before the seizure, the patient would be given an opportunity to seek a safe place. A portable, possibly implantable, device might also help mitigate or prevent an epileptic seizure by automatic pharmacological or electrotherapeutic means.

3.1.1 Prediction of the Onset of an Epileptic Seizure

Various different methods that could be used to predict the onset of an epileptic seizure are being actively researched. Some of the research activity is overviewed in [1]. A common factor in these methods is that they require computation of demanding algorithms for a large set of data. The data set has to be large because it is difficult to distinguish meaningful and reliable characteristics from short data segments. Fig-

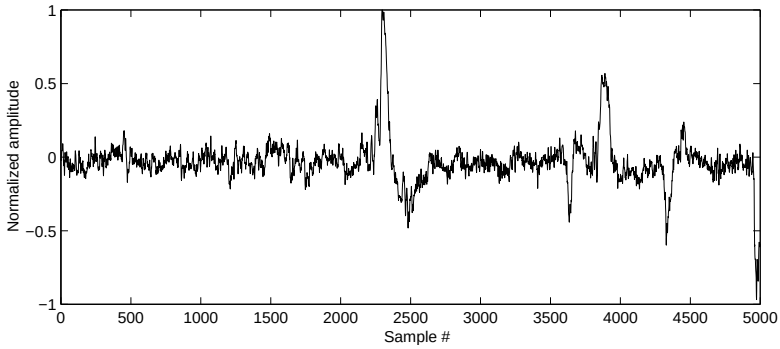

Figure 3.1 A segment of EEG with 5000 samples.

ure 3.1 shows a segment of 5000 samples of EEG during a seizure-free interval. The task of predicting an epileptic seizure could be characterized as extracting essential information from among noisy data without knowing exactly what the essential information is. The following description shows one prominent analysis method that aims at predicting the onset of an epileptic seizure.

Recent advances in research of epileptic seizures have shown that, by performing nonlinear time series analysis for invasively recorded samples of single-channel EEG, changes in brain electrical activity can be characterized and used for the prediction of the onset of an epileptic seizure [2], [3]. The analysis is based on the observation that, minutes before the seizure takes place, the epileptic area of the brain goes into a lower complexity state. The state is characterized by a measure called effective correlation dimension D_2^{eff}. In Reference [2], samples of EEG ($f_s = 173$Hz) were invasively recorded from the epileptic area, converted to digital, and low-pass filtered. Dimension D_2^{eff} was determined from half-overlapping segments of EEG with 30s duration. The number of samples MN that D_2^{eff} is determined from affects the maximum obtainable dimension so that $D_{2,max} \leq 2\log_{10}(MN)$. In [2], the number of samples MN was over 5000. This number of samples was chosen as a compromise between reliability of the dimension and computational effort. Vectors $X_m(i)$ of length m, $m \in [1,30]$ were formed from samples of EEG $v(i)$; $i \in [1,MN]$, using the method of delays

$$X_m(i) = [v(i), v(i+\tau), v(i+2\tau), ..., v(i+(m-1)\tau)] \tag{3.1}$$

with a fixed delay τ. The vectors were used to compute correlation integral

$$C_m(r) = \frac{1}{MN^2} \sum_{i=1}^{MN-1} \sum_{j=i+1}^{MN} H(r - \|X_m(i) - X_m(j)\|_\infty), \tag{3.2}$$

where H is the Heaviside (threshold) function and r is a radius that is assigned N_r values between lower bound r_l and upper bound r_u. The maximum norm was used to describe the difference between vectors $X_m(i)$ and $X_m(j)$. The derivative of the correlation dimension

$$C'_{m(r)} = \frac{d \log_2 C_m(r)}{d \log_2 r} \tag{3.3}$$

was used to compute an estimator of D_2^{eff}, the estimator being

$$D^* = \frac{1}{N_r} \sum_{r=r_l}^{r_u} C'_{m(r).} \tag{3.4}$$

The upper and lower bound r_l and r_u were selected on the basis of the derivatives of the correlation dimension with different values of m. Using the estimator D^* the effective correlation dimension was determined with

$$D_2^{eff} = \begin{cases} D^*, & \text{if } D^* \leq D_{2,max} \text{ and } N_r \geq 5 \\ D_u, & \text{else} \end{cases} . \tag{3.5}$$

Consecutive values of the effective correlation dimension were recorded. The dimension profile obtained was compared to a threshold. If the dimension remained below a threshold value for a certain number of samples, this was interpreted as a low complexity of the dimension, and, furthermore, a precursor of an epileptic seizure.

The correlation integral of Equation 3.2 is computationally demanding since a double summation of thresholded vector norms needs to be computed. Effective computation of the correlation integral using a workstation was discussed in [4]. However, in order to build a portable system that could be used for online prediction of epileptic seizures, a compact, low-power processing device is needed.

3.2 Analysis of EEG in Epilepsy using a CNN

Research into the analysis of EEG with a CNN is motivated by the fact that a CNN-based realization could potentially be used for predicting the onset of an epileptic seizure using a low-power, physically small system. Reference [5] proposed a polynomial CNN for analyzing brain electrical activity in epilepsy. The benefits of describing the EEG analysis algorithm using CNN theory are that the theory provides a convenient way of describing nonlinear dynamics and that the architecture is suitable for a parallel processor implementation. Also, fusion of information from several channels

is possible using CNN-type computers. This may be helpful in some analysis methods. Furthermore, many of the analysis methods need to be tuned for each patient individually; tuning may also be needed to cope with different levels of vigilance. Tuning can be performed by altering the CNN templates.

3.2.1 CNN-Based Algorithm for Prediction of Epileptic Seizures

References [5], [6] and [7] used a polynomial CNN to estimate D_2^{eff} with CNN-dimension D_{CNN}. The results showed that a CNN could produce reliable estimates of D_2^{eff}. In Reference [5], the CNN state equation was like that of Equation 2.5, but the feedforward templates were zero

$$C\frac{dx_{i,j}(t)}{dt} = -\frac{1}{R}x_{i,j}(t) + \sum_{k=1}^{3}\sum_{l=1}^{3}\left[A_{k,l}^{(1)}y_{m,p}(t) + A_{k,l}^{(2)}y_{m,p}^2(t)\right] + I. \qquad (3.6)$$

In Equation 3.6, symbols $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$ denote the linear and second order polynomial feedback templates. In References [6] and [7], the use of higher polynomial orders and the second neighborhood was also investigated. Reference [5] used the unity gain sigmoid of Equation 2.4 as the output nonlinearity, whereas, in [6] and [7], the piecewise linear output nonlinearity (Equation 2.3) was used. Different border conditions were also used in [5] and [6]. In all cases, CNN-dimension D_{CNN} was determined with data segments that contained 5184 samples of EEG. The data was written as the initial cell states of a 72×72 CNN. Consequent samples were loaded to the network pixel by pixel, row by row. After applying a transient with the CNN, the average of the outputs of the cells yielded the CNN-dimension D_{CNN}. The task of the CNN in determining D_{CNN} can be described as an extraction of essential information from among data in which the data units are not topographically arranged. Therefore, no input/output mapping is available for the determination of the weights. Other methods, such as evolutionary learning [8], were therefore used to determine the weights. The D_{CNN} templates have to be determined individually for each patient. In [7], it was shown that the learning of D_{CNN} templates can take more than a hundred thousand iteration rounds. To keep this time short, the hardware has to be capable of computing D_{CNN} rapidly.

3.2.2 System Requirements

Figure 3.2a) depicts a system that is composed of a processor array (CNN), a data acquisition unit including a low pass filter, a post-processing unit and a microcontroller

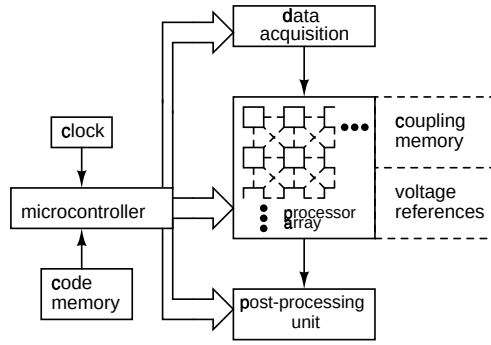

Figure 3.2 Complete system with processor array and additional components

for controlling the system. The processor array has access to voltage references and to memories that contain the weights. The data acquisition unit provides data to the processor array. CNN-dimension D_{CNN} is computed once every fifteen seconds since half-overlapping segments of 30s duration are processed. The post-processing unit is used to average the outputs of the cells and to interpret the resulting dimension profile. A long-term goal is to integrate the whole system on a single chip. However, currently the focus of the investigations is on the realization of the CNN processor. Based on the experiments of [5], [6] and [7], the realization of the CNN processor targets a capability of processing 72×72 data, linear, second- and third-order polynomial feedback terms 8-connected to first-neighborhood and programmable templates. The bias value I is chosen to be zero. Therefore, 27 multipliers are needed in each cell to realize the couplings. The cells are to have continuously valued states and outputs. The boundary condition is left a free design parameter. Also, the required accuracy is not specified. A miniaturized implementation of a processor with these qualities is studied in the rest of this thesis.

3.2.3 Characteristics of the Polynomial CNN

When a CNN is considered for the determination of D_{CNN}, it is important to stress some issues that affect the realization. Due to the many feedback connections, it is important that the dynamic behavior of the network is correct. Both delays and noise can affect this. Also, long retention times in the memories are required. In the following these characteristics are briefly overviewed.

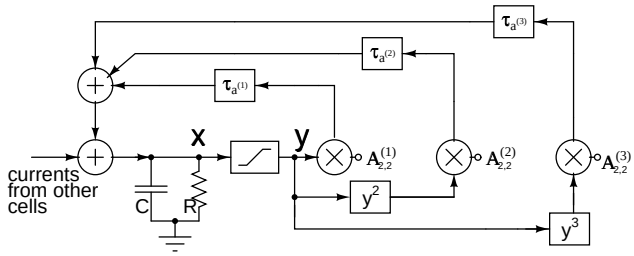

Figure 3.3 Feedback structure with delays.

3.2.3.1 Effect of Delays

Reference [9] shows how the stability of an ideal polynomial CNN depends on the feedback coefficients. In a physical realization, the stability may also be affected by delays due to parasitic capacitances. Figure 3.3 depicts the feedback structure of a polynomial type CNN with linear, second-order polynomial and third-order polynomial feedback terms. In Figure 3.3, $\tau_{a^{(1)}}$, $\tau_{a^{(2)}}$ and $\tau_{a^{(3)}}$ are used to describe delays in the feedback paths of the first-, second- and third-order terms. Cell state is marked by x, cell output by y and $A^{(1)}_{2,2}$, $A^{(2)}_{2,2}$ and $A^{(3)}_{2,2}$ are the first-, second- and third-order feedback coefficients. Blocks y^2 and y^3 produce second- and third-order polynomial terms of the cell output.

Ideally, the amount of feedback should be a function of cell output at any given time. However, there is a delay in the feedback path introduced by the multiplier and by the blocks that produce the polynomial terms. Consider that only one feedback template is used at a time, for example, $A^{(3)}_{2,2}$. In this case, all multipliers are controlled by the third-order polynomial function of cell output. Therefore, a first-order approximation can be made such that the delays are equal in all feedback paths. This is because the feedbacks of all terms are connected through similar paths and it is assumed that the delays of individual paths are equal. In reality, differences between delays of same-order feedback terms exist due to mismatches between transistor parameters that are caused by processing variations. In practice, these differences in delays are small compared to a typical time constant of a CNN; therefore the first-order approximation works well.

When different order terms with dissimilar feedback paths are combined, feedback terms can have significant differences in delays. A first-order approximation of the state current of a cell is

$$I_x(t) = I_{oc} + I_{a^{(1)}}(t - \tau_{a^{(1)}}) + I_{a^{(2)}}(t - \tau_{a^{(2)}}) + I_{a^{(3)}}(t - \tau_{a^{(3)}}), \tag{3.7}$$

where I_{OC} is the sum of currents from other cells, $I_{a^{(1)}}$ is a linear feedback current, $I_{a^{(2)}}$ is a second order polynomial feedback current, $I_{a^{(3)}}$ is a third order polynomial feedback current and $\tau_{a^{(1)}}$, $\tau_{a^{(2)}}$ and $\tau_{a^{(3)}}$ are the corresponding delays. If the differences of the delays are comparable to the time constant of the CNN, the dynamic behavior of the network can change dramatically. The stability of neural networks with different delays in the feedback paths has been studied in References [10] and [11], for example. One way of avoiding the problem of different delays is to increase the CNN time constant by using a larger state capacitor. Reference [12] used a CNN time constant that was 10 times larger than the parasitic time constant. A factor of 10 was considered large enough to make the parasitic time constants insignificant. In a polynomial CNN, the CNN time constant should probably be at least 10 times larger than the differences of delays in the feedback paths. This would require a lot of die area. Also, the delays of different order feedback terms could be matched using suitable delay elements. This alternative may not be straightforward to realize. In this thesis, the problem is solved by using discrete-time integration as descrebed in Chapter 4.

3.2.3.2 Transient Noise

When designing a large array of cells for gray-scale processing, noise in the operating voltage may also cause problems. At the start of transient, high current peaks may be drawn from the power source and the voltage level can fluctuate significantly. Such fluctuation can deflect the state evolution into a path that leads to a wrong energy minimum. Stability of the network may also be a problem. Therefore, the operating voltages should be designed to be as stable as possible. This can be achieved by using a high-performance voltage source and by designing the power distributing network so that high-frequency current components can be delivered. In a miniaturized system this may not be trivial. In this thesis, discrete-time integration (see Chapter 4) is applied. When discrete-time integration is used, transients are allowed to settle before each iteration so that the transients do not affect the integration.

3.2.3.3 Memory Retention Time

Since a sampling frequency of 173Hz is used for loading the initial states to the 72×72 network, an initial state has to be stored for up to 30 seconds. In order to store gray-scale information for such a long time in the analog domain, the size of the memory element becomes large. Data can also be stored digitally outside the cell grid, but loading a large state capacitor to its initial state before the start of processing is a time- and current-consuming task. In this thesis, the memory retention time is not a

problem since a mixed-mode design with static digital memories is used (see Chapters 4 and 5 for details).

3.3 Choice of Processor Hardware

The realization of a polynomial CNN for analysis of EEG is not an easy task. It should be highlighted here that even if the algorithm is described using CNN theory, the type of hardware that is used for the realization is left open. In this section, selected realization alternatives of processors for determining D_{CNN} are discussed. The processor architectures to be considered are overviewed in Chapter 2. The task is to find a good compromise between speed, die area, power consumption and accuracy.

3.3.1 Analog Alternatives

It is possible to include polynomial feedback terms to an analog CNN. In Reference [13], circuits for producing linear and second-order polynomial terms were included in the CNN cell. However, if an analog polynomial CNN with 72×72 cells is built, the power consumption and die area grow large. The power consumption of the 128×128 analog CNN of [14] is about 4W. Also, the die area of the chip is $145mm^2$. A cell in the circuit of [14] has 12 analog multipliers, whereas a CNN with three polynomial orders of feedback terms contains 27 multipliers. Therefore, the power and die area figures of the CNN of [14] and a 72×72 polynomial CNN are comparable. If the 72×72 data was to be partitioned into blocks and processed using a cell array with a downscaled number of cells, the power consumption and die area would be reduced. However, the determination of D_{CNN} requires global interaction of the whole input data.

Since the collection of 5124 samples (a segment) of EEG with f_s of 173 Hz takes 30 seconds, and D_{CNN} is determined from half-overlapping segments, D_{CNN} is determined every 15 seconds. Therefore, even if the power consumption during processing were 4W, the average power consumption would be rather low. However, a 72×72 analog network is unnecessarily fast for the application. Even if a high computing speed is beneficial in learning the weights, somewhat slower approaches would be adequate. Delays and transient noise may affect the operation of a continuous-time CNN. Furthermore, a major problem is the limited retention time of analog memories. In practice, samples of EEG would probably have to be written into a digital memory and, prior to processing, be converted to analog and transferred to the analog network.

The analog microprocessor [15] performs discrete time computations so the transient noise and delays would not be a problem. However, it shares the problems of the retention time, power consumption and die area with the analog CNN. Since the temperature of an implanted device stays at body temperature, the required temperature range is small. This facilitates the design of analog hardware.

3.3.2 Digital Solutions

Processing D_{CNN} digitally is potentially a good alternative. One could use mainstream alternatives such as a DSP, an FPGA or an ASIC. Alternatively, a digital bit-serial processor or a digital CNN could be used. Robust digital storage and discrete time integration would yield good results and a sufficient computing speed could most likely be obtained. However, the D_{CNN} algorithm requires many multiplications. Therefore, realization of the digital multiplication should be carefully planned in order to keep the power consumption down. The characteristics of a digital CNN realization [16] are summarized in Table 5.4.

3.3.3 Mixed-Mode Realizations

Mixed-mode realizations benefit from robust storage and compact analog multiplication. Different mixed-mode architectures were shown in Section 2.5, but none of them is directly suitable for realization of a chip for determining D_{CNN}. The network of Reference [17] is interesting, since it realizes nonlinear weight functions. The nonlinear weight functions were implemented using global nonlinearly modulated current sources and PWM and PPM modulation techniques. The cell state was expressed by oscillations. It is very likely that due to transient and switching noise the design of a 72×72 oscillator network would be very challenging. In any case, combining analog and digital computing units in some way may prove a viable approach. The limited temperature range of an implanted device is also beneficial in mixed-mode realizations.

References

[1] C. E. Elger, et al., 'Characterizing the Spatio-Temporal Dynamics of the Epileptogenic Process with Nonlinear EEG Analyses', *Proceedings of the Seventh International Workshop on Cellular Neural Networks and their Applications*, Frankfurt, Germany, pp. 228-242, 2002.

[2] K. Lehnertz, C. E. Elger, 'Can epileptic seizures be predicted? Evidence from nonlinear time series analyses of brain electrical activity', *Physical Review Letters*, 1998, Vol. 80, pp. 5019-5022, 1998.

[3] K. Lehnertz, C. E. Elger, 'Spatio-Temporal Dynamics of the Primary Epileptogenic Area in Temporal Lobe Epilepsy Characterized by Neural Complexity Loss', *Electroencephalography and Clinical Neurophysiology*, Vol. 95, pp. 108-117, 1995.

[4] G. Widman, et al., 'A Fast General Purpose Algorighm for the Computation of Auto- and Crosscorrelation Integrals from Single Channel Data', *Physica D*, Vol. 121, pp. 65-74, 1998.

[5] R. Tetzlaff, R. Kunz, C. Ames, D. Wolf, 'Analysis of Brain Electrical Activity in Epilepsy with Cellular Neural Networks (CNN)', *Proceedings of the European Conference on Circuit Theory and Design (ECCTD'99)*, Stresa, Italy, pp. 1007-1010, 1999.

[6] R. Kunz, R. Tetzlaff, D. Wolf, 'Brain Electrical Activity in Epilepsy: Characterization of the Spatio-Temporal Dynamics with Cellular Neural Networks based on a Correlation Dimension Analysis', *Proceedings of the IEEE International Symposium on Circuits and Systems*, Geneva, Switzerland, Vol. 2, pp. 389-392, 2000.

[7] R. Kunz, R. Tetzlaff, 'Feature Extraction from Brain Electrical Activity in Epilepsy using Cellular Neural Networks', *Proceedings of the European Conference on Circuit Theory and Design*, Espoo, Finland, Vol. 2, pp. 21-24, 2001.

[8] R. Kunz, R. Tetzlaff, 'Evolutionary Learning Strategies for Cellular Neural Networks', *Proceedings of the Sixth International Workshop on Cellular Neural Networks and their Applications*, Catania, Italy, pp. 241-246, 2000.

[9] F. Corinto, M. Gilli, P. P. Civalleri, 'On Stability of Full Range and Polynomial Type CNNs', *Proceedings of the Seventh International Workshop on Cellular Neural Networks and their Applications*, Frankfurt, Germany, pp. 33-40, 2002.

[10] X. Liao, J. Wang, 'Algebraic Criteria for Global Exponential Stability of Cellular Neural Networks with Multiple Time Delays', *IEEE Transactions on Circuits and Systems-I*, Vol. 50, pp. 268-275, 2003.

[11] J. Zhang, 'Globally Exponential Stability of Neural Networks with Variable Delays', *IEEE Transactions on Circuits and Systems-I*, Vol. 50, pp. 288-291, 2003.

[12] P. Kinget, M. Steyaert, *Analog VLSI Integration of Massive Parallel Processing Systems*. Dordrecht: Kluwer, 1997.

[13] M. Laiho, A. Paasio, K. Halonen, 'Structure of a CNN with Linear and Second Order Polynomial Feedback Terms', *Proceedings of the Sixth International Workshop on Cellular Neural Networks and their Applications*, Catania, Italy, pp. 401-405, 2000.

[14] G. Linan, et al., 'ACE16K: an Advanced Focal-Plane Analog Programmable Array Processor', *Proc. of the* 27^{th} *European Solid-State Circuits Conference*, Villach, Austria, pp. 216-219, 2001.

[15] P. Dudek, P. Hicks, "A General-Purpose CMOS Vision Chip with a Processor-Per-Pixel SIMD Array", *Proc. of the* 27^{th} *European Solid-State Circuits Conference*, Willach, Austria, pp. 228-231, 1999.

[16] P. Keresztes et al. 'An Emulated Digital CNN Implementation', *Journal of VLSI Signal Processing Systems*, Vol. 23, No. 2/3, 1999, pp. 291-303.

[17] H. Ando, T. Morie, M. Nagata, A. Iwata, "An Image Region Extraction LSI Based on a Merged/mixed-signal Nonlinear Oscillator Network Circuit", *Proc. of the* 28^{th} *European Solid-State Circuits Conference*, Florence, Italy, pp. 703-706, Sept. 2002.

This page is intentionally left blank.

Chapter 4

Mixed-Mode CNN

This chapter describes the structure and operation of a mixed-mode CNN. A cell in a mixed-mode CNN is composed of a digital integrator (including digital memory), analog circuits for multiplying and generating the polynomial terms and A/D and D/A converters to interface between them. An algorithm for predicting an epileptic seizure using a CNN was described in Chapter 3. Different realization constraints were also identified. A mixed-mode CNN is a prominent candidate for analysis of brain electrical activity in epilepsy. Since it performs discrete-time integration, it successfully deals with the problem of delays and transient noise; before sampling, time is given for the sum of currents at the integration node to settle down. Therefore, sampling is performed effectively under DC-conditions. Also, the initial states of the cells can be robustly stored in the digital domain.

In a mixed-mode CNN, the sign of cell state is known at all times. Consequently, one-quadrant analog operation is required from the arithmetic circuits (multiplication, polynomial terms) and extension to four quadrants can be performed by steering the output current of a one-quadrant circuit into a positive or negative sum node. This facilitates the design of the polynomial terms in particular. This is important in the epilepsy application but also in other applications that utilize polynomial CNNs. For example, the use of a polynomial CNN has been proposed for displacement vector estimation [1] and for 3D modeling of soil [2].

In many cases, it would be beneficial to trade the computing speed of a CNN for a smaller die area and lower power consumption. Such a tradeoff is possible if the number of cells in a CNN can be downscaled. In a mixed-mode CNN, the input data can be partitioned into fractions (blocks) that are processed separately. These blocks of data can then be processed with a CNN, the size of which is downscaled to match

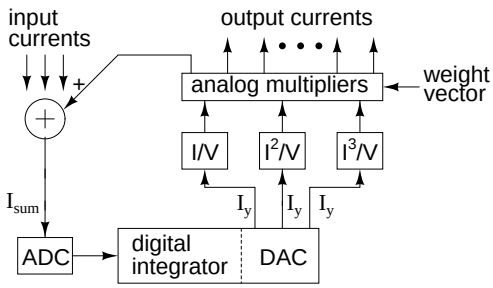

Figure 4.1 Mixed-mode CNN cell with linear, second and third order polynomial feedback terms.

that of a fraction of input data. Section 4.2 shows that, if the blocks of input data are written into the network in parallel, data in different blocks can interact and only local I/O operations are necessary during processing.

4.1 Characteristics of a Mixed-Mode CNN

4.1.1 Operating Principle and Definitions

A mixed-mode CNN is an effort to combine robust digital storage with compact analog multiplication in the same processing element. Figure 4.1 shows a mixed-mode CNN cell with linear, second- and third-order polynomial feedback terms. In a mixed-mode CNN, analog multipliers are used to realize the couplings. The sum of currents I_{sum} that enters the cell at the sum node describes the rate of change of cell state. I_{sum} is converted to digital using an ADC. The result is integrated with a digital integrator. Since the full signal range (FSR) model [3] is used, the cell output equals the digitally limited cell state. The analog output of a cell is represented by currents available at the DAC outputs. One output current is allocated for each polynomial term. Block I/V is a current-to-voltage converter that controls the multipliers that are used for the linear couplings. Blocks I^2/V and I^3/V produce square and cubic terms of cell output current and convert it to voltage. These voltages are used to control the multipliers associated with the second- and third-order polynomial feedback terms.

A CNN cell is a device that performs continuous-time integration of the sum of cell input currents $I_{sum,i,j}(t)$. When the FSR model [3] is used, $I_{sum,i,j}(t)$ is integrated in the state capacitor. This current defines the rate of change of cell state together with the state capacitor C_x. The rate of change of the state of a cell in i^{th} row and j^{th} column ($i \in [1,M]$, $j \in [1,N]$) in a continuous-time FSR CNN with linear, quadratic and cubic couplings is

$$\frac{dV_{i,j}(t)}{dt} = \frac{T_{unit}}{C_x}\sum_{k=1}^{3}\sum_{l=1}^{3} V_{m,p}(t)\left[A_{k,l}^{(1)} + A_{k,l}^{(2)}\left(\frac{V_{m,p}(t)}{V_{unit}}\right)^2 + A_{k,l}^{(3)}\left(\frac{V_{m,p}(t)}{V_{unit}}\right)^3\right] = \frac{I_{sum,i,j}(t)}{C_x}, \tag{4.1}$$

where $m = i + k - 2$, $p = j + l - 2$ and $V_{i,j}$ is the state voltage. The state voltage is bounded by unit voltage V_{unit} according to $\left|V_{i,j}\right| \leq V_{unit}$. Also shown in Equation 4.1 is unit transconductance T_{unit}. The unit transconductance is needed to transform the contributions of the feedback templates $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ into currents. In a conventional CNN, the unit current is defined using a state resistor. Since an FSR CNN does not have a state resistor, the unit current can be defined using the unit transconductance so that $I_{unit} = T_{unit} \cdot V_{unit}$. When Equation 4.1 is time-discretized, the state $V_{i,j}(n+1)$ is iteratively determined by appending $V_{i,j}(n)$ with a fraction of $I_{sum,i,j}(n)/C_x$. In a mixed-mode CNN, the sum current $I_{sum,i,j}(n)$ is digitized and the resulting digital rate of change of state $ri_{i,j}(n)$ is digitally integrated. The digitizing of $I_{sum,i,j}(n)$ with a limited number (b_A) of bits should be done so that the loss of significant information is minimized.

The absolute value of the maximum of the sum current, namely I_{max}, is upper bounded by the sum of all weights multiplied by T_{unit}. The digitized rate of change of state is represented with b_A bits so that $ri_{i,j}(n) \in [0, 2^{b_A} - 1]$. Here it is assumed that the result of the digitizing is represented in sign and magnitude form. When $I_{sum,i,j}(n)$ is positive, the maximum amplitude is assigned to $ri_{i,j}(n)$ according to

$$ri_{i,j}(n) = 2^{b_A - 1} - 1 \text{ if } I_{sum,i,j}(n) \geq k_A \cdot I_{unit}, \tag{4.2}$$

where k_A is a constant scaling factor that together with I_{unit} defines the maximum rate of change of state that is represented digitally. Similarly, when $I_{sum,i,j}(n)$ is negative, the maximum amplitude of $ri_{i,j}(n)$ saturates to

$$ri_{i,j}(n) = 2^{b_A} - 1 \text{ if } I_{sum,i,j}(n) \leq -k_A \cdot I_{unit}. \tag{4.3}$$

Therefore, if $k_A \cdot I_{unit} < I_{max}$ the maximum amplitude of the digital rate of change of state is bounded by the digitizing process. Also, since

$$ri_{i,j}(n) = 0 \quad \text{if} \quad \left|I_{sum,i,j}(n)\right| < \frac{k_A \cdot I_{unit}}{2^{b_A+1}}, \tag{4.4}$$

the choice of k_A is a tradeoff between representing small currents accurately and upper bounding the digital rate of change of state. The resolution of the digital state of the integrator (and the ADC) is b_D bits. Here it is assumed that $b_D \geq b_A$. Before being

fed to the integrator, the digital rate of change of state $ri_{i,j}(n)$ is shifted towards *msb* by $b_D - b_A$ bit positions, i.e.

$$r_{i,j}(n) = ri_{i,j}(n) \cdot 2^{b_D - b_A}, \tag{4.5}$$

where $r_{i,j}(n)$ is the shifted digital rate of change of state. This way, the most significant bits of $r_{i,j}(n)$ and the integrator are of equal value. The integration step is defined as

$$h = \frac{1}{k_A \cdot 2^{k_D}}, \tag{4.6}$$

where scaling parameter k_D is a positive integer that defines how many bit positions $r_{i,j}(n)$ is shifted towards *lsb* before it is fed to the integrator. The scaling parameter k_A is a real number that can be tuned with a reference current/voltage of the ADC and so it also affects the integration step. Therefore, the integration step does not need to be an integer power of two. When a mixed-mode CNN is designed, the integration method and scaling parameters, along with the resolutions b_A and b_D, must be selected.

4.1.2 Choice of Integration Method and Converter Resolutions

When two or more sets of feedback templates are used in a large autonomous polynomial CNN, it is important that the dynamic evolution of the states of the cells is correct. Consequently, it is important that the state integration in the mixed-mode CNN is carefully planned. The limited resolutions of the data converters affect the integration. Even when known digital integration methods are used, their performance with nonideal converters has to be investigated. Also, the cell-level realization of an integration method is a factor that affects the choice of the method. In Reference [4], a CNN was simulated using Euler-Cauchy, Heun's[1] and fourth-order Runge-Kutta integration methods and the results were compared. Here the integration methods are compared taking into account the nonideality of the converters. Comparison is made in terms of convergence time and aspects of cell-level realization. The comparison process is such that, on the basis of simulations with different integration algorithms and conversion accuracies, the best combination can be selected.

4.1.2.1 Considered Integration Algorithms

The three integration algorithms considered in this section are briefly reviewed. The Euler-Cauchy method for solving nonlinear differential equations can be described by

[1] Heun's method of integration is also known as Improved Euler integration method.

$$x_{i,j}(n+1) = x_{i,j}(n) + h \cdot r_{i,j}(n), \tag{4.7}$$

where x_n is the state of a cell at time index n; $n = [0, 1, ...]$, h is the integration step and $r_{i,j}(n)$ is the digital derivative of the state.

Heun's (predictor corrector) integration method is such that first predicted state

$$x^*_{i,j}(n+1) = x_{i,j}(n) + h \cdot r_{i,j}(n) \tag{4.8}$$

is determined. The digital derivative of state at the predicted state $r^*_{i,j}(n+1)$ is then computed and the corrected (final) state obtained using

$$x_{i,j}(n+1) = x_{i,j}(n) + \frac{1}{2}h[r_{i,j}(n) + r^*_{i,j}(n+1)]. \tag{4.9}$$

In the Runge-Kutta method, four auxiliary components $k1 - k4$ are computed. For example, the auxiliary rate of change of state $r^{(k1)}_{i,j}(n+1)$ is computed at auxiliary state $x^{(k1)}_{i,j}(n+1)$ in order to determine $k2$. At time $n+1$, the state $x_{i,j}(n+1)$ can be obtained by combining the auxiliary terms and dividing by six, as shown in Equation 4.10.

$$\begin{array}{lll}
x^{(k1)}_{i,j}(n+1) & = x_{i,j}(n) + h \cdot r_{i,j}(n) & = x_{i,j}(n) + k1 \\
x^{(k2)}_{i,j}(n+1) & = x_{i,j}(n) + \frac{h}{2} \cdot r^{(k1)}_{i,j}(n+1) & = x_{i,j}(n) + k2/2 \\
x^{(k3)}_{i,j}(n+1) & = x_{i,j}(n) + \frac{h}{2} \cdot r^{(k2)}_{i,j}(n+1) & = x_{i,j}(n) + k3/2 \\
k4 & = h \cdot r^{(k3)}_{i,j}(n+1) & \\
x_{i,j}(n+1) & = x_n + \frac{1}{6}(k1 + 2 \cdot k2, +2 \cdot k3 + k4) &
\end{array} \tag{4.10}$$

4.1.2.2 Heuristic Comparison of the Integration Algorithms

In order to choose the integration step h, accuracies of the ADC, DAC and the integration method, a heuristic method for analyzing the different combinations was used. The method was such that first the network was initialized to such a state that small errors in the integration led to convergence into wrong final states. The evolution of the states of the cells in the ideal case (very small h, ideal converters) was recorded. Then, for each integration method, different combinations of integration step and converter accuracies were tried and the dynamic evolution of the states was recorded. The heuristic method does not provide accurate quantitative results, but is useful in comparing the different alternatives for realizing an integrator for a mixed-mode CNN cell. In Reference [4], similar types of experiments were performed without the nonideali-

ties.

In the first simulation, the effect of the nonideal ADC was investigated. The resolution of the ADC was seven bits and the DAC was ideal. Since the resolution of the ADC is limited, a decision has to be made whether to detect small rates of change of state accurately and omit fast changes (small k_A) or show small changes with rough resolution and also detect fast changes of state (large k_A). In the first simulation, the maximum rate of change of state that can be detected was two times the unit current ($k_A = 2$). Also, in this simulation, the scaling parameter k_D was allowed to be a real number. Figure 4.2 shows Matlab simulations of the evolution of a selected cell's state as a function of the integration step. The simulation was performed with four cells having linear and second-order polynomial feedback and zero boundary conditions. The solid curves in the figures represent the correct behavior of the cell. The dashed line represents the case when the integration step has been made too large and the state evolves to a wrong energy minimum.

The top curves of Figure 4.2 were simulated with Euler-Cauchy method. Two hundred sixty five iterations were needed to obtain a steady state. The largest integration step h that gave the correct behavior was 0.045. If the integration step was made larger, the output followed the dashed curve. The middle curves were obtained with Heun's method. Being a two-step algorithm, the convergence time has to be multiplied by two in order to be able to compare it with the Euler-Cauchy method. Therefore, a stable output was obtained after 74 steps. Correct behavior was obtained with an h of 0.265. Finally, the curves on the bottom were simulated with the Runge-Kutta algorithm. Because it is a four-step algorithm, it results in a convergence time of 48 steps. The largest h that gave a correct behavior was 0.8.

In the second simulation the limited resolution of the DAC was also taken into account. The resolution of the ADC (b_A) was kept at seven bits. The simulation arrangement was similar to the previous simulation, but now both the integration step and DAC resolution b_D were varied. In this simulation, only integers were allowed in the scaling parameter k_D. The maximum rate of change of state was limited to four times unity current ($k_A = 4$). The Euler-Cauchy method converged into the correct final state when the DAC resolution was ten bits with integration step $1/32$. It took 195 iteration steps to converge. Similarly, Heun's integration method converged in 54 steps into the correct final state when the DAC resolution was eight bits and the integration step was $1/4$. The simulation of Runge-Kutta method demonstrated robust performance. Integration step h was fixed to $1/2$ and DAC resolution of six bits gave a correct final state in 76 steps. On the basis of these simulations it seems that Heun's method, and especially Runge-Kutta method, can tolerate the nonideality of

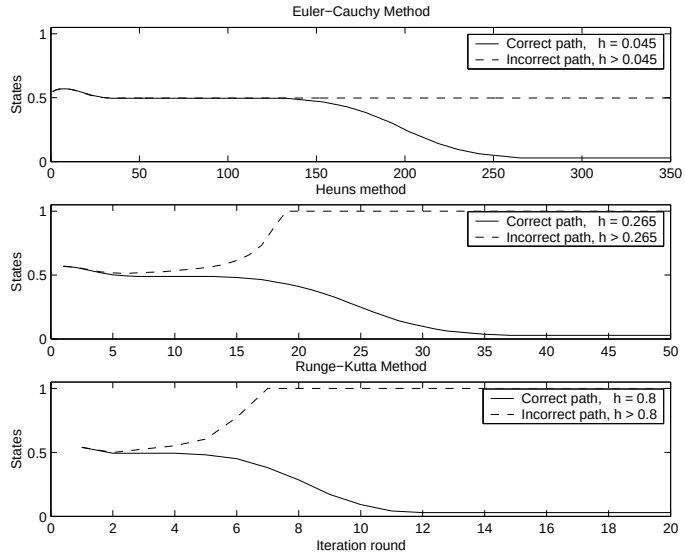

Figure 4.2 Simulation for finding the integration step and convergence time.

the converters much better than the Euler-Cauchy method.

4.1.2.3 Implementation Aspects

When the resolution of an ADC is improved, the conversion speed decreases and the die area grows rapidly. The 7-bit ADC used in the simulation mentioned in Section 4.1.2.2 is potentially a good compromise between die area and resolution. In the previous simulations, the maximum rate of change of state was limited to two or four times the unit current. Some of this nonideality may be eliminated if template values are chosen by learning methods as in the case of determining D_{CNN}. Larger ranges of the rates of change of state could be represented if a nonlinear ADC [11] were used. In a nonlinear ADC, large values of the state derivative are represented with less accuracy than small ones. However, the use of a nonlinear ADC would complicate the integrator design.

The Euler-Cauchy method showed desired performance when a 7-bit ADC with $k_A = 4$ and $b_D = 10$ and the integration step h was $1/32$. The downside is that realization of a 10-bit DAC to each cell is not practical. The Runge-Kutta method converges fast and tolerates converter nonidealities well, but its downside is the implementation. Completing the algorithm requires multiplication by 2, $1/2$ and $1/6$. Multiplication and division by two can be performed with little extra hardware, but division by six is not easy in the digital domain. Also, since memory needs to be allocated for four

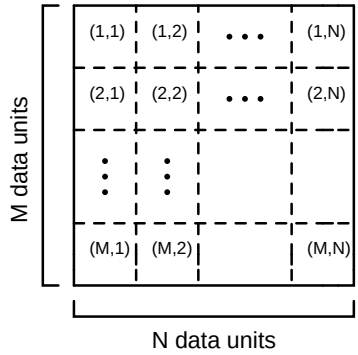

Figure 4.3 Input data.

auxiliary terms, a cell-level implementation of an integrator based on Runge-Kutta algorithm seems impractical.

According to the simulations and aspects of implementation, Heun's method seems like the method of choice. Simulations demonstrated good performance when a 7-bit ADC and an 8-bit DAC were used, the maximum rate of change of state was limited to four times the unit current and the integration step was $1/4$. Design of a 7-bit ADC and 8-bit DAC to each cell is challenging, but, nevertheless, within the limits of what can be designed in a practical realization.

4.2 Processing Data in Blocks

This section describes how data can interact globally in a mixed-mode CNN even if the data is processed blockwise. The data blocks are written to the cells in layers so that each layer of data in the cell memory corresponds to a block of data. Therefore, the number of cells is downscaled but the amount of memory in a cell is increased in the same proportion. This way, all I/O operations during processing are local (within the same die). This section also discusses different ways of partitioning the data and connecting data units in different blocks. Also, the effect of downscaling the number of processors to processing speed, die area and power consumption is described.

4.2.1 Division of Data into Blocks

Figure 4.3 shows the initial $M \times N$ data. Data unit (i, j), $i \in [1, M]$; $j \in [1, N]$, represents the initial data in i^{th} row and j^{th} column. If the data is partitioned, the size of a data block is $M/R_y \times N/R_x$ where R_y and R_x are positive integers larger than, or equal

to, one. Additionally, R_y is an integer multiple of M so that M/R_y is an integer and R_x is an integer multiple of N so that N/R_x is an integer. Therefore, the data is partitioned into $R_b = R_y \cdot R_x$ blocks. Since all data is stored in the partitioned network, the memory in each cell has R_b layers. The orientation of the data blocks can be changed during the partitioning. This affects the way data in different blocks can be coupled. If the orientations of the data blocks are not altered during the partitioning, the initial data is chopped into blocks so that data unit (i, j), is mapped into data unit (ii_c, jj_c, kk), $ii_c \in [1, M/R_y]$; $jj_c \in [1, N/R_x]$; $kk \in [1, R_b]$ so that

$$\begin{aligned} ii_c &= (i-1) \bmod \tfrac{M}{R_y} + 1 \\ jj_c &= (j-1) \bmod \tfrac{N}{R_x} + 1 \\ kk &= R_x(i-ii_c)R_y/M + (j-jj_c)R_x/N + 1 \end{aligned} \quad . \tag{4.11}$$

Another way of partitioning considered here is folding. In the folding approach, the original $M \times N$ data is folded into R_b blocks so that data unit (i, j) is mapped into data unit (ii_f, jj_f, kk), $ii_f \in [1, M/R_y]$; $jj_f \in [1, N/R_x]$ according to

$$\begin{aligned} ii_f &= \begin{cases} ii_c & \text{if } \frac{(i-ii_c)R_y}{M} \bmod 2 = 0 \\ \frac{M}{R_y} - ii_c + 1 & \text{if } \frac{(i-ii_c)R_y}{M} \bmod 2 = 1 \end{cases} \\ jj_f &= \begin{cases} jj_f & \text{if } \frac{(j-jj_c)R_x}{N} \bmod 2 = 0 \\ \frac{N}{R_x} - jj_c + 1 & \text{if } \frac{(j-jj_c)R_x}{N} \bmod 2 = 1 \end{cases} \end{aligned} \quad . \tag{4.12}$$

Figure 4.4b) shows the result of partitioning the data of Figure 4.4a) using Equation 4.11 (chopping). Figure 4.4c) results when the data is partitioned using Equation 4.12 (folding). In both cases $R_x = R_y = 4$. A combination of these partitioning methods is also possible; for example, the data can be chopped vertically and folded horizontally. The partitioned data is written to the initial states of the mixed-mode CNN. For example, if the data is folded, $(ii, jj, kk) = (ii_f, jj_f, kk)$ and the initial state of layer kk of a cell in ii^{th} row and jj^{th} column is $x_{ii,jj,kk} = (ii, jj, kk)$.

4.2.2 Realization of Intra-Block Couplings

Whether or not the orientation of the data blocks is changed during the partitioning affects the realization of couplings between blocks. If the data is chopped (the orientations of data blocks are left unaltered), data has to be transferred between the opposite side edge cells. If the data is folded, no intra-cell data transfer is needed to convey the boundary information. A downside to the folding method is that the orientation of the template also has to be flipped during processing. When data is chopped or folded in two dimensions, the realization of intra-block couplings becomes more difficult.

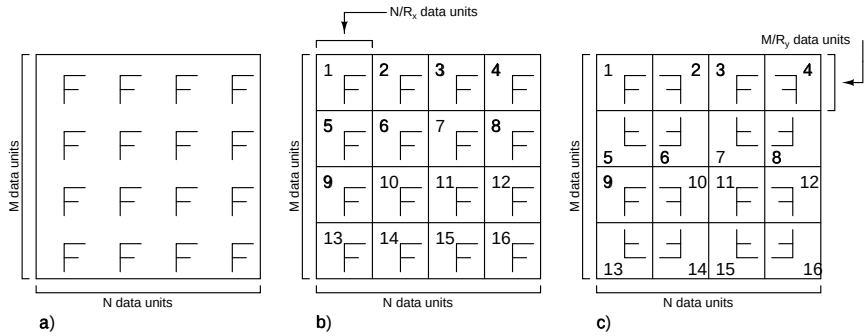

Figure 4.4 An illustration of two examples of partitioning an $M \times N$ network into 16 blocks ($R_x = R_y = 4$). a) Original data. b) Data partitioned by chopping. c) Data partitioned by folding.

4.2.2.1 Processing Data with a Partitioned Network

The problem in processing the input data in blocks lies in achieving global interaction for the data: in general, in order for the whole input data to be able to interact simultaneously, all data units need to be locally connected to their neighboring cells. This problem has been identified previously in, for example, digital CNN realizations [5]. In the analog realizations of References [6], [7], [8], a template-dependent solution for connecting pixels in different blocks was used. Overlap was added to the image fractions such that, after processing a block, a certain overlap region was discarded. This works for templates like resistive filtering template [9] where the spatial interaction of pixels decays as a function of their distance and correct temporal behavior is not important. In general, when feedback templates like hole filler [10], which need global interaction, are used, the overlap method is not sufficient. When the data is processed blockwise, correct spatial and temporal interaction of data in different blocks is needed. The integrator takes care that data is updated so that the temporal interaction is correct. In the following, realization of the correct spatial interaction between data blocks is described.

4.2.2.2 Couplings Between Chopped Data Blocks

Figure 4.5 demonstrates the partitioning of an 8×4 data into four 2×4 blocks using Equation 4.11 with $R_y = 4$ and $R_x = 1$. The dashed arrows in the figure illustrate the virtual connections between data blocks. For example, when processing block 2, state $x_{2,3,1}$ is available at the top boundary of state $x_{1,3,2}$ and state $x_{1,3,3}$ is brought to the bottom boundary of state $x_{2,3,2}$. Therefore, intra-cell transfer of data in different blocks has to be realized.

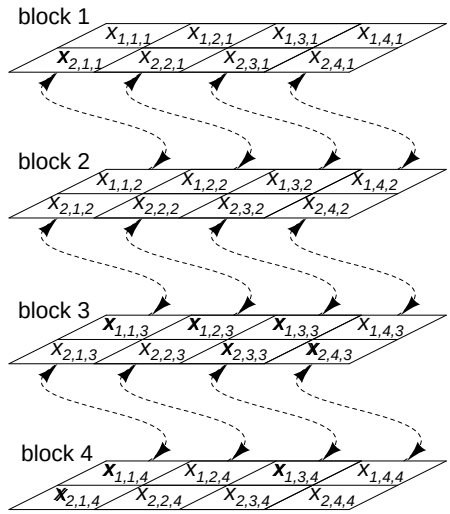

Figure 4.5 An 8×4 data partitioned into four 2×4 blocks using Equation 4.11. The dashed arrows describe connections between data blocks.

Figure 4.6 shows a combination of a top row edge cell $C(1, jj)$ and a bottom row edge cell $C(M/R_y, jj)$ when data is partitioned using Equation 4.11 with $R_x = 1$. Therefore, the partitioned network has $M/R_y \times N$ cells. Simplified cells are shown in the figure; the ADC and DAC, for example, are not shown. The elements of the template in use are $A_{k,l}$, $k = [1,3]$; $l = [1,3]$. The states of cells are described by $x_{ii,jj,kk}$. When processing block kk, the top row edge cells provide the bottom row edge cells with $x_{ii,jj,f_N(kk)}$. Similarly, the bottom row edge cells provide the top row edge cells with $x_{ii,jj,f_S(kk)}$. These intra-block/cell couplings are realized by wiring. If $M/R_y > 2$, the wires have to go either through or around cells $C(ii, jj)$, $ii = [2, M/R_y - 1]$; $jj = [1, N]$. The top and bottom row mapping functions are

$$f_N(kk) = kk + 1 \tag{4.13}$$

and

$$f_S(kk) = kk - 1. \tag{4.14}$$

The border condition of the bottom row edge cells is therefore stored to x_{1,jj,R_y+1} and the border condition of the top row edge cells is stored to $x_{M/R_y,jj,0}$.

If both R_x and R_y are larger than one, intra-block connections become more difficult to realize. Cells that are in the corners of the active cell grid (8-connected network) have to be provided with the boundary information of three other blocks. Figure 4.7a)

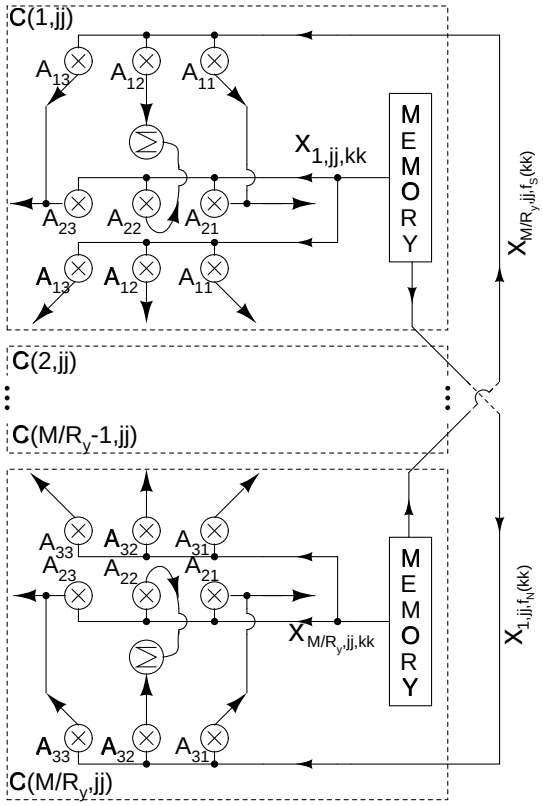

Figure 4.6 A combination of a top row edge cell and a bottom row edge cell for processing vertically chopped data.

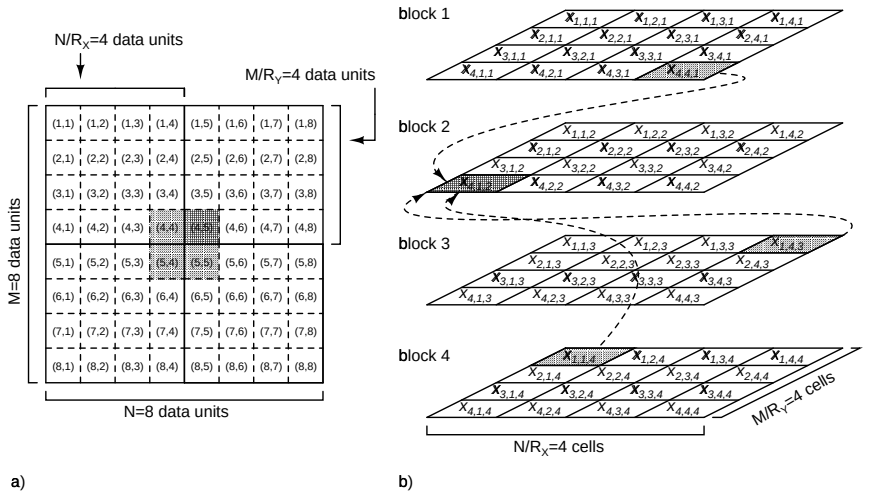

Figure 4.7 Partitioning of an 8×8 data input data into blocks 4.11 using Equation 4.11 with $R_x = R_y = 2$. a) Original 8×8 data. b) partitioned data. The dashed arrows show that during processing of block 2, the corner cells need boundary information from three other blocks.

shows an 8×8 data. The difficulty of connections between the blocks is illustrated in Figure 4.7b). In Figure 4.7b) the input data is partitioned to blocks using Equation 4.11 with $R_x = R_y = 2$. The corner cells need boundary information from three different blocks. For example, during the processing of block 2, the state $x_{4,1,2}$ of the bottom left corner cell needs to be provided with $x_{4,4,1}$, $x_{1,4,3}$ and $x_{1,1,4}$. Processing data that has been partitioned horizontally and vertically was proposed in a digital CNN [5]. Edge cells on different sides of the network were connected with a bus and data in all blocks was updated simultaneously. This could also be done in a mixed-mode CNN, but here division in one direction is assumed sufficient.

4.2.2.3 Couplings Between Folded Data Blocks

Figure 4.8 demonstrates the partitioning of an 8×4 data using Equation 4.12 with $R_y = 4$ and $R_x = 1$. Again, the data is divided into four 2×4 blocks; the dashed arrows in the figure illustrate the virtual connections between data blocks. Because the data was partitioned by folding, no intra-cell connections are needed.

Figure 4.9 shows a combination of a top row edge cell $C(1, j)$ and a bottom row edge cell $C(M/R_y, j)$ when an $M \times N$ data is partitioned into R_y blocks using Equation 4.12 with $R_x = 1$. Again, the cells shown in the figure are simplified, with the ADC and DAC, for example, not being shown. The template is composed of $A_{k,l}$, $k = [1,3]$; $l = [1,3]$ and the states of the cells are described by $x_{ii,jj,kk}$. When processing folded

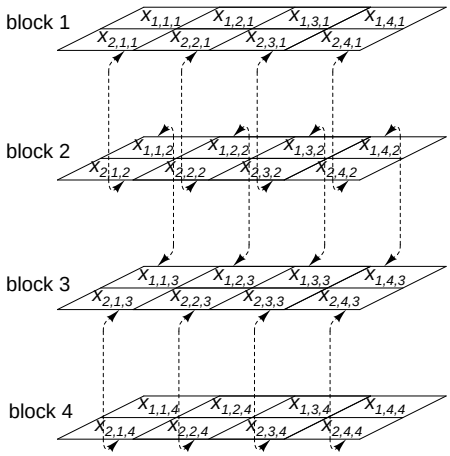

Figure 4.8 An 8×4 data partitioned into four 2×4 blocks using Equation 4.12. The dashed arrows describe connections between blocks.

data, odd and even data blocks have to be distinguished. Figure 4.9a) shows edge cells during the processing of a data block *kk* when *kk* is odd. The top row edge cells have access to $x_{1,jj,kk}$ and $x_{1,jj,f_S(kk)}$. Similarly, the bottom row edge cells have access to $x_{M/R_y,jj,kk}$ and $x_{M/R_y,jj,f_N(kk)}$. Figure 4.9b) shows the edge cells during the processing of a data block *kk* when *kk* is even. In this case, the mapping function of Equation 4.14 is used in the bottom edge cell, while the mapping of Equation 4.13 is used in top row edge cells. Furthermore, the template is flipped vertically. If R_y is even, the boundary condition of the top of the data is stored to $x_{1,jj,0}$ and the bottom boundary condition is stored to x_{1,jj,R_y+1}. If R_y is odd, the bottom boundary condition is stored to $x_{M/R_y,jj,R_y+1}$. Therefore, only inter-cell data transfers are necessary for conveying boundary information.

Again, if both R_x and R_y are larger than one, and the data is partitioned by folding, the corner cells have to be provided with border information of three other blocks. In addition to this, the mapping is much more complicated than in the 1D folding and, since the data in the blocks has four possible orientations, the template also has to be flipped to four different orientations during processing.

4.2.3 Characteristics of a Partitioned Network

In order to decide the number of blocks R_b, the effect of processing in blocks has to be evaluated in terms of different metrics. In the following, performance metrics for processing speed, die area and power consumption are shown. The metrics are

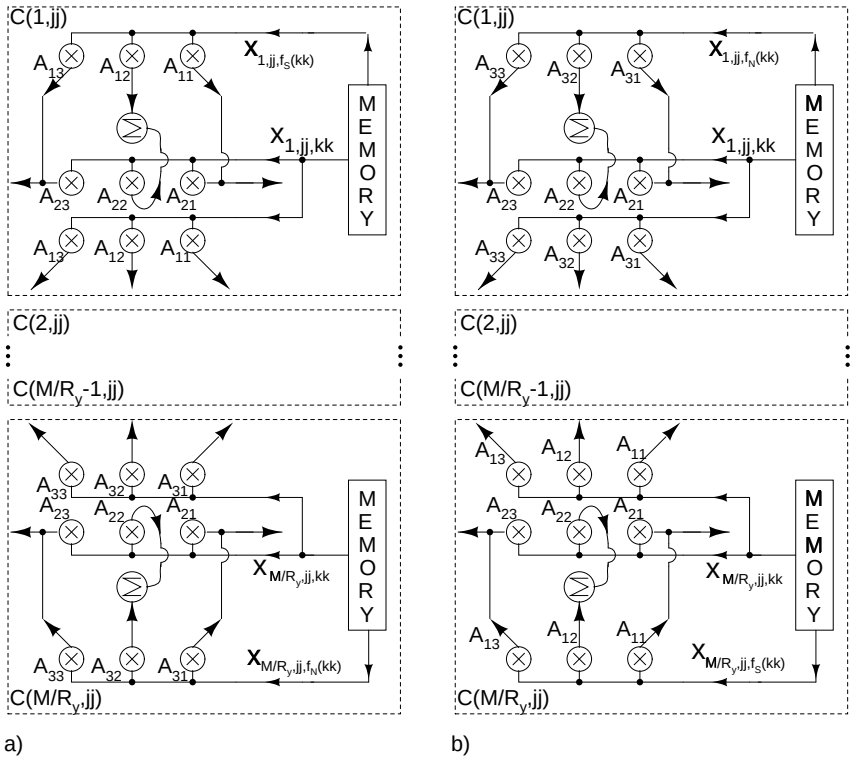

Figure 4.9 A combination of a top row edge cell and a bottom row edge cell for processing vertically folded data. a) processing of odd data blocks. b) processing of even data blocks.

given as functions of R_b. The metrics are not intended to provide accurate quantitative results but to reveal qualitative trends as a function of the number of blocks.

4.2.3.1 Computing Speed

The processing time of an $M \times N$ data with a mixed-mode CNN in which the number of cells is downscaled to $M \times N/R_b$ can be approximated by

$$T_f(R_b) = s \cdot nri[R_b(t_s + t_l + t_{sett}) + k_t \cdot t_t] + t_{wd} + t_{rd} + t_{lt}, \tag{4.15}$$

where s and nri are the number of steps in the integration method of choice and the number of iterations, respectively. If Heun's method, for example, is used for integration, s is two. Also, t_s is the ADC sampling time, t_l is the time consumed by the digital in-cell logic and t_{sett} is the settling time of the analog parts of the cell. These delays occur during processing of every block and therefore they are multiplied by R_b. Additionally, t_t is the time it takes to change the orientation of a template and k_t is the number of changes of template orientation during one iteration cycle. Finally, global write time t_{wd}, global read time t_{rd} and template load time t_{lt} are added to the result. If the orientation of the template is changed globally, t_t and t_{lt} are equal.

4.2.3.2 Die Area and Power Consumption

The reduction in die area resulting from partitioning the network can be elucidated by the ratio of the area of the partitioned network and the area of the full-size network. This ratio is approximated by

$$R_a(R_b) = \frac{(A_o/R_b + A_m)M \cdot N}{(A_o + A_m)M \cdot N} = \frac{A_o/R_b + A_m}{A_o + A_m}, \tag{4.16}$$

where A_m is the area occupied by the state memory of a cell in a full-size network and A_o is the area of other circuit blocks in a cell. Equation 4.16 is derived on the assumption that the total amount of memory on chip does not decrease as a function of partitioning. As R_b increases, the area occupied by other circuit blocks A_o becomes less significant.

The instantaneous power consumption (power consumption during processing) of a cell as a function of R_b can be approximated by

$$P_{c,inst}(R_b) = P_{alg} + P_{conv} + P_{logic} + P_{mem} \cdot R_b, \tag{4.17}$$

where P_{alg} and P_{conv} are the powers consumed in the analog components and the data

converters of a cell, respectively. P_{logic} is the power consumption of the cell digital logic and P_{mem} is the power consumption of the memory of a cell in a full-size network. The amount of state memory in a cell increases as a function of partitioning. Here it is assumed that the power consumption of the memory P_{mem} increases in the same proportion. Therefore, P_{mem} is multiplied by R_b.

The reduction of instantaneous power consumption at network level can be estimated by the ratio of the instantaneous power consumptions of a partitioned network and a full-size network according to

$$R_{P,inst}(R_b) = \frac{P_{c,inst}(R_b)/R_b}{P_{c,inst}(1)}. \tag{4.18}$$

Even though the ratio of the instantaneous power consumptions is reduced as a function of R_b, the average power consumption during processing a certain amount of data is much more stable. Since a partitioned network computes slower than a full-size network, the full-size network can be turned off for time $T_f(R_b) - T_f(1)$ to do the same task. Therefore, the ratio of the average power consumptions can be obtained by scaling the instantaneous power consumptions by the corresponding processing times.

4.2.3.3 Choice of the Number of Blocks

It is evident that partitioning the network trades instantaneous power consumption and die area for computing speed. In the following, the choice of the number of blocks R_b is examined by way of example. The example mixed-mode CNN (built with a $0.25\mu m$ CMOS process) is able to process a 72×72 data with first-, second- and third-order polynomial templates. Figure 4.10 shows $T_f(R_b)/T_f(1)$ as a function of R_b with different numbers of iterations *nri*. When *nri* is small, the effect of R_b on the computing speed is small, since global I/O and template loading dominates the computing time. When *nri* is large, the I/O and template loading are less significant. The curves in Figure 4.10 are sketched with values of Table 4.1. The sampling time, time of logic operations and analog settling time are taken from measurements of Chapter 5. It is assumed that the settling of the weight voltages takes $5\mu s$. Moreover, the global I/O times are determined assuming that one cell state can be read or written in $100ns$ and the size of the data is 72×72.

Figure 4.11 shows $R_a(R_b)$ and $R_{P,inst}(R_b)$ as a function of R_b. The $R_a(R_b)$ curve can be interpreted so that large savings in the network area as a function of R_b are obtained until the curve saturates to $A_m/(A_o + A_m)$. Similarly, $R_{P,inst}(R_b)$ saturates to $P_{mem}/R_{P,inst}(1)$. However, since P_{mem} of a digital memory is small, the power consumption scales down very well. Table 4.2 shows the values that were used in Equa-

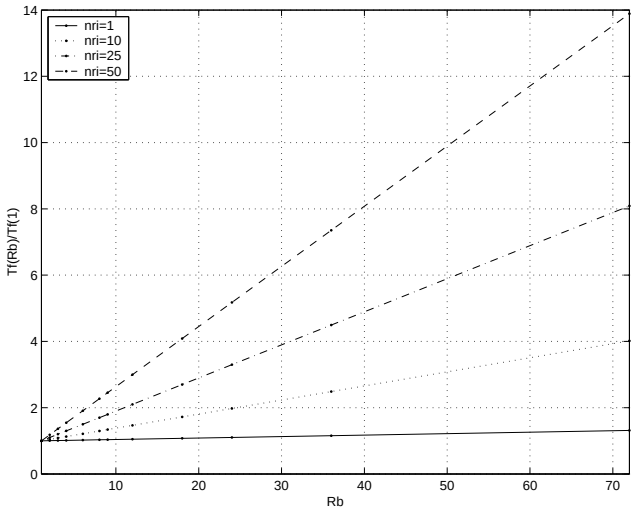

Figure 4.10 $T_f(R_b)/T_f(1)$ as a function or R_b with different numbers of iterations *nri*.

symbol	description	value
s	# of steps in the integration method	2
nri	# of iterations	1,10,25,50
t_s	sampling time	1400ns
t_l	time of logic operations	710ns
t_{sett}	analog settling time	200ns
t_t	change of template orientation	-
k_t	number of orientation changes during an iteration round	0
t_{lt}	template load time	5μs
t_{wd}	global write time	520μs
t_{rd}	global read time	520μs

Table 4.1 Explanation of symbols in Equation 4.15 and values used to produce curves of Figure 4.10.

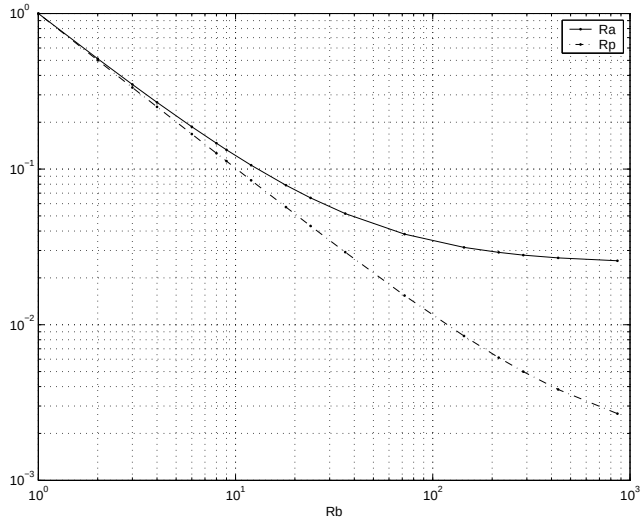

Figure 4.11 $R_a(R_b)$ and $R_{P,inst}(R_b)$ as a function of R_b.

symbol	description	value
A_m	the area occupied by the state memory of a cell in a full-size network	$560\mu m^2$
A_o	area of other circuit blocks of a cell	$22000\mu m^2$
$P_{alg}+$ $P_{conv}+$ P_{logic}	power consumption of analog parts of a cell power consumption of data converters of a cell power consumption of digital logic of a cell	$1.3mW$
P_{mem}	power consumption of digital memory of a cell in a full-size network	$1.9\mu W$

Table 4.2 Explanation of symbols in Equations 4.16 and 4.17 and values that were used to produce curves of Figure 4.11.

tions 4.17 and 4.18 to produce the curves of Figure 4.11. The power consumption figures are based on the measurements reported in Chapter 5, while the area figures are characteristics of the measured chip.

References

[1] D. Feiden, R. Tetzlaff, “Displacement Vector Estimation with Cellular Neural Networks”, *International Joint Conference on Neural Networks*, Honolulu, Vol. III, pp. 2049 -2052, 2002

[2] P. Lopez, et al., “CNN-Based 3D Thermal Modeling of the Soil for Antiper-

sonnel Mine Detection", *Proceedings of the International Workshop on Cellular Neural Networks and their Applications*, Frankfurt, pp. 307-314, 2002.

[3] S. Espejo, R. Carmona, R. Dominguez-Castro, A. Rodriguez-Vazquez , "A VLSI-oriented Continuous-time CNN Model", *International Journal of Circuit Theory and Applications*, Vol. 24, no. 3, pp. 341-356, 1996.

[4] H. Harrer, A. Schuler, E. Amelunxen, 'Comparison of Different Numerical Integration Methods for Simulating Cellular Neural Networks' *Proceedings of the First International Workshop on Cellular Neural Networks and their Applications*, Budapest, Hungary, pp. 151 -159, 1990.

[5] M. D. Doan, et al., 'Realisation of a Digital Cellular Neural Network for Image Processing', *Third International Workshop on Cellular Neural Networks and their Applications*, Rome, pp. 85-90, 1994.

[6] E. Roca, S. Espejo, R. Dominguez-Castro, G. Linan, A. Rodrizuez-Vasquez, 'A Programmable Imager for Very High Speed Cellular Signal Processing', *Journal of VLSI Signal Processing Systems*, Vol. 23, 305-318, 1999.

[7] L. Wang. J. Pineda de Gyvez, E. Sanchez-Sinencio, 'Time Multiplexed Color Image Processing Based on a CNN with Cell-State Outputs', *IEEE Transactions on VLSI Systems*, Vol. 6, pp. 314-322, 1998.

[8] A. Kananen, A. Paasio, K. Halonen, 'Overlapping issues in designing large CNNs', *Proceedings of the Sixth International Workshop on Cellular Neural Networks and their Applications*, Catania, Italy, pp. 321-324, 2000.

[9] B. Shi, L. O. Chua, 'Resistive Grid Image Filtering: Input/Output Analysis via the CNN Framework', *IEEE Transactions on Circuits and Systems I*, Vol. 39, 531-548, 1992.

[10] T. Matsumoto, et al., 'Several Image Processing Examples by CNN', *Proceedings of the International Workshop on Cellular Neural Networks and their Applications*, Budapest, Hungary, pp. 100-111, 1990.

[11] R. Gregorian, G. Temes, *Analog MOS Integrated Circuits for Signal Processing*, Wiley: New York, p. 549-552, 1986.

Chapter 5

Mixed-Mode Cellular Array Processor Realization

In this chapter, a mixed-mode cellular array processor realization is depicted. The realization is targeted to meet the requirements of the epilepsy prediction application described in Chapter 3. The chip implementation is based on the mixed-mode CNN architecture depicted in Chapter 4. The objectives of this chapter are the following:

1. Demonstrate the realizability/performance of the mixed-mode CNN architecture with a proof-of-concept chip targeted towards the epilepsy application.

2. Show measured data that can be used to assess the effect of processing variations to the accuracy and, furthermore, to the applicability of the design.

This chapter is organized so that Section 5.1 provides some general characteristics of the realization, depicts the processor array and supporting hardware and shows connections between PUs[1] and details of the integrator. Section 5.2 describes design of the analog arithmetic circuits and Section 5.3 shows the data converters. Design of peripheral devices and layout are overviewed in Section 5.4.2, while experimental results of the implemented parallel processor are presented in Section 5.5, together with a discussion of some of the important points.

[1] Instead of using “cell”, in this chapter, a processing unit is denoted by “PU”.

5.1 System and Integrator Implementation

The implemented cellular array processor realizes the functionalities of a mixed-mode CNN. Ease of testability was an important guideline in the design of the chip. In order to relax the speed requirements of the evaluation setup, all memories on chip were implemented as SRAMs. The ADC and DAC resolutions b_A and b_D were chosen to be seven and eight bits, respectively. Therefore, the state of the PU $x_{ii,jj,kk}(n)$ is truncated between -128 and 127. Programmable 8-connected, first-neighborhood linear, quadratic and cubic feedback templates are available simultaneously and Heun's integration method is used. The chip can process 72×72 data while keeping all I/O operations local. The 72×72 data is processed with a 2×72 network ($R_b = R_y = 36$). This choice of R_b scales down the die area and reduces power consumption significantly compared to a full-size network. The fact that the physical network has only two rows is advantageous in the realization of the PU (only edge cells are used). This is because only six different weights out of a 3×3 template need to be brought into an edge cell. This can be verified from Figures 4.6 and 4.9. The data is partitioned by chopping mainly for two reasons. Firstly, when processing chopped data, the mapping function is the same for odd and even data blocks. Secondly, since there are no regular cells in the network, conveying the border information between top and bottom row edge cells is easy. Furthermore, the orientation of the templates does not have to be changed when processing chopped data. Changing the orientation of a template is time consuming when analog multipliers are used.

5.1.1 Processor Array and Supporting Hardware

The implemented mixed-mode cellular array processor is depicted in Figure 5.1. Coupling strength (weight) memories, ADC control signal generator, decoders and a processor array composed of 2×72 identical coupled processing units (PUs) are shown. An external control unit is used to control the chip via a control bus and a bi-directional data bus. The control interface to the chip is completely digital. The weight memories are digital (SRAMs). Twenty seven D/A converters are included to produce the analog weight voltages that are fed to the multipliers. The weights are digitally programmable with eight bits. An asynchronous control signal generator is used to produce the control signals for the ADCs in the PUs. The signals are generated asynchronously on chip so that fast conversion speeds can also be obtained with a slow external control unit. The actual conversion speed can be measured from outside the chip via test signals. With aid of the row and column activation signals produced by the decoders,

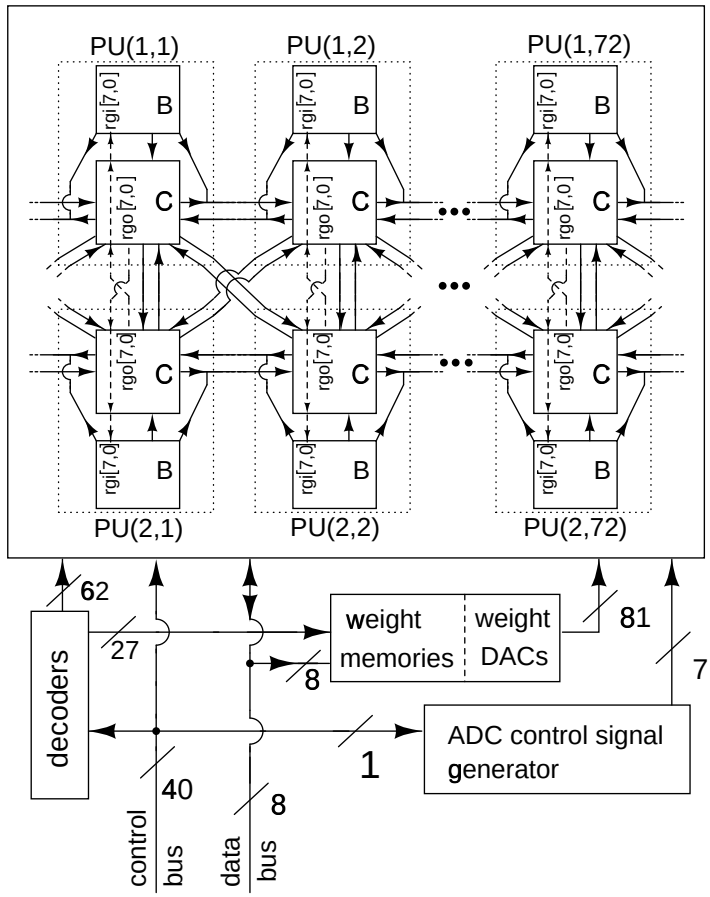

Figure 5.1 Processor array and supporting hardware.

each processing unit can be randomly accessed for I/O operations.

Odd rows of input data are written to top row PUs and even rows of input data to bottom row PUs (partitioning by chopping). The memories in the 2×72 PUs have R_b=36 layers in order to store the data blocks as suggested in Chapter 4. Solid arrows between the PUs in Figure 5.1 denote the currents that are used for coupling. The couplings to the left side of the leftmost PUs and to the right side of the rightmost PUs are connected as shown in Figure 5.18. Each PU is composed of two parts: main unit *C* and border unit *B*. The border information between top and bottom row edge cells is exchanged as suggested in Section 4.2.2.2: the border unit along with register connections $rgi[7,0]$ and $rgo[7,0]$ between the main and border units of top and bottom rows (dashed arrows in Figure 5.1) realize the virtual spatial connections. The register

connections transfer data from different layers of PU memory to the border units. As stated earlier, only six weights out of each 3×3 convolution mask have to be brought to a PU. Furthermore, currents to only five directions are conveyed outside the PU, since the self-feedback current is used within the PU. This facilitates the interconnection wiring. The PUs in the top and bottom rows are identical, but the orientations are different (the top row PU is vertically flipped compared to the bottom row PU). The border condition is cyclic in the horizontal direction so that ($x_{ii,0,kk} = x_{ii,72,kk}$ and $x_{ii,73,kk} = x_{ii,1,kk}$). In the vertical direction, the border condition is zero ($x_{1,jj,37} = 0$ and $x_{2,jj,0} = 0$).

5.1.2 Structure of a Mixed-Mode Processing Unit

The PU works so that input currents provided by analog multipliers of the neighborhood PUs are summed and A/D converted. The digitized sum of currents represents the rate of change of state ($r_{i,j}(n)$ or $r^*_{i,j}(n+1)$) and is used to digitally integrate the state of the PU using the integrator. Digital integrator outputs are D/A converted and used to produce the analog linear, quadratic and cubic activation functions that control the multipliers.

Figure 5.2 shows the contents of a mixed-mode PU drawn with bottom row denotations. The main unit (denoted by *C* in Figure 5.1) is composed of the integrator and blocks *DAP*1, *MC*18 and *ADC*. The border unit (denoted by *B* in Figure 5.1) consists of blocks *DAP*2 and *MB*9. Also shown are register connections $rgi[7,0]$ and $rgo[7,0]$. The *DAP* blocks convert a digital input into analog linear, quadratic or cubic activation functions which control the multipliers. The multiplier blocks *MC*18 and *MB*9 contain 18 and 9 analog multipliers, respectively, which are used to realize couplings between the processing units. The arrows in the top part of the PU denote the input and output currents. For example, Ic_{33} equals $A^{(1)}_{33}x + A^{(2)}_{33}x^2 + A^{(3)}_{33}x^3$ and is taken to a PU in NW direction. The integrator has two outputs: output *rg_C* ($x[7,0]$) controls the D/A converter in *DAP*1 and *rg_B* controls intra PU bus $rgo[7,0]$. Furthermore, the multipliers in *MC*18 are controlled with outputs of *DAP*1 and $x[7]$. Similarly, register input $rgi[7,0]$ controls the border unit (*DAP*2 and *MB*9).

5.1.3 Integrator Realization

5.1.3.1 Structure of the Integrator

Figure 5.3 shows the structure of the integrator that is used in the cellular array processor. It realizes Heun's method of integration. The timing diagrams of Figure 5.3

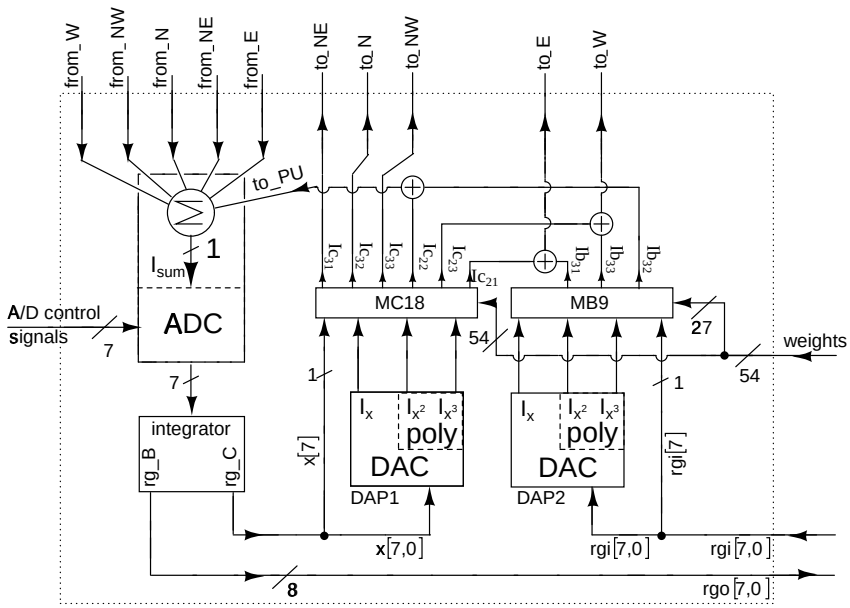

Figure 5.2 A processing unit drawn with bottom row denotations.

are used in Section 5.1.3.3. The integrator contains three SRAM memories (*REG_1*, *REG_2* and *REG_Y*), each with 36×8 bits of storage capacity (36 layers), and one 8-bit register (*REG_CMB*). In addition to the memories, the integrator contains a 9-bit full adder, switches and a digital nonlinear block. Since FSR model is used, the digital nonlinear block limits the state between -128 and 127. The integration step h equals $1/k_A$ since $k_D = 1$ (see Equation 4.6). As indicated in Equation 4.5, the 7-bit data that comes from the ADC is multiplied by two before it is fed to the integrator since $b_D - b_A = 1$. Two's complement arithmetic is used. Consequently, the data that comes from the ADC is in two's complement form. Decoder output signals $act[kk]$, $kk \in [1,36]$ determine which data block is being processed. Signals *wr_1*, *wr_2*, *wr_y* and *wr_cmb* (active HI) are the write signals of registers *REG_1*, *REG_2*, *REG_Y* and *REG_CMB*, respectively. Signals *rg2_ct*, *add_ct* and *out_ct* control the switches that select the input of *REG_2*, the input of the adder and the output of the integrator, respectively. The switches in the figure are drawn to their position when control signals *rg2_ct*, *add_ct* and *out_ct* are HI.

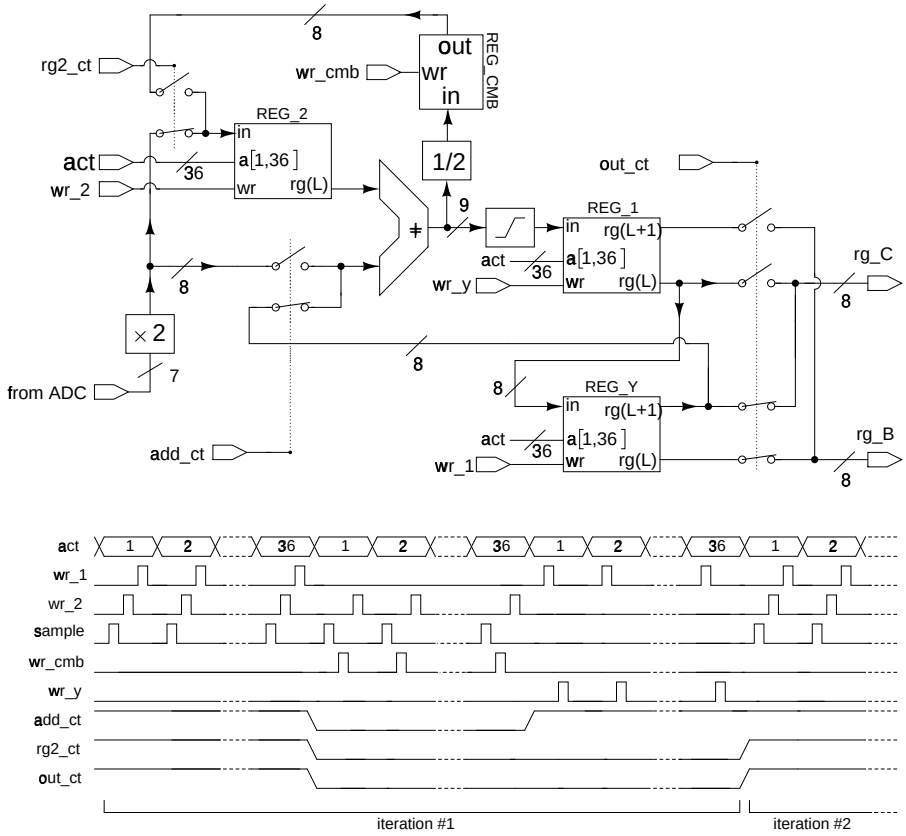

Figure 5.3 Heun's integrator.

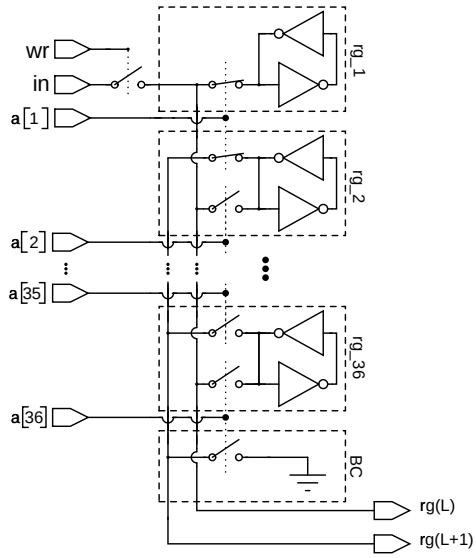

Figure 5.4 SRAM with simultaneous outputs from two layers (top row orientation).

5.1.3.2 Memory and Logic

Figure 5.4 shows the structure of SRAMs REG_1 and REG_Y which can be accessed simultaneously from two layers. SRAM REG_2 is like REG_1 and REG_Y but with only one output. Only one bit per layer is shown in the figure for the sake of simplicity. Input signals $a[L]$; $L \in [1,36]$, select the active layer of memory. In the figure, $a[1]$ is active and therefore the contents of rg_1 and rg_2 are written to outputs $rg(L)$ and $rg(L+1)$, respectively. When $a[36]$ is active, output $rg(L+1)$ is connected to ground, which is the real border condition. In a top row PU $a[1,36] = act[1,36]$, data block kk is written to layer $L = kk$ and output $rg(L+1)$ contains the data of block $kk+1$. The bottom row PU (and the integrator with it) is flipped vertically in contrast to a top row PU, but the activation signals are kept in the same order ($act[1]$ being the highest). Therefore, $a[1,36] = act[36,1]$ and data block kk is written to layer $L = 36 - kk + 1$. Consequently, output $rg(L+1)$ contains data of block $kk-1$. This mapping is as desired in Section 4.2.2.2. A pre-charging logic is used to set the voltages at $rg(L)$ and $rg(L+1)$ to midpoint before reading. The pre-charging logic is not shown in Figure 5.4. The operating voltage of the integrator and the memories is 2.5V. The switches in Figures 5.3 and 5.4 are realized with one transistor pass-gates that are controlled by 3V control signals. The 2.5V outputs rg_B and rg_C are scaled to 3V before they are fed to the DACs. The nonlinear block is a conditional pass-gate. If the magnitude of the input to the nonlinear block is larger than allowed, an output of

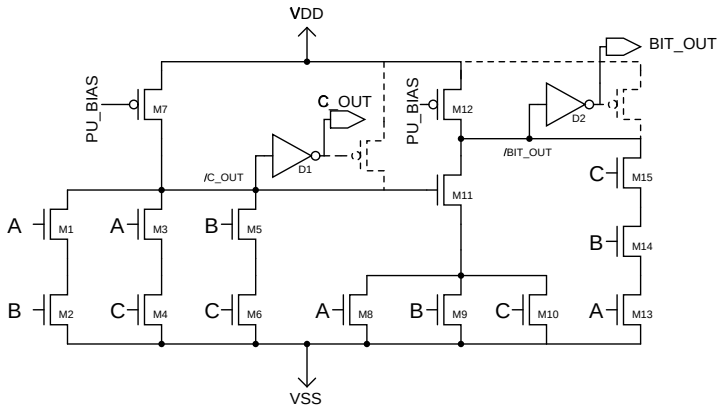

Figure 5.5 Full adder.

maximum magnitude is passed through. Otherwise, the input data is passed to the output of the nonlinear block.

The full adder that was used in the chip realization is shown in Figure 5.5. Input signals *A*, *B* and *C* (carry) are added, which yields output bits BIT_OUT and C_OUT (carry). The full adder is realized as a pseudo NMOS gate [1], since transistors $M7$ and $M12$ are used for pull-up all the time. The bias voltage PU_BIAS is selected so that the pull-up current (and the static power consumption) is small. This full adder can be fitted into a small area, it requires no clocking and the logic levels are almost full. However, it consumes static power. In order to eliminate the static power consumption, this gate can easily be turned into a latched domino logic gate [2]. This is accomplished by adding keeper transistors (shown dashed in the figure) and by replacing PU_BIAS with a clock signal and by increasing the driving capabilities of transistors $M7$ and $M12$.

5.1.3.3 Processing of Data

The cellular array processor can process linear, quadratic and cubic feedback templates simultaneously with the aid of Heun's integration method. The processor works as described in the following procedure. The procedure is carried out so that items 3-5 are repeated for all data blocks before entering the next item. The corresponding timing diagrams are shown in Figure 5.3. The durations of the control signals in the figure are not drawn to proportion.

1. Write initial state data to register REG_Y, set time index n to 1.
2. Load the feedback templates $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$.

3. Use register REG_Y to control rg_C and rg_B (which control the DACs). Convert the result of the analog computation to digital ($r_{i,j}(n)$) and store it to REG_2. Then add the contents of REG_Y and REG_2 and store the result to REG_1.

4. Use REG_1 to control rg_C and rg_B and perform analog computing. Next, convert the sum of currents to digital ($r^*_{i,j}(n+1)$) and add it to the contents of REG_2. Divide the result by two and store it to REG_CMB. Write contents of REG_CMB to REG_2.

5. Add contents of REG_2 and REG_Y, store the result to REG_1 and then copy it to REG_Y.

6. Increment n by unity, return to item 3 if $n \leq nri$; nri is the number of iterations.

7. Read the result of the processing from register REG_Y.

All input-output operations during the integration are local because the memories are located in the PUs. In order to perform a temporally correct integration, data in all blocks always has to correspond to the same iteration step. This holds for Heun's method because when determining the predicted state $x^*_{i,j}(n+1)$, REG_Y is used to control the DAC and results are written to REG_1. Similarly, when determining the corrected state $x_{i,j}(n+1)$, REG_1 controls the DAC and the result is written to REG_2.

5.1.4 Simulation of the Mixed-Mode System

Efforts are put towards developing a simulator that would work in a mixed-mode environment [3], but currently there is much to improve in the simulation time, for example. The system level simulation of the mixed-mode cellular array processor would be very time consuming if an analog simulation tool, such as HSPICE, would be used. In the following, the means that were used to simulate the cellular array processor are summarized.

First, the quantitative performance of individual circuit blocks, such as multipliers and logic gates, was simulated with HSPICE. Then, the quantitative simulations were extended to combinations of circuit blocks to an extend that was reasonable in terms of computing time. A lot of effort was devoted to verify the correct interaction between groups of circuits. After the operation of a PU was verified using simple tests, a model of the whole network was programmed using MATLAB. Then, network level simulations were performed using the MATLAB model. This simulation strategy worked well in the simulations of this thesis.

5.2 Analog Arithmetic Circuits

This section deals with design of the analog arithmetic circuits of the cellular array processor. An overview of the aspects that affect the accuracy of the circuits is given. This is followed by descriptions of the analog multiplier and the circuits that are used to produce the quadratic and cubic terms. Since a digital integrator is used, the sign of the state of a PU is known at all times. Consequently, analog multiplication and analog quadratic and cubic terms can be produced quite straightforwardly in one quadrant and four-quadrant operation is obtained by digitally steering the current into a positive or negative sum node. Therefore, the complexity of analog circuits is reduced at the cost of increased digital hardware. Experimental results of the analog arithmetic circuits are shown in Section 5.5.2.

5.2.1 Accuracy

The accuracy of analog circuits relates directly to the accuracy of the semiconductor models that are used to describe the circuits. In addition to this, random temporal fluctuations (noise) in the performance of a circuit have an effect on the accuracy. Furthermore, random differences in nominally identical transistors affect the accuracy.

The accuracy of the analog circuits and data converters affects the digital rate of change of state $ri_{i,j}(n)$ (see Section 4.1) and, furthermore, the system level performance of the network. When the nonlinearity, mismatch and noise of the circuit are known, the error in $ri_{i,j}(n)$ can be statistically modeled [4] and used to model the network level performance. The accuracy of the analog circuits and data converters can be observed from the measurement results of Section 5.5.2.

5.2.1.1 Effect of Transistor Models

The accuracy of analog circuit design is affected by the models and model parameters (obtained from the process vendor) that are used to describe the devices. The basic characteristics of MOS transistors in saturation and linear regions can be described using simple equations that are well suited for hand calculations. However, when using processes with small gate lengths, several deviations from the simple models need to be considered in the design process. In the following, some important second-order effects are briefly reviewed. A description of these effects is available in [5], for example.

- The threshold voltage of a transistor decreases as the channel length decreases. This is because the depletion regions of the source and drain extend under the

gate and take part in inverting the channel. The larger the channel length, the less significant this phenomenon, and the larger the threshold voltage.

- The threshold voltage of a transistor decreases as a function of the drain-source voltage. This is because a large drain voltage draws minority carriers to the conducting channel in a similar fashion as the gate voltage. This phenomenon is denoted as drain-induced barrier lowering (DIBL). Again, for large channel lengths this effect is less significant.

- The mobility of carriers in the channel of a MOS transistor degrades as a function of the electric field that is applied perpendicular to the channel (vertical field). High vertical fields attract minority carriers into a thin layer next to the oxide and collisions reduce the speed of the carriers.

- The mobility of carriers in the channel of a MOS transistor degrades as a function of the electric field that is applied in the direction of the channel (lateral field). The average speed of carriers in the channel is a product of the mobility and the lateral electric field. However, the maximum average speed is bounded below speeds smaller than about 10^7cm/s. Therefore, when using devices with short gate lengths, the mobility is degraded with rather small drain-source voltages.

- High lateral fields tend to increase the speed of a carrier. Even if the maximum average speed of carriers in the channel is bounded, individual electrons can reach high speeds and, consequently, obtain high kinetic energies. These high energy (hot) electrons collide with silicon atoms and the collisions extricate electron-hole pairs (impact ionization), which contribute to drain and substrate currents. Some of the hot electrons may acquire enough energy to go through the oxide yielding a nonzero gate current. The presence of hot carriers has an impact on the long-term reliability of the devices.

- The effective channel length of a transistor reduces with drain-source voltage since the pinch-off point moves further into the channel. The shorter the channel, the more significant the effect on the effective gate length. Because of the short-channel effect the, output impedance of a transistor increases with drain-source voltage. However, as drain-source voltage is further increased, the DIBL and hot carrier effects tend to lower the output impedance of the device.

The analog circuits in this thesis were designed so that first the circuits were described using simple equations in hand analysis. The impact of the second-order effects on

the behavior of the circuits was considered qualitatively. Then the circuits were simulated using HSPICE with level 50 parameters (Chapter 5) or ELDO with level 59 parameters (Chapter 6). These parameters allow an accurate characterization of the second-order effects of the transistors also. The circuits in Sections 5.2.4 and 5.2.5 are biased to weak inversion region. Therefore, also the EKV model that can model the drain current of a transistor in weak, moderate and strong inversion regions using one equation was used in the hand analysis. Despite of careful process characterization and circuit design, the real characteristics of a circuit can only be revealed by measuring an implemented chip.

5.2.1.2 Noise and Interference

Reference [6] showed that noise does not limit the accuracy of an analog CNN. However, if precautions are not taken, this may not be the case in a mixed-mode design. This is because, even if the noise of the analog devices were negligible, noise (interference) caused by the digital gates may be significant. In many digital CMOS processes, n-type transistors are not built in a well. Also, a low resistivity substrate is required to prevent latch-up. Since the drain of an NMOS is coupled to the substrate via the drain-bulk capacitance, and the substrate is connected to global ground via an inductor (bonding wire), switching digital gates introduce noise to the substrate. Also, high current peaks drawn by standard digital gates during switching further introduce noise to the analog ground/substrate. Additionally, capacitive coupling between fast switching digital signal lines and analog signals introduce noise if precautions are not taken during the layout phase. There are methods to reduce substrate noise, for example, isolation with guard rings [7]. Also, low noise current mode logic [8] can be used in the digital gates to reduce the transient currents. The good thing in a mixed-mode CNN is that, during analog computation, the only digital logic that is allowed to switch is in the A/D converters. In the following, it is assumed that with a careful design of the digital parts in the A/D converters, noise is not an accuracy-limiting factor. In this thesis, layout techniques were used to reduce the interference (see Section 5.4.2).

5.2.1.3 Mismatch

Processing variations introduce random variations to transistor parameters. Therefore, the effect of device mismatch on accuracy is a major issue when it comes to sizing the devices. In this thesis, threshold voltage and current factor mismatch are considered and the accuracies of the analog circuits are derived within the presence of these non-idealities. Device mismatch due to distance of devices is omitted. The variances of

the threshold voltage and current factor mismatches were defined in [9] as

$$\sigma^2(\Delta V_t) = \frac{A_{Vt}^2}{W \cdot L} = 2\sigma^2(V_t) \tag{5.1}$$

and

$$\left(\frac{\sigma(\Delta\beta)}{\beta}\right)^2 = \frac{A_\beta^2}{W \cdot L} = 2\left(\frac{\sigma(\beta)}{\beta}\right)^2, \tag{5.2}$$

where A_{Vt} and A_β are process-dependent constants. $\sigma^2(\Delta V_t)$ is the variance of the threshold voltage between two nominally identical devices and $\sigma^2(V_t)$ is the variance of threshold voltage of an individual device. The variations of the devices are assumed normally distributed and independent. The variances of the current factor are defined similarly.

Even if the mismatch of the analog transistors in this thesis is described with the model of Reference [9], it is useful to acknowledge that the model has some shortcomings when transistors of different geometries and operating points are described. Research is ongoing in order to improve the simple mismatch model of Reference [9]. In Reference [10], the effect of the aspect ratio of the transistor channel for the mismatch was investigated and Reference [11] demonstrated a mismatch model that can be used in all operating regions of a transistor. The mismatch models in References [9], [10] and [11] use threshold voltage and current factor mismatches to characterize the transistor mismatch. Reference [12] proposed a mismatch model that is based on variations of process parameters. It is said that a process parameter-based mismatch model gives more accuracy in predicting the matching of devices of different geometries and biasing conditions.

Once the mismatch of an individual transistor is known, the effect of transistor mismatch to the circuit performance needs to be sought. Different methods for identifying which transistors in a circuit are least tolerant to mismatch have been developed. Reference [13], for example, describes an algorithm for identifying mismatch-relevant transistor pairs. In Reference [14], the effect of a mismatch of individual transistors to circuits, and, furthermore, to systems was derived analytically. In this thesis, the analog circuits are rather simple and single-ended. Therefore, it is quite straightforward to apply the method of Reference [14]. Since a current mirror is extensively used in the cellular array processor, the mismatch of a current mirror is shown in the following.

The difference of drain-source currents in a current mirror with two nominally identical transistors can be derived to [14]

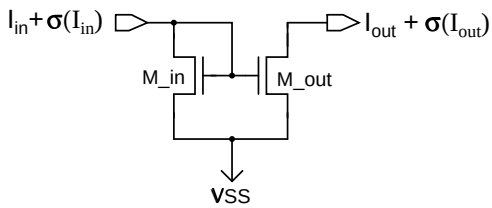

Figure 5.6 Current mirror.

$$\Delta I_{ds} = \frac{\partial I_{ds}}{\partial \beta}\Delta\beta + \frac{\partial I_{ds}}{\partial V_t}\Delta V_t = I_{ds}\frac{\Delta\beta}{\beta} - g_m \cdot \Delta V_t. \tag{5.3}$$

The corresponding standard deviation of the current of an individual transistor is

$$\sigma(I_{ds}) = \frac{1}{\sqrt{2}}\sqrt{\left(I_{ds}\frac{\sigma(\Delta\beta)}{\beta}\right)^2 + \left(I_{ds}\frac{g_m \cdot \sigma(\Delta V_t)}{I_{ds}}\right)^2}. \tag{5.4}$$

Now consider the current mirror shown in Figure 5.6. Assume that input transistor *M_in* is composed of U_{in} identical (unit) transistors in parallel and that U_{out} parallel unit transistors are used to construct the output transistor *M_out*. In Figure 5.6, $\sigma(I_{in})$ and $\sigma(I_{out})$ are the standard deviations of the input and output currents of the current mirror. The mirror factor (gain) of the current mirror is $A = U_{out}/U_{in}$. The standard deviation of the current through a unit transistor is simplified to

$$\sigma(I_{ds,unit}) = \frac{2I_{in} \cdot A_{Vt}}{\sqrt{2 \cdot W \cdot L} \cdot (V_{gs} - V_t)}. \tag{5.5}$$

Equation 5.5 is simplified compared to Equation 5.4, under the assumption that the effect of the current factor mismatch is insignificant compared to the threshold voltage mismatch. This assumption is reasonable if relatively small gate overdrive voltages are used. In that case, the effect of threshold voltage mismatch is dominant compared to the effect of the current factor mismatch [14]. This assumption is utilized also elsewhere in this thesis. The standard deviation of the output current of the current mirror can be determined to

$$\sigma(I_{out}) = \sqrt{\frac{A}{U_{in}}\left(\sigma(I_{ds,unit})\right)^2 + A^2\left[\left(\sigma(I_{in})\right)^2 + \frac{1}{U_{in}}\left(\sigma(I_{ds,unit})\right)^2\right]}, \tag{5.6}$$

where the first term is due to the variances of the parallel output transistors. The second and third terms in Equation 5.6 correspond to the variances of the input current and the input transistors, respectively.

The mismatch-relevancy of individual transistors can be reasoned from the mismatch models of the circuits. Mismatch parameters A_{Vt} and A_β were not available for the process that was used. However, suggestive quantitative estimates of mismatch were computed for the circuits of Sections 5.2.3, 5.2.4 and 5.2.5. In the calculations, the mismatch parameters A_{Vt} and A_β were estimated to 6mVμm and 1.8%μm, respectively. These estimates are based on References [6] and [9]. The mismatch parameters of p- and n-type transistors were assumed equal.

5.2.2 Choice of Multiplier for a Mixed-Mode CNN

5.2.2.1 Commonly Used Multipliers in Programmable CNNs

In CMOS realizations of programmable CNNs, multipliers with the multiplying transistor operating in the linear region are commonly used. Even if the linear region is selected, a choice between many different types of multipliers still exists [15]. The 20×20 CNN [16] used a two-quadrant differential transconductance multiplier multiplexed to four-quadrant operation. Being capable of effectively suppressing second-order nonlinear components, this multiplier provides high linearity at the cost of low signal to bias ratio and large size. In most cases, such high linearity is not needed in a CNN, so it is worth investigating other alternatives. A one-transistor, four-quadrant multiplier [17] was used in, for example, circuits of References [18] and [19]. It highlights a compact size, rather good signal to bias ratio, sufficient linearity and an inherent four-quadrant operation. The downsides are that a current conveyor is needed in each cell and that resistive voltage drops affect the weight distribution.

5.2.2.2 Multiplication in a Mixed-Mode CNN

In a mixed-mode CNN, analog circuits are used to carry the main computational burden; these are complemented with digital circuits. The emphasis is on using simple analog circuits to make the design efforts leading towards low operating voltages easier. Also, if the analog circuits are simple, it may be possible that designs could be re-used in different silicon processes without major modifications in the circuit topologies. In addition to this, compact size and low power consumption are obvious requirements. When the multiplication scheme is considered for a mixed-mode CNN, it is important to notice that the sign of cell output is known at all times. Consequently, a one-quadrant multiplier can be used and four-quadrant operation can be obtained by steering the output current of the one-quadrant multiplier to a positive or negative sum node. This may lead to a good compromise between compactness, signal to bias ratio

and accuracy. A one-quadrant multiplier is also easy to use with one-quadrant circuits used to produce the polynomial terms. A one-quadrant analog transconductance multiplier was chosen for the cellular array processor realization. It is a single-ended variant of the multiplier used in [16].

5.2.3 Multiplier Structure

Figure 5.7 depicts a four-quadrant multiplier that uses an analog one-quadrant transconductance multiplier, current steering switches and digital logic. The one quadrant multiplier is composed of analog transistors MR, $M1$ and $M2$. Transistor MR is used to convert current I_{in} to voltage. Transistor $M1$ is biased to the linear region and transistor $M2$ (in saturation) sets voltage $V_{ds,1}$ (drain-source voltage of transistor $M1$) to a value that corresponds to the desired weight. Bias voltage V_r is used to set the channel resistance of transistor MR to a desired level and bias voltage V_c sets the bias current of transistor $M1$. Multiplier bias voltages V_r and V_c are globally denoted as $VRESX1$ and $VCMX1$ (see Table B.8). Because the weight voltage V_w is connected to the gate of transistor $M2$, it does not draw any current. However, bias voltage V_c is attached to the source of transistor MR. Therefore, the driver of voltage V_c has to be strong enough, while resistive drops in the distribution of voltage V_c have to be minimized in the layout phase. Voltages V_c, V_r and V_w are referenced to VSS.

Four-quadrant operation is obtained using transistors $M3$-$M6$ that steer current I_d to one of four scaling nodes. In these scaling nodes, the current is scaled by 1, 4, -1 or -4 using current mirrors, and the scaled currents are combined. The bias current component of the combined current is removed by sampling (see Section 5.3). The current scaling circuits $1X$, $4X$, $-1X$ and $-4X$ are shared by several multipliers. Signals $S+$, $S4+$, $S-$ and $S4-$ control the current steering transistors $M3$, $M4$, $M5$ and $M6$. The digital control logic uses the sign bit of input data ($x[7]$ or $rgi[7]$), sign bit of weight W_{sign} and msb of weight W_{msb} as inputs. The contents of control blocks $CT1$-$CT4$ are shown in Figure 5.7 surrounded by a dotted line. Pull-down transistor MC is biased to weak inversion so that a low static power dissipation is obtained. Inputs a and b are conditional pull-up signals. Later in this section, simulation results of the one-quadrant multiplier are shown. The simulations were carried out by using HSPICE level 50 parameters of a standard $0.25\mu m$ digital CMOS process and a 3V operating voltage.

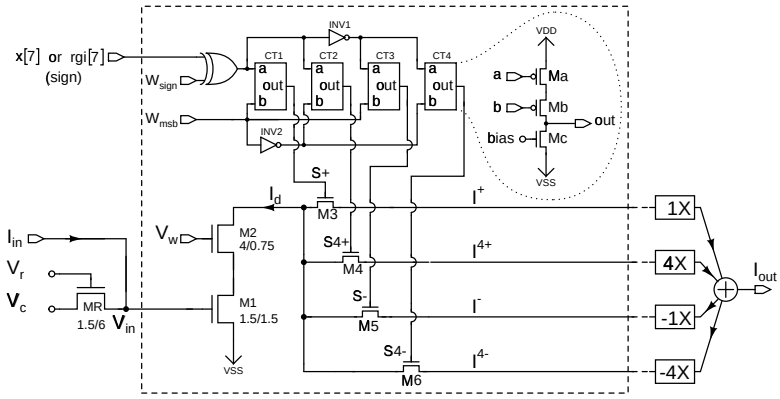

Figure 5.7 One quadrant multiplier, current steering switches and control logic. The sizes (W/L, in μm) of the analog transistors are also shown.

5.2.3.1 Input-Output Characteristics

The one-quadrant multiplier of Figure 5.7 is composed of an I/V converter and a variable gain V/I converter in cascade. The transfer curve of the I/V converter (transistor *MR*) is obtained by noting that $V_{gs,1} = V_{ds,R} + V_c$. Since transistor *MR* is biased to the linear region, voltage $V_{gs,1}$ can be derived to

$$V_{gs,1} = V_r - V_{t,R} - \sqrt{(V_{rc} - V_{t,R})^2 - \frac{2I_{in}}{\beta_R}}, \tag{5.7}$$

where $\beta_R = \mu_n C_{ox} W_R/L_R$ is the current factor of transistor *MR* and $V_{rc} = V_r - V_c$. The channel resistance of transistor *MR*, namely R_R, is defined as $\partial V_{gs,1}/\partial I_{in}$. It can be verified that $\partial R_R/\partial I_{in}$ and $\partial^2 R_R/\partial I_{in}^2$ are positive. Therefore, R_R increases as a function of I_{in} so that the speed of increase of R_R also increases with I_{in}. This can be observed from the dashed curve in Figure 5.8 that shows $V_{gs,1}$ as a function of current I_{in}.

As a first-order approximation, the current through transistors $M1$ and $M2$ is

$$I_d = \beta_1 (V_{gs,1} - V_{t,1}) V_{ds,1} - \beta_1 \frac{V_{ds,1}^2}{2}, \tag{5.8}$$

where

$$V_{ds,1} = V_w - V_{t,2} - \sqrt{2I_d/\beta_2}. \tag{5.9}$$

Therefore, voltage $V_{ds,1}$ decreases with current I_d . The fact that $V_{ds,1}$ is a function of I_d causes nonlinearity because reduction in $V_{ds,1}$ is equivalent to lowering the weight.

What further decreases $\partial I_d/\partial V_{gs,1}$ with I_d is that the mobility of electrons of transistor $M1$ is reduced as a function of gate voltage due to the vertical electric field. However, the nonlinearity of the I/V converter counteracts this and some linearization takes place. Figure 5.8 shows simulations of I_d vs. I_{in} with weight voltages V_w ranging from 0.55V to 0.75V with increments of 50mV.

Transistor $M1$ is in the linear region if $V_{gs,1} > V_{ds,1} + V_{t,1}$. Since $V_{ds,1}$ increases with V_w, transistor $M1$ is closer to saturation region the larger the weight and smaller the input. Close to saturation, the current through transistor $M1$ is a combination of characteristics of the linear region (Equation 5.8) and square law characteristics. The square law term counteracts the effect of the term $-\sqrt{I_d}$ in Equation 5.9 and some linearization takes place. This can be seen from the top solid curve (corresponding to largest weight, closest to saturation) in Figure 5.8. First the curve bends upwards (effect of the square law term) and, when $I_{in} \approx 1\mu A$, the effect of the square law term becomes less significant than the square root term of Equation 5.9 and the curve starts to turn downwards. Even if some linearization takes place, in some applications the linearity of this multiplier may not be adequate.

The signal to bias ratio of the simulated one-quadrant multiplier increases approximatively linearly with the weight, being 0.6 with $V_w = 0.55$V and 3.8 with weight $V_w = 0.75$V. In order to keep the signal to bias ratio as large as possible, the weight range is extended by scaling the current conditionally up by four. If compared to a two quadrant multiplier, the input voltage range of the V/I converter is effectively doubled because the input range is folded. That is because only half of the total input voltage range has to fit into the linear region of the V/I converter. Also, when a four-quadrant multiplier is formed by multiplexing a one-quadrant multiplier, the nonlinearities are odd-symmetric around origin.

5.2.3.2 Accuracy

Next, the accuracy of the multiplier is determined. The accuracy of the I/V converter is assessed by using a simplified expression for the voltage over transistor MR, namely

$$V_{ds,R} = \frac{I_{in}}{\beta_R(V_{rc} - V_{t,R})}, \tag{5.10}$$

where $V_{rc} = V_r - V_c$. Variation of $V_{ds,R}$ in two nominally identical transistors MR due to mismatch is determined to

$$\Delta V_{ds,R} = \frac{\partial V_{ds,R}}{\partial \beta_R}\Delta\beta_R + \frac{\partial V_{ds,R}}{\partial V_{t,R}}\Delta V_{t,R} + \frac{\partial V_{ds,R}}{\partial I_{in}}\Delta I_{in}, \tag{5.11}$$

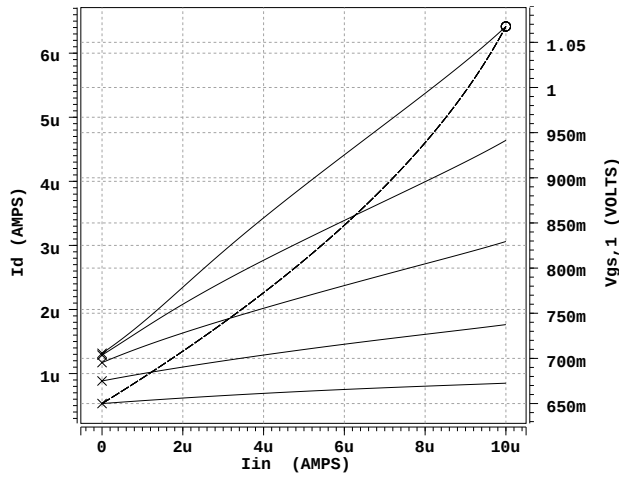

Figure 5.8 Multiplier transfer curves and $V_{gs,1}$ vs. I_{in}.

which yields

$$\Delta V_{ds,R} = V_{ds,R}\left(-\frac{\Delta\beta_R}{\beta_R} + \frac{\Delta V_{t,R}}{V_{rc}-V_{t,R}} + \frac{\Delta I_{in}}{I_{in}}\right). \tag{5.12}$$

The standard deviation of voltage $V_{ds,R}$ is thus

$$\sigma(V_{ds,R}) = \frac{V_{ds,R}}{\sqrt{2}}\sqrt{\left(\frac{\sigma(\Delta\beta_R)}{\beta_R}\right)^2 + \left(\frac{\sigma(\Delta V_{t,R})}{V_{rc}-V_{t,R}}\right)^2 + \left(\frac{\sigma(\Delta I_{in})}{I_{in}}\right)^2}. \tag{5.13}$$

This is present at the gate of transistor $M1$. Therefore, the accuracy of the I/V converter is optimized by using a large gate overdrive voltage. The difference of current I_d between two identical multipliers is described by

$$\Delta I_d = \frac{\partial I_d}{\partial \beta_1}\Delta\beta_1 + \frac{\partial I_d}{\partial V_{ds,1}}\Delta V_{t,2} + \frac{\partial I_d}{\partial V_{gs,1}}\Delta V_{ds,R}. \tag{5.14}$$

The threshold voltage mismatch of transistor $M1$ is not considered because it affects the bias current of the multiplier and is removed by sampling. Also, the current factor mismatch of transistor $M2$ is omitted since $V_{gs,2}$ is small (β_2 is large). The relative standard deviation of current I_d can be approximated using

$$\frac{\sigma(I_d)}{I_d} = \sqrt{\left(\frac{\sigma(\beta_1)}{\beta_1}\right)^2 + \left(\frac{\sigma(V_{t,2})}{V_{ds,1}}\right)^2 + \left(\frac{\sigma(V_{ds,R})}{V_{gs,1}-V_{t,1}}\right)^2} \tag{5.15}$$

where the first and third terms are determined using Equation 5.8 and by assuming that $V_{ds,1} \ll V_{gs,1} - V_{t,1}$. Thus, the relative standard deviation of I_d is minimized by using large weights. Also, the range of $V_{gs,1}$ should be as large as possible. However, the range of $V_{gs,1}$ is constrained by linearity requirements.

Current I_d is mirrored once or twice depending on the sign of the multiplication. Assuming that U_{in} is unity and that identical (unit) transistors are used in parallel, according to Equation 5.6, the relative standard deviations of the output current of the multiplier with scaling factors 1, 4, -1 and -4 are

$$\frac{\sigma(I_{out,1})}{I_{out,1}} = \frac{1}{I_{out,1}}\sqrt{(\sigma(I_d))^2 + 2\,(\sigma(I_{ds,P}))^2}, \tag{5.16}$$

$$\frac{\sigma(I_{out,4})}{I_{out,4}} = \frac{1}{I_{out,4}}\sqrt{16\left[(\sigma(I_d))^2 + (\sigma(I_{ds,P}))^2\right] + 4\,(\sigma(I_{ds,P}))^2}, \tag{5.17}$$

$$\frac{\sigma(I_{out,-1})}{I_{out,-1}} = \frac{1}{I_{out,-1}}\sqrt{(\sigma(I_{out,1}))^2 + 2\,(\sigma(I_{ds,N}))^2} \tag{5.18}$$

and

$$\frac{\sigma(I_{out,-4})}{I_{out,-4}} = \frac{1}{I_{out,-4}}\sqrt{(\sigma(I_{out,4}))^2 + 2\,(\sigma(I_{ds,N}))^2}, \tag{5.19}$$

respectively. Terms $\sigma(I_{ds,P})$ and $\sigma(I_{ds,N})$ in Equations 5.16 - 5.19 are the standard deviations of currents in p- and n-type unit transistors in saturation. Using the equations provided, the accuracy of the multiplier can be estimated.

Table 5.1 shows numerical estimates of relative standard deviations of the output current of the multiplier and voltages that were used in the estimation. The standard deviations were determined using the largest weight and the maximum input current that were used in the simulation of Figure 5.8. The estimations are based the equations above and the voltage levels are obtained from the simulation of Figure 5.8. The current mirrors in the sum block of Figure 5.13 are composed of p- and n-type transistors, the sizes (W/L, in μm) of which are 6/2.5 and 5/3. Using these dimensions and a gate overdrive voltage of 250mV, standard deviations $\sigma(I_{ds,P})$ and $\sigma(I_{ds,N})$ were computed. In Section 5.5.2, the relative standard deviation at the output of a multiplier was measured to 4.4%. The estimated and measured results cannot be compared directly because the measured figures also include the deviations of the data converters. However, they are in the same order of magnitude.

Voltage or Ratio	Value @ maximum I_{in}
$V_{rc} - V_{t,R}$	1500mV
$V_{ds,R}$	420mV
$\frac{\sigma(\Delta I_{in})}{\Delta I_{in}}$	0
$\sigma(V_{ds,R})$	1.8mV
$V_{ds,1}$	70mV
$V_{gs,1} - V_{t,1}$	510mV
$\frac{\sigma(I_d)}{I_d}$	3.62%
$\frac{\sigma(I_{ds,P})}{I_{ds,P}}$	0.88%
$\frac{\sigma(I_{ds,N})}{I_{ds,N}}$	0.88%
$\frac{\sigma(I_{out,1})}{I_{out,1}}$	3.82%
$\frac{\sigma(I_{out,4})}{I_{out,4}}$	3.75%
$\frac{\sigma(I_{out,-1})}{I_{out,-1}}$	4.02%
$\frac{\sigma(I_{out,-4})}{I_{out,-4}}$	3.95%

Table 5.1 Relative standard deviations of the output current of the multiplier and voltages that were used in the estimation.

5.2.4 Circuit for Producing the Quadratic Term

Since one-quadrant analog operation is required from the circuit for producing the quadratic term, the bias currents are low compared to, for example, the two-quadrant current squarer of [21]. In the following, the current squarer that was used in the cellular array processor is analyzed and simulation results are shown.

5.2.4.1 Current Squarer

The quadratic term was realized using a circuit shown in Figure 5.9. Transistors MR (I/V converter) and $M1$ (nonlinear V/I converter) produce the square of the input current and transistors $M2$ and $M3$ mirror the output current I_{x^2} to a multiplier. P-type transistor MR is biased to the linear region so that $V_r - V_c < V_{t,R}$. The current through transistor MR is

$$I_{in} = \beta_R \left[\frac{V_{sd,R}^2}{2} + (V_{cr} + V_{t,R}) V_{sd,R} \right], \tag{5.20}$$

where $\beta_R = \mu_p C_{ox} W_R / L_R$ is the transconductance parameter of transistor MR and $V_{cr} = V_c - V_r$. The bias voltages V_r and V_c are chosen so that $V_{cr} + V_{t,R}$ is much larger than $V_{sd,R}$, so that the I/V converter is approximately linear. Transistor $M1$ is biased into the weak inversion region using bias voltages V_c and $Vss2$. Bias voltages V_r, V_c and $Vss2$

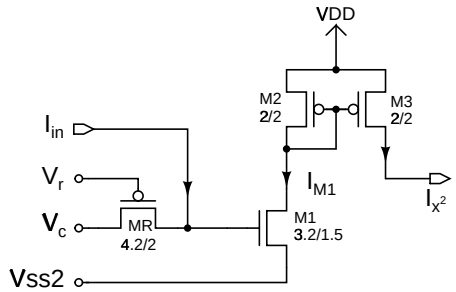

Figure 5.9 The circuit for generating the quadratic term of cell output current. The sizes (W/L, in μm) of the transistors are also shown.

of the current squarer are globally denoted as $VRESX2$, $VCMX2$ and $VSS2$. They are referenced to VSS (bulk). The current through a transistor can be accurately modeled by the Enz-Krummenacher-Vittoz (EKV) transistor model [22], but the EKV model is also useful in hand calculations. That is because one equation can be used to describe the current of a transistor in weak, moderate and strong inversion regions. The current through transistor $M1$ is

$$I_{M1} = I_S \left[\ln \left(1 + \exp \left[\frac{V_p - V_{s,1}}{2U_T} \right] \right) \right]^2, \tag{5.21}$$

where U_T is the thermal voltage, $V_{s,1} = V_{vss2}$, I_S is a specific current proportional to slope factor n_s and V_p is the pinch-off voltage proportional to $V_g = V_{s,R}$. The slope factor is

$$n_s = 1 + \frac{\gamma}{\sqrt{V_p + 2\Phi_F}}, \tag{5.22}$$

where γ is the body effect factor and Φ_F is the bulk Fermi potential. In the EKV model all voltages are referenced to bulk.

Equation 5.21 shows that as voltage V_g (and consequently V_p) increases, the current characteristics of transistor $M1$ shift continuously from the exponential current growth of weak inversion to square law characteristics of strong inversion. The voltage range of V_g has to be carefully selected in order for I_{M1} to obtain close resemblance to a quadratic current. Transistor $M1$ is biased into weak inversion so that $\delta I_{M1}/\delta I_{in}$ at $I_{in} = 0$ would be as small as possible, while with $I_{in} > 0$ the current would have a close resemblance to an ideal quadratic term. Because transistor $M1$ is biased into the weak inversion region, the operating point current is small. Comparison of the HSPICE-simulated and ideal quadratic terms of Figure 5.10 shows that the simulated quadratic

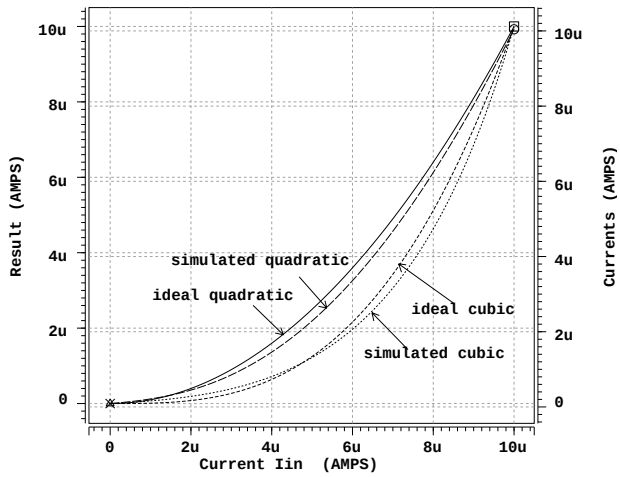

Figure 5.10 Simulated and ideal quadratic and cubic terms.

term approximates fairly well to the ideal quadratic term. In the simulations, a 3V operating voltage and level 50 parameters of a $0.25\mu m$ digital CMOS process were used. The voltage range of $V_{g,1}$ was from 600mV to 920mV (referenced to bulk) and $Vss2$ was 0.1V. The nominal threshold voltage of a p-type transistor in the process is 0.55V. The measured and ideal quadratic terms are available in Figure 5.23 for comparison.

5.2.4.2 Accuracy

Next, the accuracy of the current squarer is determined. The standard deviation of $V_{sd,R}$ of transistor MR is

$$\sigma(V_{sd,R}) = \frac{V_{sd,R}}{\sqrt{2}}\sqrt{\left(\frac{\sigma(\Delta\beta_R)}{\beta_R}\right)^2 + \left(\frac{\sigma(\Delta V_{t,R})}{V_{cr}+V_{t,R}}\right)^2 + \left(\frac{\sigma(\Delta I_{in})}{I_{in}}\right)^2}. \tag{5.23}$$

It was derived using Equation 5.20 by assuming that $V_{sd,R} \ll V_{cr}+V_{t,R}$. Like Equation 5.13, Equation 5.23 also is minimized with large gate overdrive voltages. The standard deviation of current I_{M1} of the current squarer was determined so that the current factor mismatch of transistor $M1$ was discarded and threshold voltage mismatch was used. That is because $V_{g,1}$ is small and the threshold voltage mismatch cannot be removed by sampling because of the nonlinear V/I characteristics of transistor $M1$. The square law equation of the strong inversion region was used in computing the standard deviation of I_{M1} since the transistor operates mostly in this region. The relative standard deviation of the current with large values of I_{in} can be approximated by

Voltage or Ratio	Value @ maximum I_{in}
$V_{cr}+V_{t,R}$	950mV
$V_{sd,R}$	320mV
$\frac{\sigma(\Delta I_{in})}{\Delta I_{in}}$, current squarer	0
$\sigma(V_{ds,R})$	1.5mV
$V_{gs,1}-V_{t,1}$	260mV
$\frac{\sigma(I_{x^2})}{I_{x^2}}$	2.02%
$\frac{\sigma(\Delta I_{in})}{\Delta I_{in}}$, multiplier	$\frac{\sigma(I_{x^2})}{I_{x^2}}$
$\frac{\sigma(I_{out,4})}{I_{out,4}}$	4.1%
$\frac{\sigma(I_{out,-4})}{I_{out,-4}}$	4.29%

Table 5.2 Relative standard deviations of the output current of a current squarer and the output current of a multiplier that is controlled by a current squarer. Also shown are the voltages used in computing the estimate.

$$\frac{\sigma(I_{M1})}{I_{M1}} = \frac{2}{(V_{gs,1}-V_{t,1})}\sqrt{(\sigma(V_{t,1}))^2+(\sigma(V_{sd,R}))^2}, \tag{5.24}$$

where $V_{gs,1} = V_c + V_{sd,R} - Vss2$. Since the output current of transistor $M1$ is sensitive to the gate overdrive voltage, the swing of $V_{gs,1}$ should be as large as possible. The standard deviation at the output current of the current squarer $\sigma(I_{x^2})$ is obtained by adding the contributions of transistors $M2$ and $M3$ to $\sigma(I_{M1})$. When a current squarer is used to control a multiplier, the standard deviation of the output current of the multiplier increases. Table 5.2 shows the estimated relative standard deviation of the current squarer. It also shows an estimate of the relative standard deviation of the output current of a multiplier that is controlled by a current squarer. The voltages used in the estimates are from the simulation shown in Figure 5.10. In Section 5.5.2, the measured relative standard deviation of a multiplier that is controlled by a current squarer was determined to 7.3%. Even if the measured figure includes the deviations of the data converters, the estimate produced analytically is too optimistic. This is probably because transistor $M1$ in the current squarer is biased to weak inversion region and the simple square law model of Equation 5.24 is not accurate enough. It is also possible that in the measurement the internal signal ranges of the current squarer were not the same as in the simulations.

5.2.5 Circuit for Producing the Cubic Term

Figure 5.11 shows the circuit used for producing the cubic term of input current I_{in}. The circuit is otherwise similar to that used for the quadratic term but in this case

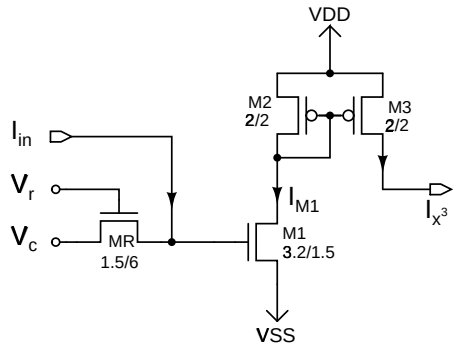

Figure 5.11 The circuit for generating a cubic term of input current I_{in}. The sizes (W/L, in μm) of the transistors are also shown.

the input current I_{in} is converted to a voltage with an n-type transistor MR and $V_{s,1}$ is connected to ground. Bias voltages V_r and V_c of the circuit are globally denoted as $VRESX3$ and $VCMX3$. In Section 5.2.3, it was shown that the channel resistance of n-type transistor MR increases as a function of I_{in}, so that the speed of the increase is faster the larger $V_{ds,R}$ is. This provides extra boost to $\partial I_{M1}/\partial I_{in}$ at large I_{in} and helps the output current gain resemblance to the ideal cubic term. This can be verified from Figure 5.10, which shows the HSPICE simulated and ideal cubic terms. Again, since the operating point of transistor $M1$ is biased into the edge of the subthreshold region, the bias current is very small. The voltage range of $V_{g,1}$ extended from 460mV to 780mV. The nominal threshold voltage of an n-type transistor is 0.56V. The accuracy of the circuit can be defined as for the circuit for producing the quadratic term. An estimate was computed for the relative standard deviation at the output of the circuit for producing the cubic term and for a multiplier that was controlled by such a circuit. Table 5.3 shows the results and the voltages used in the computation. The measured relative standard deviation of a multiplier that is controlled by a circuit for producing the cubic term is 7.4%. This is shown Section 5.5.2. As in case of the current squarer, the computational estimate of the accuracy gives too optimistic figures.

5.3 Data Converters

A/D and D/A converters are needed in this design to interface between the analog arithmetic circuits and the digital integrator. In addition to this, A/D conversion is also utilized in subtracting the bias current from the input current of a PU. Therefore, the A/D converter block contains two converters, namely ADC_Q that is devoted to eliminating the bias currents and ADC_D to sample the state derivative. Since the

Voltage or Ratio	Value @ maximum I_{in}
$V_{rc} - V_{t,R}$	1540mV
$V_{ds,R}$	320mV
$\frac{\sigma(\Delta I_{in})}{\Delta I_{in}}$, cubic circuit	0
$\sigma(V_{ds,R})$	1.4mV
$V_{gs,1} - V_{t,1}$	220mV
$\frac{\sigma(I_{x3})}{I_{x3}}$	2.29%
$\frac{\sigma(\Delta I_{in})}{\Delta I_{in}}$, multiplier	$\frac{\sigma(I_{x3})}{I_{x3}}$
$\frac{\sigma(I_{out,4})}{I_{out,4}}$	4.20%
$\frac{\sigma(I_{out,-4})}{I_{out,-4}}$	4.38%

Table 5.3 Relative standard deviations of the output current of the circuit for producing the cubic term and the output current of a multiplier that is controlled by such a circuit. Also shown are the voltages used in computing the estimate.

integrator operates with two's complement numbers, the output of ADC_D is converted to two's complement form before it is fed to the integrator and the D/A converters take two's complement numbers as inputs. In the mixed-mode CNN architecture, the A/D and D/A converters are used to determine intermediate processing values also, instead of just the final processing result. Therefore, a relatively high conversion speed is required. Since A/D and D/A converters are included in every processing unit, they must occupy a small die area and have a low power consumption. A 7-bit binary weighted current D/A converter is used in this design. The eighth bit (sign bit) of the D/A conversion is realized by the current steering switches of the multipliers. The A/D converter used here is a successive approximation register (SAR) ADC with 7-bit resolution.

5.3.1 Choice of A/D Converter for a Mixed-Mode CNN

An A/D converter in a mixed-mode CNN must occupy a small die area and have a low power consumption. Simultaneously, a relatively fast conversion speed is required but the accuracy requirement is moderate (around 7-8 bits). The same kind of performance metrics are required from converters in pixel-level A/D converters. In [23], an area efficient single slope ADC was used in which a global analog ramp voltage is compared to sensor output. When the comparator switches, a global gray code that changes with the analog ramp is latched into the in-sensor memory. In [23], the conversion time was 20μs. A similar type of converter is multi-channel bit-serial A/D converter, which in [24] achieved a 10μs conversion time. The difference from a mixed-mode CNN is

that, in the sensor application, the converter is not used in the computation of intermediate values but just for converting the final result. Therefore, the speed requirements are not so strict. The conversion time using approaches like [23] or [24] cannot be significantly reduced because the determination of one bit takes several clock cycles. A faster A/D converter should be used in a mixed-mode CNN. In a flash A/D converter the whole conversion is performed during one clock cycle. However, flash converters occupy too much area and the power consumption is too high, even at moderate accuracies.

A good compromise between speed and die area can be achieved when a successive approximation register (SAR) ADC or algorithmic (cyclic) ADC is used [25]. A SAR-type ADC, for example, is capable of converting one bit during one clock cycle. These types of converters can be built either in current or voltage mode. Current mode converters are the preferred choice for a mixed-mode CNN: couplings between PUs are represented using currents that are summed at the ADC input and linear resistors (for I/V conversion) are difficult to build using a digital CMOS process. In Reference [26], current mode algorithmic converters are analyzed and in Reference [27] a current mode, SAR-type converter is presented. The operation of SAR and algorithmic converters rely on the same principles: the conversion is determined bit by bit starting from the most significant bit (*msb*). In the cellular array processor realization, a current mode SAR-type ADC was used because it provides a good combination of speed, die area and accuracy.

The current entering the A/D converter may contain bias currents (operating point currents) or offset currents (due to device mismatch), which have to be eliminated before the actual sum of currents can be converted. These can be eliminated with, for example, current sampling techniques using analog current memories. The hold time of the current memory sets constraints to the speed of the control circuitry that is used to control the mixed-mode CNN. Here the offset/quiescent currents are eliminated by sampling with another ADC. Since the result of the A/D conversion is stored into a static memory, this is a time-independent method. Therefore, the speed requirements of the control/evaluation setup are relaxed. It turned out that this facilitated the debugging and evaluation phase significantly.

5.3.2 Asynchronous Control Signal Generator

The control signals of the A/D converters are generated asynchronously on chip. Therefore, fast conversion speeds are available even if the control setup is slow. The realization of the asynchronous control signal generator is depicted in Figure 5.12.

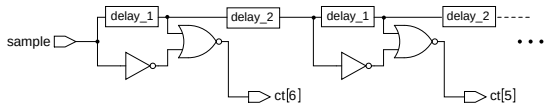

Figure 5.12 Asynchronous control signal generator.

The delay block consists of a current-starved inverter [28] (current in pull-down path is limited) and a regular inverter in cascade. The fall time of the current-starved inverter is controlled by an adjustable NMOS current source. When signal *sample* is turned HI, signal $ct[6]$ turns HI for a time determined by block *delay_1*. When the signal has propagated through the delay block, $ct[6]$ turns LO. Then the signal starts to propagate through block *delay_2*. This delay block is included in order to guarantee that the control signals do not overlap. After the delay, control signal $ct[5]$ is activated and the signal generation proceeds similarly for all subsequent control signals $ct[4,0]$. Control signal $ct[0]$ is also driven to a pad and can be measured outside the chip. Therefore, the actual conversion time and pulse ratio can be measured.

5.3.3 Current Mode SAR A/D Converter Block

Figure 5.13 shows the contents of the *ADC* block. It contains two current mode 7-bit SAR-type A/D converters (ADC_Q and ADC_D). ADC_Q is composed of register block *reg_Q* and block *cs_Q* that contains binary weighted current sources. Similarly, ADC_D is composed of blocks *reg_D* and *cs_D*. Furthermore, the converters share the asynchronously generated control signals $ct[6,0]$ and use the same current comparator. The sum block removes part of the bias current using current source transistors $M+$, $M4+$, $M-$ and $M4-$. The remaining currents at nodes I^+, I^-, I^{4+} and I^{4-} are scaled using current mirrors with mirror factors of +1, -1, +4 or -4, respectively. The magnitudes of V_{C+}, V_{C4+}, V_{C-} and V_{C4-} are determined in a dummy PU outside the processor grid. The dummy PU is shown in Figure 5.17 of Section 5.4.2. The W/L ratios of transistors $M+$, $M4+$, $M-$ and $M4-$ were chosen to be 87% of the width/length ratios of transistors $M1$-$M4$ in Figure 5.17. Only the bias currents corresponding to the quadratic terms are subtracted using these current sources because the bias currents due to linear and cubic terms are signal (sign) dependent.The amount of bias current in the resulting current I_{sum} is approximately

$$I_{q,sum} = \sum_{k=1}^{3}\sum_{l=1}^{3}\left[\left|\left[I_q(A_{k,l}^{(1)}) + I_q(A_{k,l}^{(3)}\right]\right| + 0.13\left|I_q(A_{k,l}^{(2)})\right|\right], \qquad (5.25)$$

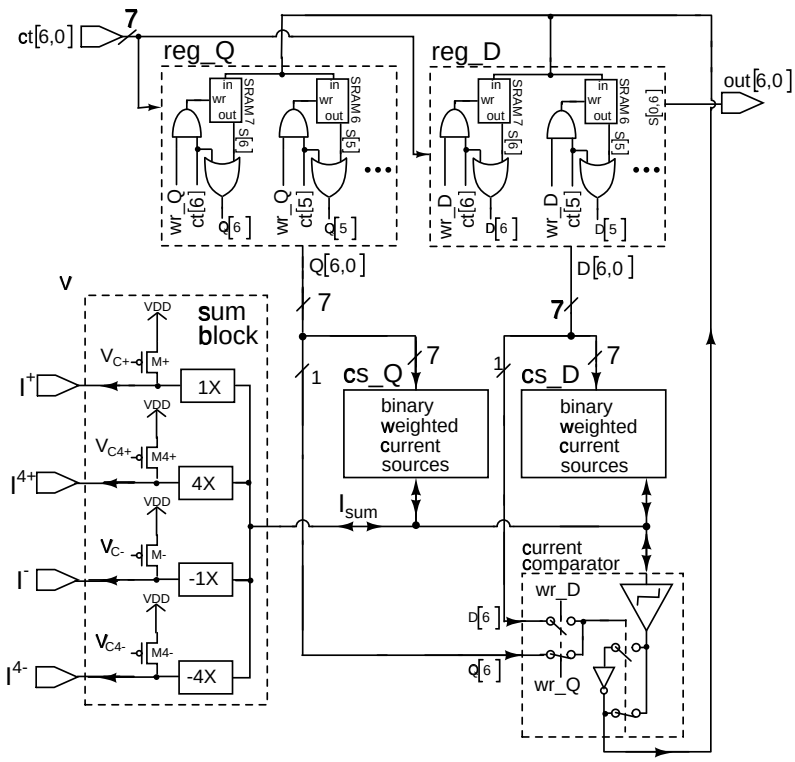

Figure 5.13 Block-level description of the SAR type A/D converter.

where $I_q(A_{k,l}^{(1)})$, $I_q(A_{k,l}^{(2)})$ and $I_q(A_{k,l}^{(3)})$ are the bias current components of $I_{q,sum}$ due to linear, quadratic and cubic couplings. $I_{q,sum}$ is the bias current that is left for ADC_Q to remove.

Register block *reg_Q* in ADC_Q is composed of logic gates and seven SRAMs. The logic AND gates are used to control the writing of the SRAMs using control signal *wr_Q* and the asynchronous control signals $ct[6,0]$. The outputs of *reg_Q* are defined by a logic OR function of SRAM outputs and the asynchronous control signals. The conversion proceeds bit by bit towards *lsb* as control signals corresponding to lower bits activate. Output $Q[6]$ of *reg_Q* (the sign bit of the conversion) is used to set the direction of the output current of current source block *cs_Q*. Outputs $Q[5,0]$ are used to control the binary weighted current sources. ADC_D works similarly to ADC_Q. If $out[6]$ is HI, the outputs $out[5,0]$ are inverted and appended by unity before they are taken to the integrator (conversion from sign and magnitude to two's complement number representation). Half adders are used to perform the addition by unity. The half adder is constructed in a similar fashion to that of the full adder shown in Figure 5.5. Figure

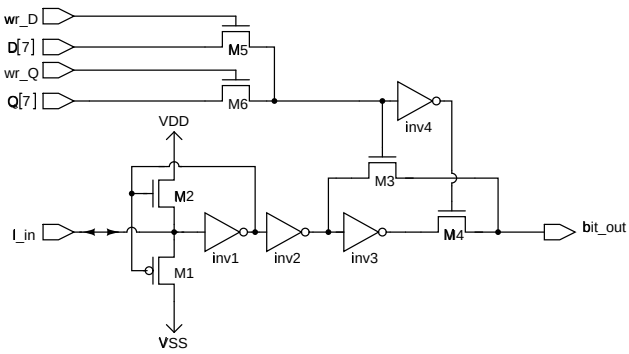

Figure 5.14 Current comparator.

5.14 depicts the current comparator. As in Reference [29], transistors $M1$ and $M2$ are included to reduce the voltage swing of the input node *I_in*. This way, faster operation is obtained. Depending on which one of signals *wr_D* and *wr_Q* is active, one of the sign bits $Q[6]$ or $D[6]$ is used to choose the polarity of the comparator output.

The current sources of blocks *cs_D* and *cs_Q* are as depicted in Figure 5.15 (drawn with denotations of *cs_D*). Output currents of binary weighted current sources (transistors $M1$-$M6$) are steered either to transistor $M7$ or to the output. This way the current through the current sources stays more stable during conversion. The current sources are scaled so that the W/L ratios of transistors $M1$-$M6$ in *cs_D* are twice as large as those of *cs_Q*. Therefore, scaling parameter k_A (in Equation 4.6) of ADC_D is twice as large as that of ADC_Q. Because of this, ADC_Q can convert small currents with a better resolution than ADC_D.

5.3.4 D/A Converter

Figure 5.16 shows the 7-bit binary weighted current D/A converter used in this design. The eighth bit (sign bit) of the D/A conversion is realized by the current steering switches of the multipliers shown in Figure 5.7. Since the digital integrator operates with two's complement numbers, the D/A converter is able to use them as inputs. The binary weighted current sources $M1$-$M7$ are controlled with $ci[6,0]$. $ci[6,0]$ equals the digital cell output ($x[6,0]$ or $rgi[6,0]$) if the sign bit $S[7]$ ($x[7]$ or $rgi[7]$) is LO. Otherwise $ci[6,0]$ equals the inverted digital cell output. The conversion from two's complement to sign and magnitude representation is completed so that a current corresponding to *lsb* is added to the D/A converter output current. This is performed by controlling current source transistor $M8$ with the sign bit $S[7]$. Analog addition of unity is more compact than when half adders are used. Because linear, quadratic and

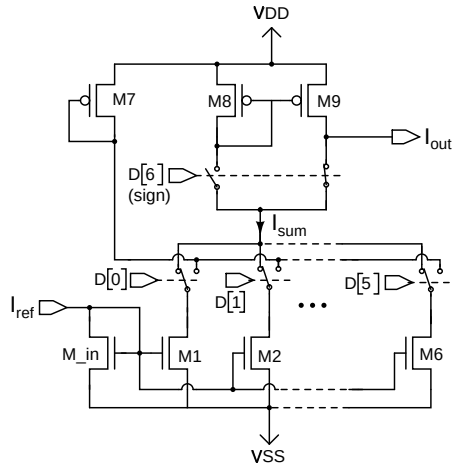

Figure 5.15 The current source block in the ADC (drawn with denotations of *cs_D*).

cubic couplings are used in this design, the converter has three equal output currents I_x, I_{out2} and I_{out3}. Output current I_x is taken directly to multipliers. Output currents I_{out2} and I_{out3} are forwarded to the circuits that produce the quadratic and cubic terms, respectively.

5.4 Peripheral Devices and Layout Design

As Figure 5.1 shows, the array of PUs is complemented by decoders, digital memories for storage of weights, D/A converters for converting the digital weights to analog voltages and an ADC control signal generator. Also included on chip is a dummy PU. The decoders were included in order to distinguish the row, column and the layer of a PU in a random access fashion. Furthermore, a decoder was put into the chip to control the writing of the weights. The 27 D/A converters that are used to produce the analog weight voltages were built using binary weighted current sources. The current was transformed to voltage using a similar one-transistor I/V converter that was used in the multiplier. The control signal generator is depicted in Section 5.3. The rest of this section shows the dummy PU that was included outside the array of PUs. Also, the layout arrangement of PUs and the layout of one PU are shown.

5.4.1 Dummy Processing Unit

Figure 5.17 depicts the dummy PU that was placed to the periphery of the chip. The purpose of the dummy PU is to provide the sum block shown in Section 5.3.3 with

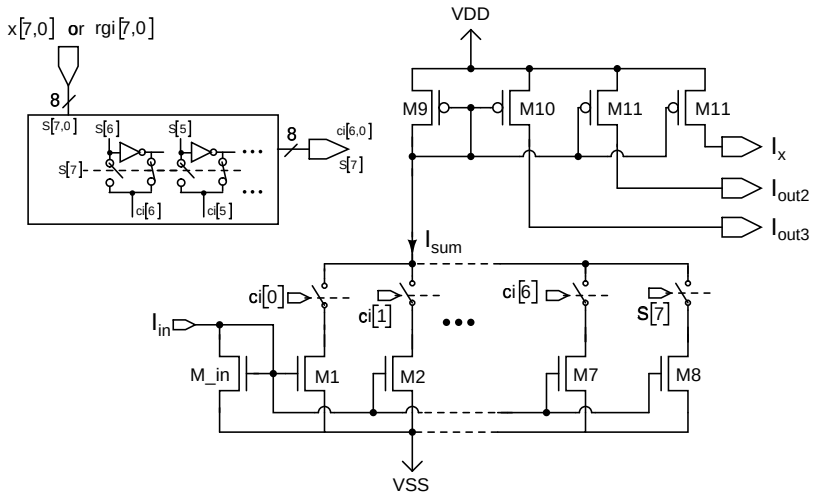

Figure 5.16 Current mode D/A converter.

voltages V_{C+}, V_{C4+}, V_{C-} and V_{C4-}. It can also be used to test the analog arithmetic circuits and the DAC. The dummy PU contains register REG_B that is used to control the DAC and the multipliers. The three output currents of the *DAP* block are turned into voltages by the I/V converters of the multipliers. The corresponding voltages V_x1, V_x2 and V_x3 can be measured from outside the chip. As can be reviewed from Figure 5.7, each multiplier has four output currents. In the dummy PU, the four output currents of all 27 multipliers are added together and mirrored using transistors $M1$-$M4$. The I^+ output current can be measured from outside the chip via transistor $M5$, the drain of which is connected to a pad. During processing, voltages V_x1 and V_x3 are connected to ground and zero is written to REG_B. This way, only the bias currents corresponding to the quadratic term are mirrored and, consequently, contribute to voltages V_{C+}, V_{C4+}, V_{C-} and V_{C4-}. This is as required in Section 5.3.3.

5.4.2 Arrangement and Layout of PUs

A prototype chip as described in Sections 5.1, 5.2 and 5.3 was manufactured using a standard 6-metal, single poly digital 0.25μm CMOS process. The chip contains 2×72 PUs arranged in four rows of 2×18 PUs. Figure 5.18 illustrates the arrangement of the PUs by showing a 2×4 network arranged in two rows of 2×2 PUs. The orientations of adjacent PUs are different, for example $PU(1,1)$ is horizontally flipped compared to $PU(2,1)$. The left and right edges of the 2×4 network are connected because the border condition is cyclic in the horizontal direction.

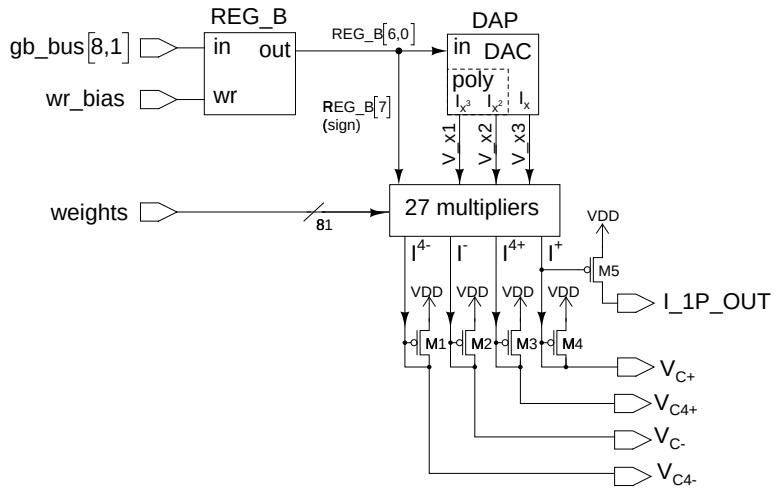

Figure 5.17 Dummy processing unit.

A microphotograph of the chip is shown in Appendix A. The aspect ratio of the implemented PU is relatively small so that a rectangular die is obtained. Figure 5.19 shows the layout of one PU without the global wiring. The width and length of the PU are 131μm and 320μm, respectively. Different circuit blocks are circumscribed and identified in the figure. Each PU has 7092 transistors in an area of 0.042mm^2 and the processor core of 144 processing units occupies 6.1mm^2. The total transistor count is 1.027 million and with pads, template memories and other supporting hardware the design occupies an area of 10mm^2. The whole layout was manually drawn in order to minimize the area.

In order to reduce interference, guard rings were used to isolate the analog and digital parts of the design. In addition to this, the analog and digital operating voltages and grounds were distributed to the chip using separate wires. In order to reduce the inductance of the bonding wires, several pins were used to bring the operating voltages and ground levels to the chip. Eight pins were used for both the 3V digital operating voltage and the digital 2.5V operating voltage, while seventeen pins were used for digital ground. Eight pins were also used for the 3V analog operating voltage, while nine pins were used for the analog ground.

5.5 Experimental Results

In this section, measurement results of the chip are shown. The chip uses three power supplies, 3V in analog parts, 2.5V in the digital integrator and 3V in the logic of the

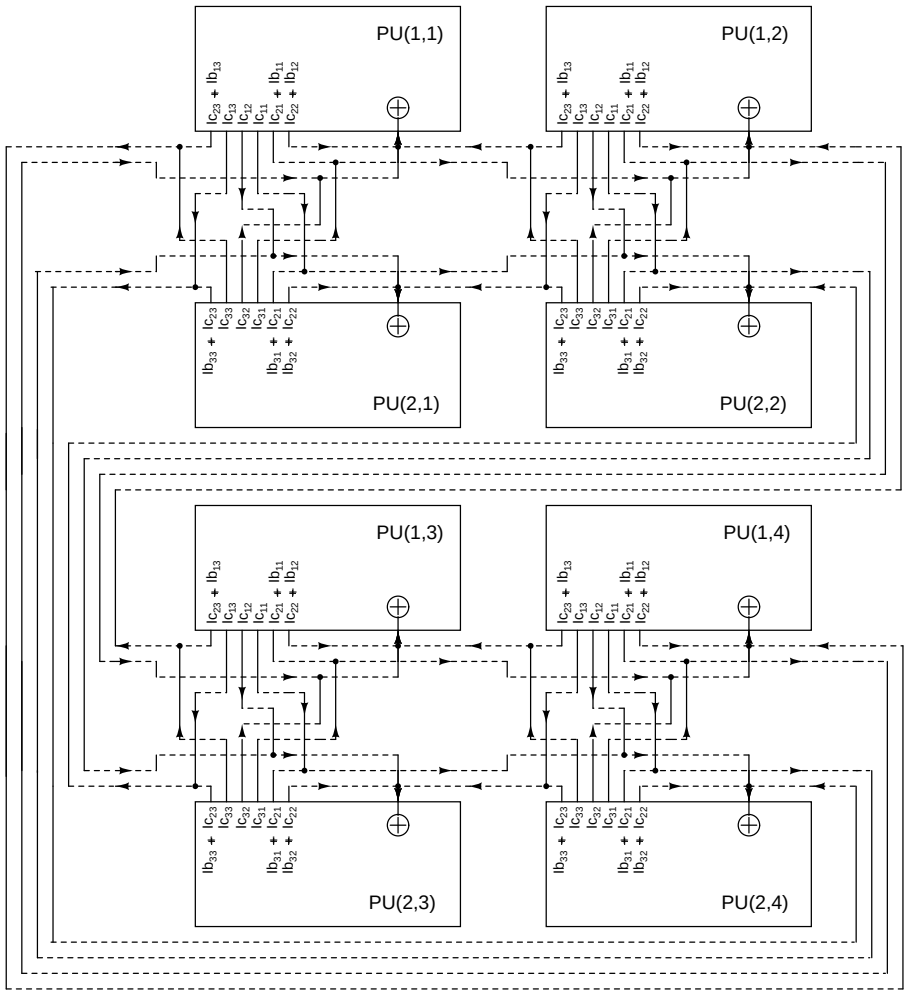

Figure 5.18 Illustration of the arrangement of the PUs by showing a 2×4 network arranged in two rows of 2×2 PUs.

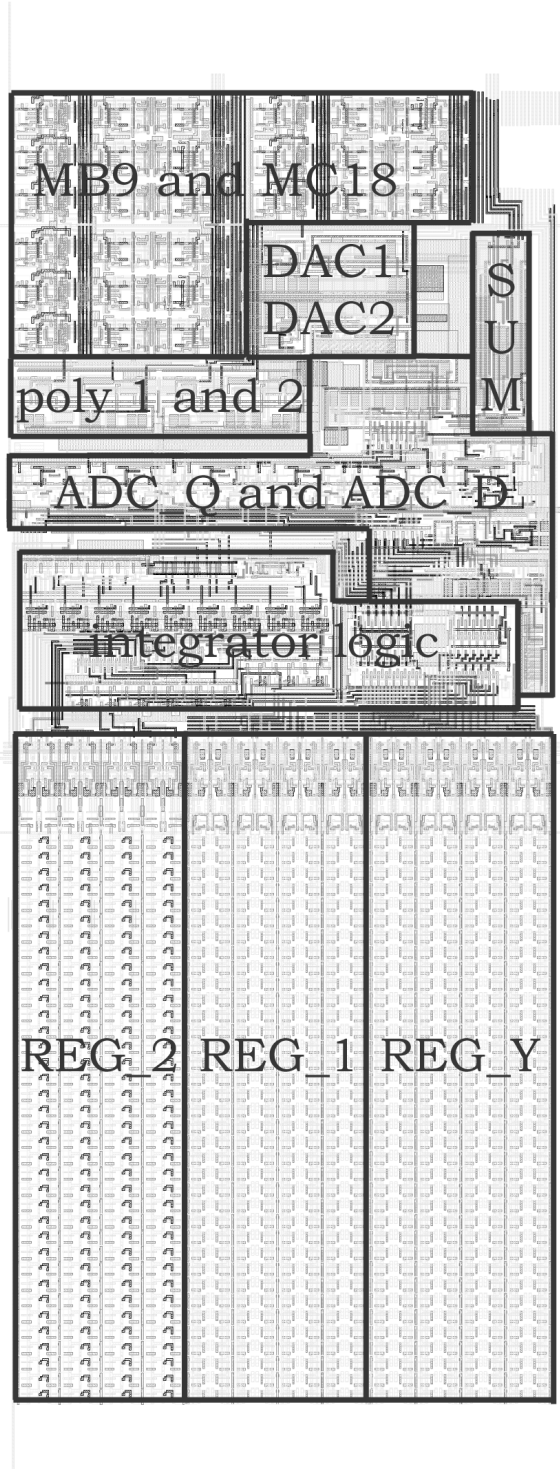

Figure 5.19 Layout of a mixed-mode processing unit.

A/D and D/A converters. The three 36×8-sized SRAMs within each processing unit can be randomly accessed for I/O operations. Also, the SRAMs in ADC_Q of each PU can be read. The functional testing of the chip was carried out by using a measurement board that was attached to a PC. The power consumption and speed measurements were completed by using a pattern generator for controlling the fast signals.

5.5.1 Measurement Arrangement

Appendix B lists the digital control signals and bias/reference voltages that are needed to use the chip. The control and I/O of the chip is completely digital. Since all memories on chip are SRAMs, and the fast control signals of the ADC are produced asynchronously on chip, functional measurements can be carried out using a slow control setup. Appendix C depicts the measurement board that was constructed for the measurements. A PC was used to control the chip via the measurement board and to show the measurement results. This provided a flexible, easily programmable, measurement environment. The speed and power measurements were completed so that the fast switching integrator signals were generated using a pattern generator.

5.5.2 Analog Parts and Data Converters

Figure 5.20 shows measured A/D output codes vs. D/A input codes with different weights controlling the multiplier. The measurement was carried out so that $A_{22}^{(1)}$ was altered and all other weights were zero. First the bias current was eliminated with ADC_Q and then $DAP1$ was used to control a multiplier within a PU. The resulting current was sampled with ADC_D. The voltage swing at the input of the analog multiplier was 0.28V, while the analog weights in this example ranged from 0.56 to 0.68V. It should be noted that the currents entering a processing unit are summed before the A/D conversion. Consequently, the contribution of small weights is also significant. Figure 5.21 shows the signal to bias ratio with 17 different weights, namely 0, ± 0.4 , ± 0.55, ± 0.75, ± 1, ± 1.6, ± 2.3, ± 3 and ± 4. In this measurement, the magnitude of the state of the processing unit (D/A input) was at maximum. The ratio can be used to assess the power efficiency. If no current scaling were performed in the multipliers, the power efficiency with small weights would be worse.

The next measurement was carried out to assess the accuracy of the A/D and D/A converters and the linearity of the multiplier with different weights. A DAC was used to control a multiplier and the result was A/D converted. The same was repeated for weights 1.6, 2.3, 3 and 4. For each weight, the ADC reference current was set so that the maximum DAC input code corresponded to the maximum ADC output code. The

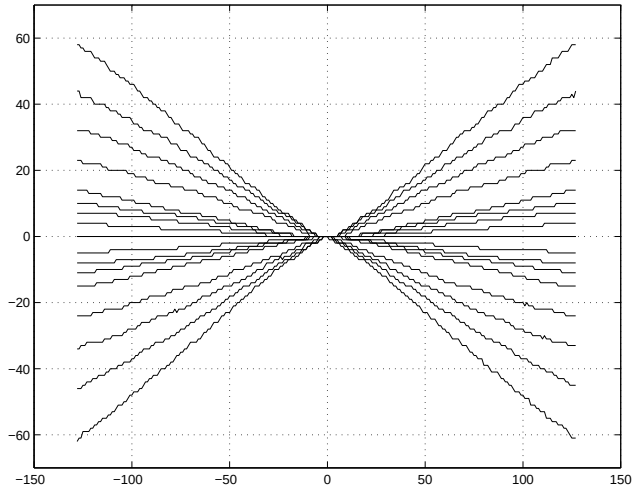

Figure 5.20 Measured A/D output codes vs. D/A input codes with different weights controlling the multiplier.

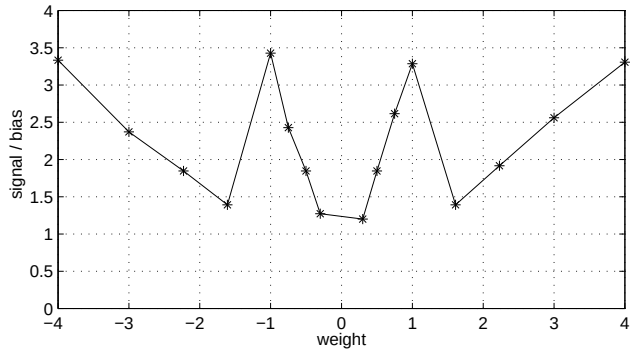

Figure 5.21 Measured signal to bias ratio of the multiplier with weights ranging from -4 to 4.

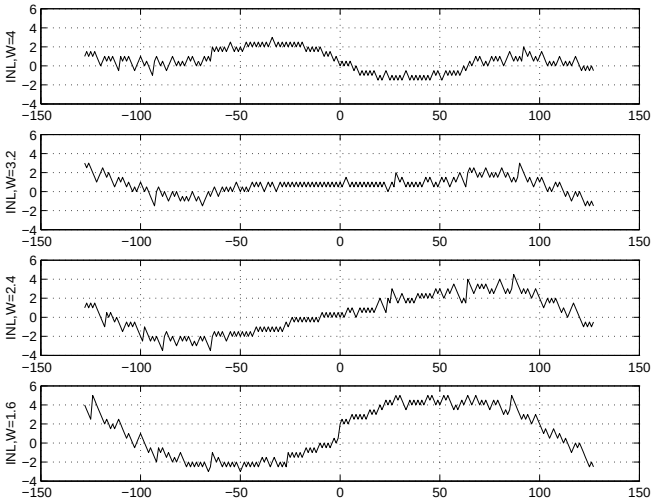

Figure 5.22 Difference of measured A/D output and desired A/D output vs. D/A input codes for weights 1.6, 2.3, 3 and 4.

aim was to keep the reference current as large as possible in order to maintain the conversion speed. Therefore, in this measurement, ADC_D was used to remove the bias current, while ADC_Q was used to sample the state derivative. Figure 5.22 shows the ADC_Q output codes subtracted by the desired A/D output code vs. D/A input codes for weights 1.6, 2.3, 3 and 4. This differs from standard INL because, in addition to nonidealities of the A/D converter, the multiplier and D/A converter affect the result. Since the large weights are formed by scaling, the nonlinearities of weights 1 and 4, for example, are the same.

Figure 5.23 shows the quadratic and cubic couplings measured with the largest positive weight from a multiplier within a processing unit. The bias current was eliminated with ADC_Q, while *DAP*1 was used to generate the quadratic and cubic activation functions. The resulting current was sampled with ADC_D. First, the multiplier connected to the quadratic self-feedback term was controlled with the largest positive weight (four), while the cubic term was controlled with zero weight. The A/D-converted result appears as the dashed curve in the figure. The procedure was repeated for the cubic term (solid curve). Ideal quadratic (dash-dot curve) and cubic (dotted curve) terms are included in the figure for comparison.

Data for assessing the accuracy of the analog couplings was measured so that the D/A converters in all PUs were controlled with maximum magnitude. Also, weight $A_{22}^{(1)}$ was at maximum magnitude (four) and all other weights were zero. The multiplier output current was measured (with the ADCs of the PUs) from all four quadrants and

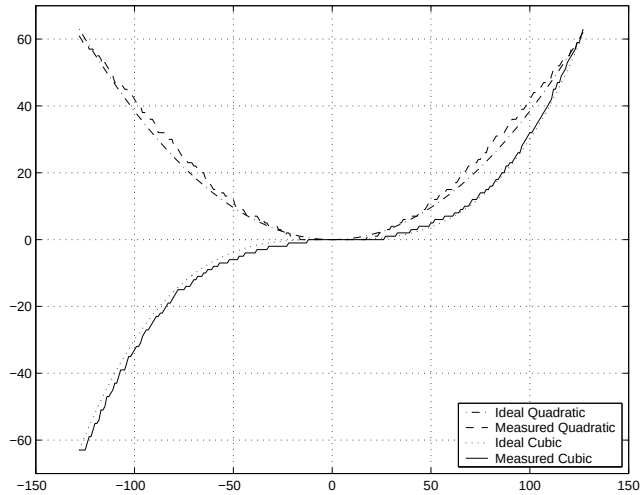

Figure 5.23 Measured A/D output codes vs. D/A input codes with quadratic and cubic couplings.

all processing units of a chip. The same was repeated with weights $A_{22}^{(2)}$ and $A_{22}^{(3)}$. The ratio of standard deviation and absolute value of mean for linear couplings was 4.4%, for square couplings 7.3% and for cubic couplings 7.4%. These figures include the data of 144 processing units measured from the same chip. In all other cases, the sampling time in the measurements of this section was 3.3μs, except the INL measurement with weight 1.6 was measured with a sampling time of 5.9μs.

5.5.3 Integrator

The evolution of the state of an individual PU can be easily tracked. The following example is included in order to demonstrate this feature. Figure 5.24 shows the evolution of the state of a PU when $r_{ii,jj,kk}(n)$ increases with n. The measurement was carried out so that the state $x_{ii,jj,kk}(n)$ was initialized to 70 (the allowed range extends from -128 to 127). Then a template with positive cubic self-feedback was run and integration was performed. The solid line represents the state evolution when Heun's method of integration is used and the dashed line is measured using Euler's integration method. The slope of the solid curve increases faster because of the prediction/correction property of Heun's method.

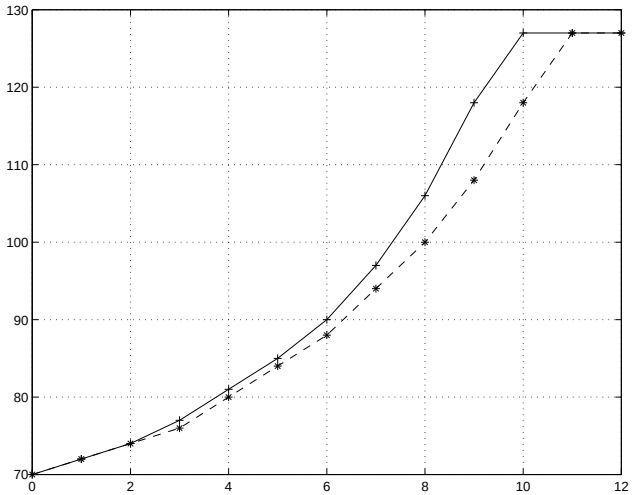

Figure 5.24 Measured state of a processing unit vs. iteration round. The solid curve is integrated with Heun's method and the dashed curve with Euler's method.

5.5.4 Processing of Data

Even if the chip is designed to process non-topographically arranged data, in order to illustrate the operation, it is meaningful to show measured examples of data processing in which the results can be visually verified. As Equation 4.6 shows, the integration step h can be altered with the aid of k_A. Furthermore, k_A can be scaled with the ADC reference current. In this measurement, the integration step was scaled to 0.1 ($k_A = 10$) and the sampling time was 700ns. Here a faster sampling time was possible than with the measurements of Section 5.5.2 because the higher ADC reference current also speeds up the A/D conversion. A pattern generator was used to produce the integration signals; the time of performing one complete iteration round with Heun's method of integration was 166.4μs. Figure 5.25 shows the 72×72 data that was obtained after 220 iterations with linear and cubic positive coupling to SE neighbor when the initial data was such that one pixel (5,67) was initialized to black (-128), pixel (55,60) to white (127) and the rest of the pixels were initialized to gray (zero). The processing example shown in Figure 5.25 demonstrates that couplings between different data blocks (data is processed in blocks of 2×72) and the horizontal cyclic boundary work.

The next processing example demonstrates the use of the quadratic and cubic activation functions. In the following, the previous processing result was used as the initial state of the next operation. Pixels, (65,68) and (66,68) were initialized to white, pixels (69,68) and (70,68) to black and the rest to gray. Figure 5.26a) presents a 10×9 data

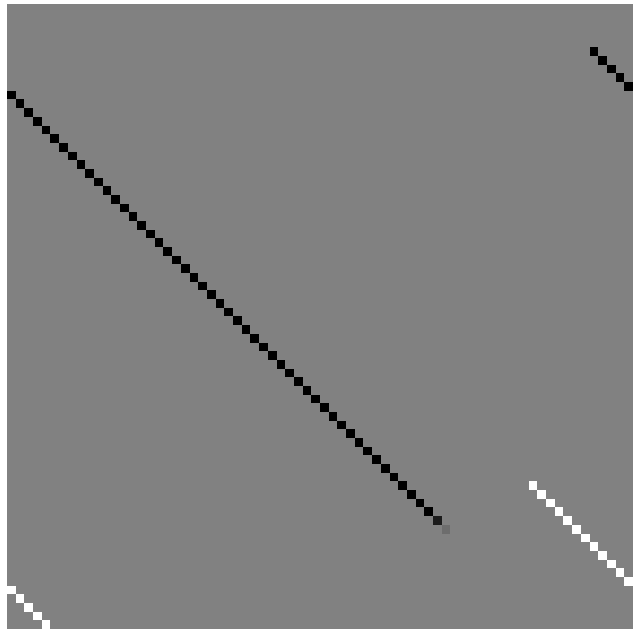

Figure 5.25 Data after growing 220 iterations to SE direction.

showing these pixels. Figure 5.26b) shows the data after running five iterations with a linear negative coupling to W and E directions and a cubic positive self-feedback. Figure 5.26c) shows the data after running five iterations with a negative cubic self-feedback and a weak linear positive self-feedback. This reduces the contrast. Figure 5.26d) shows the data after five iterations with a positive quadratic self-feedback and a weak positive self-feedback. This makes the white areas whiter.

A problem with the implemented chip is that the bias voltages $VRESX1$, $VRESX2$ and $VRESX3$ of the multiplier and the polynomial circuits were delivered to the PUs

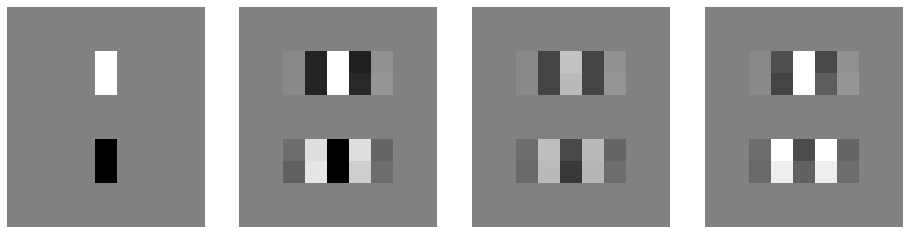

Figure 5.26 Processing of data: a) initial state, b) growing to W and E directions, c) contrast reduction, d) whitening of white pixels.

a) b)

Figure 5.27 Initial 72×72 data. Measured low pass filtered data.

using too narrow wires. Therefore, when there are a lot of nonzero states of PUs, the bias voltages do not stay constant. This results in the subtraction of bias currents not working properly. The problem can be overcome by scaling down the DAC reference currents and by increasing the resistances of the I/V converters. The resistances can be increased by lowering gate voltages $VRESX1$, $VRESX2$ and $VRESX3$ of the transistors that are used for I/V conversion. However, this slows down the speed and degrades the quality of the polynomial terms.

Figure 5.27 demonstrates the processing of a 72×72 data in which there are a lot of nonzero states. It was measured with downscaled DAC reference currents and the slow control setup. The processor was initialized with data shown in Figure 5.27a). A low pass filtering template was programmed to the network. Five iterations with Heun's method were processed. The measured low-pass filtered data is shown in Figure 3b).

5.5.5 Processing Speed and Power Consumption

The maximum processing speed and power consumption were measured by using a pattern generator for controlling the fast switching integrator signals. As mentioned above, the time of performing one complete iteration round with Heun's method of integration was $166.4\mu s$. The sampling speed of ADC_Q and ADC_D was 700ns. Prior to conversion with ADC_Q or ADC_D, 100ns was allocated for analog computation. Remembering that the sampling of the bias current is performed before every sampling of signal current and that Heun's integration method is a two-step process, the sampling of both ADC_Q and ADC_D takes 1600ns during the processing of one data block,

together contributing 72.2% of the total time of an iteration round. The changing of the layers takes 1020ns per data block which contributes 22% of the total time of the iteration round. The rest of the 166.4μs goes to digital integration. Two things made the speed slower than expected. Changing the layer took a long time because too weak drivers were used in the decoders. Also, the 700ns sampling speed was worse than the simulated 200ns; the degradation was probably due to too weak drivers of the A/D control signal generator. It should be noted that here the bias current was sampled before each sampling of the signal current. If an analog current memory were used to sample the bias current, the speed would increase. The computing speed of the cellular array processor can also be described using multiplication and addition operations per second (XOPS) [30]. Since the computing time of one Heun's iteration for a 72×72 data is 166.4μs, the peak computing speed is $72\cdot72\cdot29\cdot2\cdot2/166.4\mu s = 3.6$G XOPS. It was calculated by assuming that all 27 multipliers and both polynomial terms are used.

The power consumption during processing was measured by setting the magnitudes of all weights to unity and by initializing the 72×72 data randomly to have a value between -128 and 127. Then the Heun's integration algorithm that was used in the previous processing examples was modified so that contents of REG_Y were not updated. Otherwise the integration algorithm remained the same and the bias settings were the same as in the processing examples of Figures 5.25 and 5.26. The power consumption was measured while the integration algorithm was running in a loop. The measured analog and digital power consumptions were 154.2mW and 37.7mW, respectively, yielding a total power consumption of 191.9mW. Therefore, the chip can perform 18.7G XOPS/W.

5.5.6 Discussion

The proof of concept cellular array processor described in this chapter gives some insight into the potential computing capabilities of this type of design. The chip was designed to enhance the ease of testing; all memories on chip are static, for example. The first objective of this chapter was to demonstrate that the computing capabilities required by the epilepsy application (see Chapter 3) can be realized with the mixed-mode CNN architecture. Even if a lot of optimizations can be performed to improve the design, the measured chip showed good performance.

The second objective was to give data for assessing the accuracy and applicability of a mixed-mode CNN, in other words, to indicate whether reliable results can be obtained even if analog processing is used. The statistical data provided in Section

5.5.2 can be used for this purpose. The accuracies of the linear, quadratic and cubic couplings were described by showing the measured standard deviation over mean values. If more accuracy is needed, the matching of analog transistors can be improved by scaling up their sizes.

The sizes (W/L, in μm) of transistors $M1$ and $M2$ in the multiplier of Figure 5.7 are 1.5/1.5 and 4/0.75, respectively. Noting that transistors $M1$ and $M2$ occupy only 25% of the area of the multipliers, accuracy can be improved without a large penalty in terms of the area. Also, as can be observed from the figures, the sizes of transistors in Figures 5.9 and 5.11 as well as the size of transistor $MR1$ in Figure 5.7 are small. Since the total area of a PU is 0.042mm^2, these transistors can also be increased without a large effect on the total area. However, in any case, the accuracy is ultimately limited by processing variations and this prevents the use of this kind of a mixed-mode architecture for applications requiring high accuracy. The analog transistors occupy roughly 17% of the total area of a PU. Therefore, the design scales down relatively well with the process.

5.5.7 Summary of Characteristics

In this section, characteristics of the implemented cellular array processor (CAP) are summarized. Also provided is the corresponding information of selected array computer realizations of other research groups. Since the CAP is the first implemented CNN chip with polynomial feedback templates, a direct comparison of the chips is not meaningful. However, Table 5.4 summarizes some common characteristics that can be used to assess the CAP design with respect to the state of the art. Only realizations that can process gray-scale information and have programmable templates are included in the table.

The digital emulated CNN (CASTLE [31]) realization holds 24 PUs on chip. Each PU has an arithmetic unit including a digital multiplier. A section of data is stored on chip, while the main memory is outside the chip. Euler's integration method is used to update the FSR state data. The three analog realizations shown in Table 5.4 all employ continuous-time analog integration to update the cell state and use analog multipliers. The APAP chip [16] is the only one that realizes the CNN dynamics according to the original CNN theory, while the ACE chips rely on the FSR model. The virtual array size shown in Table 5.4 is used to describe how large the data can be for the array to process it without having to rely on global I/O operations. The I/O of CASTLE, ACE16k and CAP is digital, whereas APAP and ACE4k transmit and receive data in analog form. The weights are conveyed in analog form to the APAP, whereas the rest

Design	CASTLE [31]	APAP [16]	ACE4k [18]	ACE16k [19]	CAP
Technology	5M 0.35	2M 0.7	3M 0.5	5M 0.35	6M 0.25
Array Size	24	20×20	64×64	128×128	2×72
Virtual Array Size	N/A	N/A	N/A	N/A	72×72
# of Transistors	N/A	N/A	$\approx 10^6$	$3.75 \cdot 10^6$	$1.027 \cdot 10^6$
# of Transistors per PU	N/A	N/A	172	198	7092
Array Area in mm^2	N/A	24	51	91	6.1
Die Size in mm^2	100	N/A	87	130	10
PUs/mm^2	N/A	16.7	81	180	23.6
State Representation	digital	analog	analog	analog	digital
State Dynamics	FSR	CNN	FSR	FSR	FSR
Integration Method	Euler	CT analog	CT analog	CT analog	Heun
Number of Multipliers/PU	1	10	20	12	27
I/O	digital	analog	analog	digital	digital
# of bits to represent state	12	N/A	N/A	N/A	8
weight programmability	digital	analog	digital	digital	digital
Polynomial Order	1	1	1	1	1,2 and 3
Available Templates	**A**,**B**,*I*	**A**,**B**,*I*	**A**,**B**,*I*	**A**,**B**,*I*	3×**A**
Power consumption	N/A	150mW	1.2W	4W	192mW

Table 5.4 Summary of characteristics of selected array processors.

of the chips have digitally programmable weights.

References

[1] T. A. DeMassa, Z. Ciccone, Digital Integrated Circuits, New York: Wiley, 1996, pp. 467-472.

[2] J. A. Pretorius, A. Shubat, C. Salama, 'Latched Domino CMOS Logic', *IEEE Journal of Solid-State Circuits*, Vol. 21, pp. 514-522, 1986.

[3] T. Grasser, S. Selberherr, 'Mixed-Mode Device Simulation', *Proceedings of 22nd International Conference on Microelectronics*, Nis, Serbia, 2000, Vol. 1, pp. 35-42.

[4] P. Földesy, 'Statistical Error Modeling of CNN-UM Architectures: The Grayscale Case', *Proceedings of the Seventh IEEE International Workshop on Cellular Neural Networks and their Applications*, Frankfurt, pp. 475-482, 2002.

[5] B. Razavi , *Design of Analog CMOS Integrated Circuits*, . McGraw-Hill, 1997.

[6] A. Rodriguez-Vazquez, et al., 'Mismatch-Induced Tradeoffs and Scalability of Mixed-Signal Vision Chips', *IEEE International Symposium on Circuits and Systems*, Scottsdale, USA, Vol. 5, pp. 93-96, 2002.

[7] D. K. Su, et al. 'Experimantal Results and Modeling Techniques for Substrate Noise in Mixed-Signal Integrated Circuits', *IEEE Journal of Solid-State Circuits*, Vol. 28, pp. 420-430, 1993.

[8] J. Musicer, J. Rabaey, 'MOS Current Mode Logic for Low Power, Low Noise CORDIC Computation in Mixed-Signal Environments', *International Symposium on Low Power Electronics*, Rapallo, Italy, pp. 102-107, 2000.

[9] M. Pelgrom, A. Duinmaijer, A. Welbers, "Matching Properties of MOS Transistors", *IEEE Journal of Solid-State Circuits*, Vol. 24, pp. 1433-1440, 1989.

[10] S. J. Lowett, M. Welten, A. Mathewson, B. Mason, 'Optimizing MOS Transistor Mismatch', *IEEE Journal of Solid-State Circuits*, Vol. 33, pp. 147-150, 1998.

[11] J. A. Croon, et al., 'An Easy-to-Use Mismatch Model for the MOS Transistor', *IEEE Journal of Solid-State Circuits*, Vol. 37, pp. 1056-1064, 2002.

[12] P. G. Drennan, C. C. McAndrew, 'Understanding MOSFET Mismatch for Analog Design', *IEEE Journal of Solid-State Circuits*, Vol. 38, pp. 450-456, 2003.

[13] F. Schenkel, M. Pronath, H. Graeb, K. Antreich, 'A Fast Method for Identifying Matching-Relevant Transistor Pairs', *Proc. of the IEEE Custom Integrated Circuits Conference*, San Diego, USA, pp. 361-364, 2001.

[14] P. Kinget, M. Steyaert, *Analog VLSI Integration of Massive Parallel Processing Systems*. Dordrecht: Kluwer, 1997.

[15] G. Han, E. Sanchez-Sinencio, 'CMOS Transconductance Multipliers: A Tutorial', *IEEE Transactions on Circuits and Systems II*, Vol. 45, pp. 1550-1563, 1998.

[16] P. Kinget, M. Steyaert, 'An Analog Parallel Array Processor for Real-Time Sensor Signal Processing, *Proc. of the IEEE Solid-State Circuits Conference,* San Francisco, USA, pp. 92-93, 1996.

[17] R. Dominquez-Castro, A. Rodriguez-Vazguez, S. Espejo, R. Carmona, 'Four-Quadrant One-Transistor-Synapse for High-Density CNN Implementations, *Proceedings of the Fifth IEEE International Workshop on Cellular Neural Networks and their Applications*, London, pp. 243-248, 1998.

[18] G. Linan et al., "A 0.5μm CMOS 10^6 Transistors Analog Programmable Array Processor", *Proc. of the* 25^{th} *European Solid-State Circuits Conference*, Duisburg, Germany, pp. 358-361, Sept. 1999.

[19] G. Linan, et al., 'ACE16K: an Advanced Focal-Plane Analog Programmable Array Processor', *Proc. of the* 27^{th} *European Solid-State Circuits Conference*, Villach, Austria, pp. 216-219, 2001.

[20] S. M. Sze, *Physics of Semiconductor Devices*, New York: Wiley, 1981.

[21] K. Bult, H. Wallinga, 'A Class on Analog CMOS Circuits Based on the Square-Law Characteristic of an MOS Transistor in Saturation', *IEEE Journal of Solid-State Circuits*, Vol. 22, pp. 357-365, 1987.

[22] M. Bucher, C. Lallement, C. Enz, F. Krummenacher, 'Accurate MOS Modelling for Analog Circuit Simulation Using the EKV Model', *IEEE International Symposium on Circuits and Systems,* Atlanta, USA, Vol. 4, pp. 703-706, 1996.

[23] S. Kleinfelder, S. Lim, A. El Gamal, 'A 10000 Frames/s CMOS Digital Pixel Sensor', IEEE Journal of Solid-State Circuits, Vol. 36, pp. 2049-2059, 2001.

[24] D. Yang, B. Fowler, A. El Gamal, 'A Nyquist-Rate Pixel-Level ADC for CMOS Image Sensors', *Proceedings of the IEEE Custom Integrated Circuit Conference*, Santa Clara, USA, 1998, pp. 237-240, 1998.

[25] D. Johns, K. Martin, *Analog Integrated Circuit Design*, NY: Wiley, 1997.

[26] D. Nairn, C. Salama, 'Current-Mode Algorighmic Analog-to Digital Converters', *IEEE Journal of Solid-State Circuits*, Vol. 25, pp. 997-1004, 1990.

[27] K. Chen, C. Svensson, J-R. Yuan, 'A CMOS Implementation of a Video-Rate Successive Approximation A/D converter', *Proceedings of the International Symposium on Circuits and Systems*, Espoo, Finland, pp. 2577-2580, 1988.

[28] M. Saint-Laurent, G. Muyshondt, 'A Digitally Controlled Oscillator Constructed Using Adjustable Resistors', *Proceedings of the Southwest Symposium on Mixed-Signal Design*, Austin, USA, pp. 80-82, 2001.

[29] A. Rodriguez-Vazquez et al., "Current-Mode Techniques for the Implementation of Continuous- and Discrete-Time Cellular Neural Networks", *IEEE Transactions of Circuits and Systems II*, Vol. 40, No. 3, pp. 132-146, March 1993.

[30] G. Linan, R. Dominguez-Castro, S. Espejo, A. Rodriguez-Vazquez, 'ACE16k: A Programmable Focal Plane Vision Processor with 128×128 Resolution, *Proceedings of the European Conference on Circuit Theory and Design*, Espoo, Finland, Vol. 1, pp. 345-348, 2001.

[31] P. Keresztes et al. 'An Emulated Digital CNN Implementation', *Journal of VLSI Signal Processing Systems*, Vol. 23, No. 2/3, 1999, pp. 291-303.

Chapter 6

Development of Future Array Processors

When hardware is selected for a certain processing task in a commercial product, the manufacturer chooses the best possible alternative. The underlying criteria that the decision is based on can include, for example, power consumption, speed, accuracy, size, cost, yield and long and short term reliability. Other aspects are re-usability in different designs and processes and the programmability of the device. Even if a particular hardware alternative were to yield the best combination of qualities at a certain moment, the development of technology might quickly change the situation. What makes things even more complicated is that combinations of different technologies may yield the best total performance. Totally new technologies have to be superior to existing ones in order to become commercially appealing. This is because of, for example, the risks associated with new technologies and the lack of circuit designers that have competence in the area of the new technology. Anyhow, if an emergent technology has large advantages in terms of characteristics such as computing speed, it may act as an enabling technology for a whole new range of applications. When it comes to array processors, analog realizations may have, for example, a speed advantage over their digital counterparts, whereas digital realizations are less risky. Section 6.1 highlights some aspects regarding the choice of hardware for future array processors. Section 6.2 describes how the cellular array processor described in Chapter 5 could be improved.

6.1 Choice of Hardware for Future Array Processors

So far commercial array processors have mostly been digital realizations; for example, C•RAMs have been used in graphics accelerator cards and CAMs in lookup tables. Also, chips that combine sensing and processing (NSIP type processors [1]) are commercially available. In addition to this, commercial analog/mixed-mode realizations are being pursued. Commercialization of analog CNN chips, for example, is ongoing. Since the development of technology during the next couple of years can be predicted quite well [2], some trends of building future array processors can also be formed. In the rest of this section, estimates are given as to how technological development may affect the choice of hardware in future array processors.

6.1.1 Digital Array Processors

It is obvious that digital array processors will be appealing also as technology develops. This is because digital array processors can readily utilize dense processes by packing more transistors into the chip. Also, the reduction of parasitic capacitances allows faster clock speeds. The downscaling of threshold voltages and maximum operating voltages somewhat complicates the design, but also enables large savings of power consumption for competent designers. Furthermore, as the interconnection capacity has increased, the interconnection has become easier.

Multiplication continues to require a lot of computational effort from digital hardware. Therefore, digital hardware may not be the optimal way to realize algorithms that require a lot of multiplications. One way to overcome this is to revise the algorithm so that multiplications do not need to be performed. One example of this is video coding standard H.26L. Instead of using DCT, H.26L uses a transform in which only multiplications/divisions with a power of two are allowed [3]. These can be very conveniently realized digitally by using shifts and additions.

6.1.2 Analog and Mixed-Mode Array Processors

The development of silicon processes is governed by the needs of digital designs. This is because digital circuits, microprocessors and memories, for example, occupy a large die area and are manufactured in large quantities. Analog designers that use the latest digital processes have to live with what is available.

The benefits of the diminishing minimum feature sizes for building large analog/mixed-mode computational arrays are not as obvious as the benefits for digital designs. Analog processing units cannot be significantly scaled down with the minimum feature

sizes. This is because even if the threshold voltage mismatch improves as processes advance [4], the mismatch relative to the operating voltage stays more stable [5]. Also, as highlighted in [6], the current factor mismatch does not seem to scale down with the process. Furthermore, the decreasing operating voltages make the analog design more challenging. However, the increasing number of metal layers is a significant benefit also in the design of analog/mixed-mode array computers. Also, due to the scaling down of contacts, vias and minimum distances between devices, the total area of an analog transistor scales down.

One approach is to choose not to use the latest silicon processes. This approach is suitable for analog designs as [7], for example, shows. Another approach is to use dedicated or mixed-mode designs. In this approach, simple analog components are used to carry the main computational burden and these are complemented with digital circuits. Since the analog parts are composed of simple units (for example, a combination of current mirrors), re-use even in different processes may be possible with minor modifications. An important observation is that the sizes of digital memories scale down with the technology. This makes it appealing to replace at least some analog memories with a combination of an SRAM and an A/D/A converter.

When commercial analog/mixed-mode array processors are to be build, their performance in the intended operating conditions has to be verified; for example, some temperature compensation of bias/reference voltage may be necessary in order to obtain correct operation in the whole temperature range. In the epilepsy application that was described in Chapter 3, an implanted device is targeted. Therefore, the temperature range of the environment is limited to changes of human body temperature, which is limited to a few degrees. If an acceptable yield and reliability in different environments can be obtained with an analog/mixed-mode array processor hardware, in some commercial applications the dominance of digital hardware could be threatened.

6.2 Improvements in the Mixed-Mode Cellular Array Processor

In this section, some improvements that could be applied to the cellular array processor realization of Chapter 5 are shown. Since the transition to processes of smaller minimum feature sizes is associated with the downscaling of operating voltages, it is beneficial to design analog circuits using simple circuit primitives. Section 6.2.1 describes how binary weighted current sources could be used as multipliers in a cellular array processor. Section 6.2.2 shows an example of an improved polynomial circuit

for producing quadratic and cubic terms. The simulated DC characteristics of the improved polynomial circuit follow the ideal quadratic and cubic terms more closely than the characteristics of the polynomial circuits of Chapter 5. In Section 6.2.3, it is assumed that a fast control setup is available to control the chip. This opens new possibilities for realizing the processing units. Section 6.2.3 shows a combination of top and bottom row PUs in which current memories are used to convey the border information.

6.2.1 Multiplication with Binary Programmable Current Sources

6.2.1.1 Description of Operation

Figure 6.1 shows a multiplier composed of binary weighted current sources. The input current is first mirrored using transistor M_in to binary weighted transistors $M1-M6$. Control signals $B1-B6$ control the currents, the sum of which builds up the sum current I_{sum}. Depending on the sign of the multiplication, this current is conditionally mirrored using current mirror transistors $M7$ and $M8$. The use of this kind of a multiplier has not been favored in CNN implementations so far, mainly because it requires a lot of global digital control signals to determine the weight. However, in a mixed-mode CNN, this may turn out to be the optimal multiplier because high end digital processes have six or more metal layers available for routing. The cellular array processor, for example, realized with a 6-metal $0.25\mu m$ CMOS process, utilizes the multiplier described in Section 5.2.3. Three global signals per multiplier (sign, msb, V_w) were routed to each PU and more could have easily been fitted by decreasing the aspect ratio of the PU.

The binary weighted current multiplier has some very attractive qualities. Firstly, the weights are very fast to program digitally. Also, the multiplier does not need to be biased. Consequently, no autozeroing/sampling of the bias current is needed because the output current at zero input current is inherently zero. Furthermore, since the structure is simple, it performs well with low operating voltages.

6.2.1.2 Accuracy

The standard deviation of the output current of the programmable multiplier of Figure 6.1 can be determined using Equation 5.6. Assume that input transistor M_in is composed of U_{in} identical (unit) transistors in parallel, and transistors $M1$-$M6$ are composed of $2^{n_m}-1$ unit transistors in parallel, then the number of unit transistors connected to the output is $U_{out} \in [0, 2^{n_m}-1]$. n_m is the number of magnitude

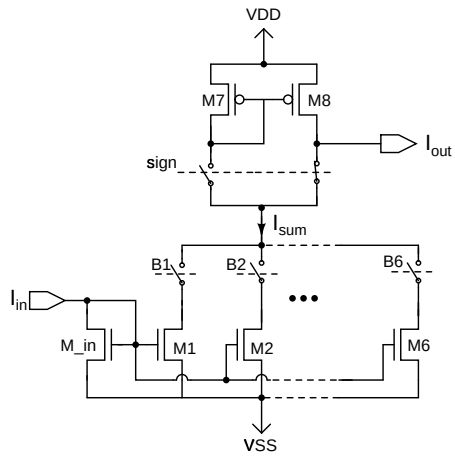

Figure 6.1 Multiplier that utilizes binary weighted current sources.

bits of the weight (n_m is six in Figure 6.1). Therefore, the input current is multiplied by gain $A = U_{out}/U_{in}$. The standard deviation at the output of the multiplier is $\sigma(I_{out}) = \sigma(I_{sum})$ for negative weights, while for positive weights it is obtained from

$$\sigma(I_{out}) = \sqrt{2\left(\sigma(I_{ds,7,8})\right)^2 + \left(\sigma(I_{sum})\right)^2}. \tag{6.1}$$

6.2.2 Improved Polynomial Circuits

Figure 6.2 shows a circuit that can be used to produce either a quadratic or a cubic term. The parallel combination of transistors $M6$ (in the linear region) and $M7$ (in saturation) converts current $I_R = I_{B1} + I_{in}$ to voltage $V_{g,1}$. The currents through transistors $M6$ and $M7$ are

$$I_{M6} = \beta_6(V_r - V_{t,6})V_{g,1} - \frac{\beta_6 V_{g,1}^2}{2} \tag{6.2}$$

and

$$I_{M7} = \frac{\beta_7}{2}(V_{g,1} - V_{t,7})^2, \tag{6.3}$$

respectively, which yield

$$I_R = (\beta_7 - \beta_6)\frac{V_{g,1}^2}{2} + \left[\beta_6\left(V_r - V_{t,6}\right) - \beta_7 V_{t,7}\right]V_{g,1} + \frac{\beta_7}{2}V_{t,7}^2. \tag{6.4}$$

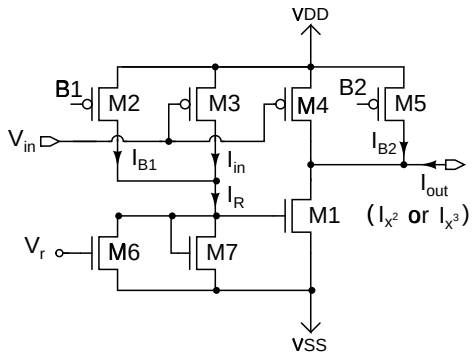

Figure 6.2 Improved circuit for generating the quadratic and cubic terms.

Voltages V_r and $V_{g,1}$ are referenced to VSS. If the current factors are chosen so that $\beta_7 = \beta_6$, the I/V converter is approximately linear. This is the scaling that is used for producing the quadratic term. The slope of the I/V converter is $R_{6,7} = \partial V_{g,1}/\partial I_R$. When $\beta_6 > \beta_7$, the slope of the I/V converter increases with I_R. This kind of scaling is used in order to obtain I/V characteristics that resemble the cubic term. The benefits of this I/V converter compared to the one-transistor I/V converters shown in the previous sections are that no current drawing common mode voltage is needed and only n-type transistors are used. Transistors $M1 - M5$ are biased to saturation. The output current I_{out} is

$$I_{out} = \frac{\beta_1}{2}\left[R_{6,7} \cdot I_R - V_{t,1}\right]^2 - \frac{\beta_4}{\beta_3} I_{in} - I_{B2}. \tag{6.5}$$

Current I_{B2} is chosen so that I_{out} at zero I_{in} is zero. Alternatively, the operating point current can be subtracted by sampling. The slope of Equation 6.5, namely

$$\frac{\partial I_{out}}{\partial I_R} = R_{6,7} \cdot \beta_1 \left[R_{6,7} \cdot I_R - V_{t,1}\right] - \frac{\beta_4}{\beta_3}, \tag{6.6}$$

should be zero when I_{in} is zero ($I_R = I_{B1}$) . This holds if

$$R_{6,7} \cdot \beta_1 \left(R_{6,7} \cdot I_{B1} - V_{t,1}\right) = \frac{\beta_4}{\beta_3}. \tag{6.7}$$

Simulation results of the quadratic and cubic terms are shown in Figure 6.3. The simulation was carried out using ELDO with level 59 parameters of a $0.18\mu m$ CMOS process. The operating voltage in the simulations was 1.8V. Bias currents (I_{B2}) in the simulations were $1.1\mu A$ for the quadratic term and $0.5\mu A$ for the cubic term. Compared to the quadratic and cubic terms shown in Figure 5.10, the curves in Figure 6.3 follow

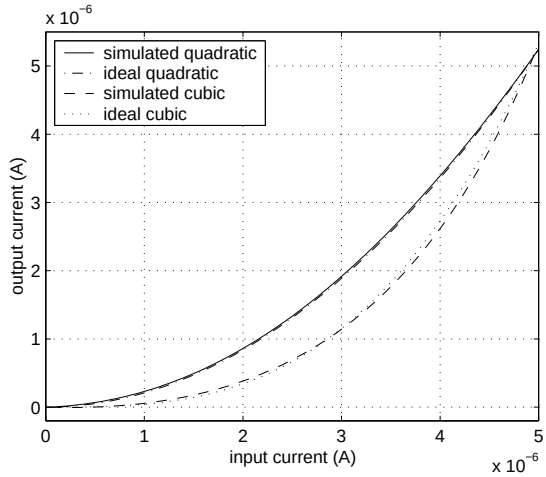

Figure 6.3 Simulated and ideal quadratic and cubic terms using circuit shown in Figure 6.2.

the ideal quadratic and cubic terms more closely.

The standard deviation of the output voltage $V_{g,1}$ of the I/V converter is determined using

$$\Delta V_{g,1} = \frac{\partial V_{g,1}}{\partial I_R}\Delta I_R + \frac{\partial V_{g,1}}{\partial V_{t,6}}\Delta V_{t,6} + \frac{\partial V_{g,1}}{\partial V_{t,7}}\Delta V_{t,7}, \tag{6.8}$$

where the current factor mismatches have been omitted. The partial derivatives of Equation 6.8 can be obtained by turning Equation 6.4 into implicit form and by differentiating the implicit function. The resulting standard deviation can be obtained from

$$\sigma(V_{g,1}) = \sqrt{\left(\frac{\sigma(I_R)}{g_{m,R}}\right)^2 + \left(\beta_6 \cdot V_{g,1}\frac{\sigma(V_{t,6})}{g_{m,R}}\right)^2 + \left(\frac{\beta_7(V_{g,1}-V_{t,7})\sigma(V_{t,7})}{g_{m,R}}\right)^2}, \tag{6.9}$$

where transconductance $g_{m,R}$ is defined as $g_{m,R} = \partial I_R/\partial V_{g,1}$. The variance of current I_R is obtained from the sum of variances of transistors $M2$ and $M3$. The standard deviations of the currents of transistors $M2$-$M5$ can be computed using Equation 5.4. Furthermore, the standard deviation of the drain-source current of transistor $M1$, namely $\sigma(I_{ds,1})$ can be computed similarly to Equation 5.24. The variance at current I_{out} is obtained by appending variances of the currents of transistors $M4$ and $M5$ to $\sigma^2(I_{ds,1})$.

Equation 6.9 shows that the standard deviation of voltage $V_{g,1}$ may cause accuracy problems. Since transistor $M1$ is biased using current I_{B1}, the size of transistor $M2$ needs to be large. It is also important that transistors $M6$ and $M7$ are sized so the their

threshold voltage variations are small. Transistor $M5$ can be made small since I_{B2} is small. If the operating point cannot be defined accurately enough using current biasing, transistors $M2$, $M6$ and $M7$ can be replaced with, for example, an I/V converter such as that of Figure 5.9 or 5.11.

6.2.3 Intra-Block Couplings Using Current Memories

If a fast control environment is available for chip measurements, the couplings between data blocks can be realized by utilizing current memories. This way, only one D/A converter is needed in a PU and the digital memories in the integrator do not need to be accessible from two layers simultaneously. Figure 6.4 shows the top and bottom row PUs of column jj when a 72×72 data is processed and the data is partitioned using Equation 4.11 with $R_y = 36$ and $R_x = 1$. The PUs are simplified in the figure so that the A/D converter is not shown and only the digital memory within the digital integrator is shown. Also, no polynomial circuits are shown in the figure. Data unit $(1, jj, 37)$ is the bottom row border condition and data unit $(2, jj, 0)$ is the top row border condition. The switches in the figure are drawn to their position when control signal *ct* is HI (and control signal $\overline{ct}$ is LO). Write signals ϕ_1 and ϕ_2 are used to sample current memories *c_mem*1 and *c_mem*2. The layer being processed is identified by $L \in [0, 36]$, while signal S is used to identify the moment at which a sample is taken with the A/D converter. The control signals as a function of time during one iteration round are shown in the bottom of Figure 6.4. The processing proceeds so that first L equals zero and control signal *ct* is HI. Next, current memories *c_mem*1 in both PUs are written using ϕ_1. Then $L = 1$, *ct* is turned LO and memories *c_mem*2 in both PUs are written using ϕ_2 and the state derivative is sampled with the aid of control signal S. After that, L is consecutively incremented by unity and *ct* is HI for even L and LO for odd L. Furthermore, *c_mem*1 is written using $\phi_1$when L is even and *c_mem*2 is written using ϕ_2 when L is odd. This way, the data available for the multipliers is what is required in Equations 4.13 and 4.14.

References

[1] J-E Eklund, C. Svensson, A. Åström, 'VLSI Implementation of a Focal Plane Image Processor - A Realization of the Near-Sensor Image Processing Concept', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 4, pp. 322-335, 1996.

[2] International Technology Roadmap for Semiconductors, 2002 update.

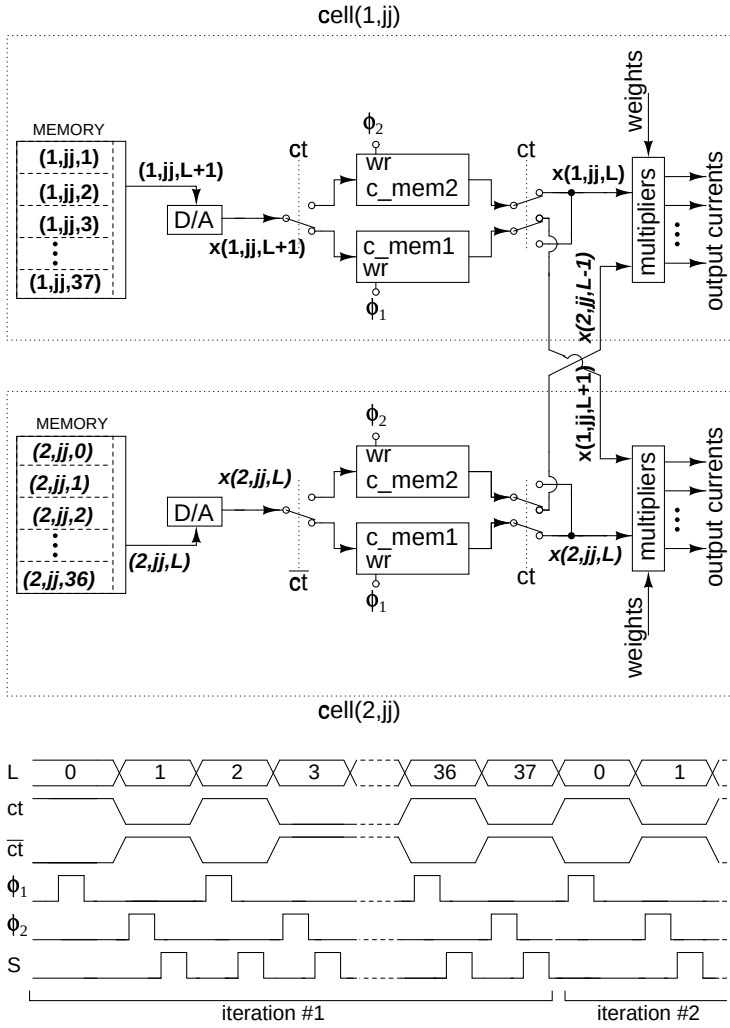

Figure 6.4 Top and bottom row PUs of column jj. Current memories are utilized in intra-block couplings.

[3] H. Malvar, A. Hallapuro, M. Karczewicz, L. Kerofsky, 'Low-Complexity Transform and Quantization with 16-Bit Arithmetic for H.26L', *Proceedings of the IEEE International Conference on Image Processing*, Rochester, USA, Vol. 2, pp. 489-492, 2002.

[4] M. Pelgrom, A. Duinmaijer, A. Welbers, "Matching Properties of MOS Transistors", *IEEE Journal of Solid-State Circuits*, Vol. 24, pp. 1433-1440, 1989.

[5] P. Kinget, M. Steyaert, *Analog VLSI Integration of Massive Parallel Processing Systems*. Dordrecht: Kluwer, 1997.

[6] A. Rodriguez-Vazquez, et al., 'Mismatch-Induced Tradeoffs and Scalability of Mixed-Signal Vision Chips', *IEEE International Symposium on Circuits and Systems*, Scottsdale, USA, Vol. 5, pp. 93-96, 2002.

[7] G. Linan, et al., 'ACE16K: an Advanced Focal-Plane Analog Programmable Array Processor', *Proc. of the* 27^{th} *European Solid-State Circuits Conference*, Villach, Austria, pp. 216-219, 2001.

Chapter 7

Conclusions

In this thesis, the realization of an array processor for computing algorithms that can be described using the theory of polynomial CNNs was described. The array processor was to meet the needs of an algorithm for analyzing brain electrical activity in epilepsy in order to predict the onset of an epileptic seizure. The work towards completion of this thesis proceeded so that first the characteristics of the epilepsy prediction algorithm were highlighted and potential realization alternatives were assessed. A mixed-mode CNN that could be used to realize a polynomial CNN was introduced. The mixed-mode CNN stores data robustly in the digital domain and uses analog arithmetic circuits for multiplication and for generating the polynomial terms. In a mixed-mode CNN, the polynomial terms can be realized using one-quadrant circuits, while four-quadrant operation can be achieved by multiplexing. Furthermore, this thesis describes how a mixed-mode CNN can be realized so that data can interact globally even if partitioned data is processed.

Based on the requirements of the epilepsy algorithm and the mixed-mode architecture, a mixed-mode cellular array processor was realized. In the realized array processor, the processing units are coupled with programmable polynomial (linear, quadratic and cubic) first-neighborhood feedback terms. The processor combines analog and digital processing so that the couplings and the polynomial terms are implemented with analog blocks, whereas the integrator is digital and A/D and D/A converters are used to interface between them. A $10mm^2$, 1.027 million transistor cellular array processor with 2×72 processing units and 36 layers of memory in each was manufactured using a 0.25μm digital CMOS process. The array processor can perform gray-scale Heun's integration of spatial convolutions with linear, quadratic and cubic activation functions for a 72×72 data while keeping all I/O operations during processing local.

One complete Heun's iteration round takes 166.4μs, while the power consumption during processing is 192mW. Experimental results of statistical variations in the multipliers and polynomial circuits were shown.

The realized mixed-mode cellular array processor shows that it is possible to realize a polynomial CNN using the mixed-mode CNN architecture. The measured processing variations can be used to assess the obtainable accuracy, and furthermore, the applicability of the mixed-mode design to the epilepsy prediction algorithm and other algorithms described using polynomial CNN theory. Descriptions regarding potential improvements in the implemented cellular array processor were described.

Being a combination of analog and digital processing units, the mixed-mode CNN is an attempt to combine the benefits of the continuously diminishing digital gates and the compactness of analog computing. The world of electronics is becoming increasingly digital. However, when it comes to devices that have to process large quantities of sensory information, in some applications there may be a demand for analog/mixed-signal array processing also in the future.

Appendix A

Chip Microphotograph

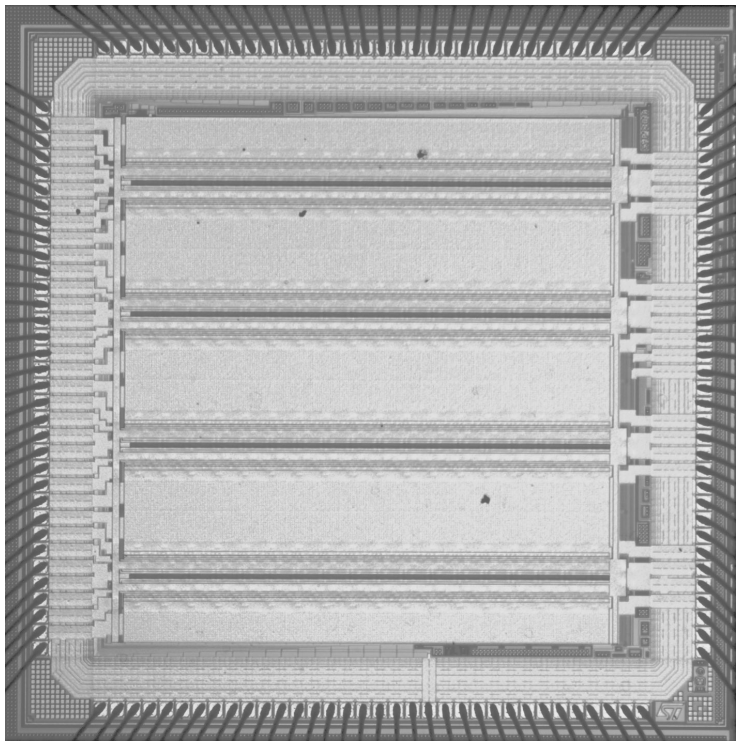

Figure A.1 Microphotograph of the chip.

This page is intentionally left blank.

Appendix B

Description of Control Signals

The following tables show the control signals of the implemented cellular array processor. The chip is bonded into a PGA-144 package. The control, bias and test signals are grouped so that Tables B.1 and B.2 are integrator control signals and signals in Table B.3 control the global I/O. Table B.4 shows the two least significant row address signals and the column activation signals. Table B.5 shows the addresses that are used to identify the layer and weight addresses, *msb* of row address and write signal of weights. Table B.6 shows the 8-bit global I/O bus. The bus can be written using 3V signals, but the output HI level of the bus is 1.8V. Table B.7 describes test pins that can be used to measure the test structures of the dummy PU and the A/D control signals. Table B.8 describes the different bias and reference voltages and currents that are needed on chip. Table B.9 shows the operating voltages and corresponding pins used in this design. Finally, Table B.10 shows typical values of the bias and reference voltages.

signal name	pin #	explanation
SAMPLE	98	ADC sample signal, rising edge starts A/D conversion
WR_1	131	write signal of REG_1, active HI
WR_2	129	write signal of REG_2, active HI
WR_Y	130	write signal of REG_Y, active HI
WR_CMB	118	write signal of REG_CMB, active HI
ADD_CT	120	controls whether REG_Y or from_A/D controls the bottom input of the adder (when HI, from_A/D is chosen)
RG2_CT	121	controls whether REG_CMB or from_A/D is fed to REG_2 (when HI, REG_CMB is chosen)
WR_Y	122	controls whether REG_1 or REG_Y controls integrator outputs rg_C and rg_B (when HI, REG_1 is chosen)

Table B.1 Integrator control signals, byte 1. Denoted by $INTEG1(8,1)$ in Figure C.1.

signal name	pin #	explanation
Z_AD	115	zero SRAMs in A/D_D
Z_PRT	116	zero SRAMs in A/D_Q
WR_D	114	enables writing of SRAM in ADC_D together with CT[6]-CT[0], active HI
WR_Q	113	enables writing of SRAM in ADC_Q together with CT[6]-CT[0], active HI
N_GATE	83	pass-gate control, HI connects voltage V_MID to data bus (precharge)
SW	132	HI opens a conditional pull-down path in the output driver of the SRAM (LO during precharge)
COL_EN	140	column decoder enable, active HI
EN_DEC_W	48	enable decoder that produces layer activation signals act[36]-act[1] or weight activation signals WR[27]-WR[1], active HI

Table B.2 Integrator control signals, byte 2. Denoted by $INTEG2(8,1)$ in Figure C.1.

signal name	pin #	explanation
RD_GBRG1	125	global read of SRAM REG_1, active HI
RD_GBRG2	128	global read of SRAM REG_2, active HI
RD_GBRGY	127	global read of SRAM REG_Y, active HI
RD_GB_OFF	117	global read of SRAM in A/D_Q, active HI
WR_GBRG1	123	global write of register REG_1, active HI
WR_GBRG2	124	global write of register REG_2, active HI
WR_GBRGY	126	global write of register REG_Y, active HI
WR_BIAS	3	write SRAM in dummy PU, active HI

Table B.3 Global I/O control signals. Denoted by $GB\_IO(8,1)$ in Figure C.1.

signal name	pin #	explanation
ROW_A1	97	row address bit (*lsb*), when row address is address 0H the highest row is activated
ROW_A2	96	row address bit
COL_A6	139	column decoder address bit (*msb*), address 00H activates the left column
COL_A5	138	column decoder address bit
COL_A4	137	column decoder address bit
COL_A3	136	column decoder address bit
COL_A2	135	column decoder address bit
COL_A1	134	column decoder address bit (*lsb*)

Table B.4 Column activation signals and two least significant row address signals. Denoted by $DEC1(8,1)$ in Figure C.1.

signal name	pin #	explanation
WR_W	55	write signal of SRAMs that are used to store the weights, active HI
ROW_A3	89	row address bit (*msb*)
AD6	49	decoder address bit (*msb*) identifies layer activation signals act[36]-act[1] and weight activation signals WR[27]-WR[1]
AD5	50	decoder address bit
AD4	51	decoder address bit
AD3	52	decoder address bit
AD2	53	decoder address bit
AD1	54	decoder address bit (*lsb*)

Table B.5 Layer and weight decoder addresses, *msb* row address and write signal of weights. Denoted by $DEC2(8,1)$ in Figure C.1.

signal name	pin #	explanation
GB_BUS8	43	global input/output bus (*msb*)
GB_BUS7	42	global input/output bus
GB_BUS6	41	global input/output bus
GB_BUS5	40	global input/output bus
GB_BUS4	39	global input/output bus
GB_BUS3	38	global input/output bus
GB_BUS2	37	global input/output bus
GB_BUS1	36	global input/output bus (*lsb*)

Table B.6 Global digital I/O bus. Output HI level is 1.8V. Denoted by $GB\_BUS(8,1)$ in Figure C.1.

signal name	pin #	explanation
WX1_SW	67	pin that allows the weight voltage corresponding to $A_{1,3}^{(1)}$ to be measured outside the chip
V_X1	143	voltage corresponding to linear term in dummy PU, connected to ground during processing
V_X2	141	voltage corresponding to quadratic term in dummy PU
V_X3	142	voltage corresponding to cubic term in dummy PU, connected to ground during processing
I_1P_OUT	144	I^+ output current of dummy PU, from a p-mirror (scaled up by two)
I_TO_1P	44	external current to I^+ input of leftmost PU in second row for testing of ADC, divided by 6 before conversion
SW1_OUT	82	ADC control signal CT[0] driven outside the chip, included in order to find out conversion time and length of pulse (pulse ratio)
V_MID	88	precharge voltage, generated on chip, can be driven also externally

Table B.7 Test voltages and currents that can be measured outside the chip.

signal name	pin #	explanation
VRESX1	74	gate bias of n-type I/V converter corresponding to linear term
VRESX2	111	gate bias of p-type I/V converter corresponding to quadratic term, negative, unprotected pad
VRESX3	75	gate bias of n-type I/V converter corresponding to cubic term
VCMX1	70	bias voltage of n-type I/V converter corresponding to linear term
VCMX2	69	bias voltage of p-type I/V converter corresponding to quadratic term
VCMX3	73	bias voltage of n-type I/V converter corresponding to cubic term
VSSA_X2	68, 112	source voltage of transistor M1 in the circuit for producing the quadratic term
BIAS_W	56	bias voltage of the I/V converter used in the generation of weights
DELAY_BIAS	81	controls delay of current starving inverter in A/D generation block, connected to a current mirror
BI_PULLD	133	pull-down bias of multiplier, connected to gate
BI_PU	119	pull-up bias of full adder and half adder in integrator, connected to gate
V_I_DAC	45	reference voltage/current of in-cell DACs, the current is divided by 2 in the converter
		NOTE: V_I_DAC is connected to VSS if WR_Q is active (HI)
		if WR_Q is LO, V_I_DAC is at bias voltage, see typical values in Table B.10
V_I_AD	46	reference voltage/current of ADCs, the current is divided by 2 in the converter
V_I_W	47	reference voltage/current of the ADC that is used to generate the weights, input current is divided by 2

Table B.8 Bias and reference voltages.

signal name	pin #	explanation
VDD	77, 78, 85, 86, 93, 94, 101, 102	3V digital operating voltage
VDD25	76, 79, 84, 87, 92, 95, 100, 103	2.5V digital operating voltage
VSS	4, 6, 9, 11, 12, 14, 17, 19, 20, 22, 25, 27, 28, 30, 33, 35, 90	digital ground
VDDA	5, 10, 13, 18, 21, 26, 29, 34	3V analog operating voltage
VSSA	7, 8, 15, 16, 23, 24, 31, 32, 91	analog ground
VDD_MEM	57	3V operating voltage of weight memory and register REG_B in dummy PU

Table B.9 Operating voltages.

signal name	voltage (and the corresponding current if applicable)
BI_PU	1.807 (fast)-1.945 (slow)
BI_PULLD	0.352
VSSAX2	0.224, the load current is up to 1.7mA
VCMX1	0.617, the load current is up to 5mA
VCMX2	0.615, the load current is up to 1.7mA
VCMX3	0.502, the load current is up to 1.7mA
VRESX1	2.347
VRESX2	-0.795
VRESX3	2.053
BIAS_W	0.265
V_I_W	0.941, corresponding current 14.2μA
V_I_AD	0.746, corresponding current 3.7μA, 0.776, corresponding current 5.2μA.
V_I_DAC	0.685, corresponding current 2,1μA, 0.713, corresponding current 2.75μA
DELAY_BIAS	0.454, corresponding ADC sampling time t_s is 3.3μs. 0.527, corresponding ADC sampling time t_s is 700ns

Table B.10 Typical bias voltages/currents.

Appendix C

Measurement Setup

The functional measurements of the chip were carried out using a PCB measurement board that was designed specifically for the implemented chip. Since the speed of the measurement board is limited, the speed and power measurements were completed using a pattern generator. In this appendix, the measurement setup that was used in the functional measurements and the speed/power measurements is explained.

C.1 Measurement Board

A PCB measurement board was built using discrete digital logic circuits, latches, buffers (74HC series) and SRAMs. The parallel port of a PC was used to control the measurement board. The port was programmed using C programming language. The PC was also used to provide the data to/from the chip and to display the data. The parallel port of a PC is composed of three ports, namely data port, control port and status port. In the measurement board the eight bits of the data port serve as a bi-directional data bus, while the control port is used to provide the board with four control signals.

Figure C.1 shows a block diagram of the measurement board. The board is composed of five SRAMs, latches, buffers and logic circuits. The bi-directional data port of the PC is connected to bus $PC\_IO(8,1)$ in the measurement board. Bi-directional data bus $GB\_BUS(8,1)$ interfaces between the measurement board and the chip. When data is transferred from PC to the chip, resistors $R1$ and $R2$ divide the 5V output voltages of buffer $B1$ so that 3V signals are conveyed to the chip through buffer $B2$. The output HI voltage of data bus $GB\_IO(8,1)$ is 1.8V when the chip drives it. The operating voltage of buffer $B3$ is 3V and the operating voltage of buffer $B1$ is 5V. Therefore,

the output HI voltage is scaled up before writing it to the PC. Latches 7 and 8 control chip inputs $DEC1(8,1)$ and $DEC2(8,1)$ while SRAMs 3-5 control chip inputs $INTEG1(8,1)$, $INTEG2(8,1)$ and $GB\_IO(8,1)$.

Ten control signals (shown on the left side of Figure C.1) are needed to control the measurement board. However, the parallel port only provides four control signals. Therefore, a logic part is also included in the measurement board. This is used to turn the four parallel port signals into ten control signals. The logic part of the measurement board is shown in Figure C.2.

C.1.1 Control of the Board

C.1.1.1 Control Signals

Table C.1 shows the bits of the control port that are used to control the measurement board. Bit5 (*ena_bidir*) is used internally in the parallel port of the PC. It determines whether the data port is in transmit or receive mode. Signals *pcr/w, D, A* and *pc_clk* are controlled by control port bits bit3, bit2, bit1 and bit0. Signals *prc/w, A* and *pc_clk* are inverted in the parallel port. For example, when HI is written to bit3, *pcr/w* is LO. Signal *pcr/w* is used to control the drivers of the bi-directional data bus on the board.

control port bit	inverted	signal name	explanation
bit5	-	*ena_bidir*	When LO, data is transferred from PC to board. When HI data is transferred from board to PC.
bit3	yes	*prc/w*	When *pcr/w* is LO, data is written from PC to board. When *pcr/w* LO, *ena_bidir* must be HI and data is transferred from board to PC.
bit2	no	*D*	mode bit, controls decoder *DEC_M* (shown in Figure C.2)
bit1	yes	*A*	mode bit, controls decoder *DEC_M* (shown in Figure C.2)
bit0	yes	*pc_clk*	clock signal of the board

Table C.1 Bits of the PC control port and corresponding signals.

C.1.1.2 Operating Modes

The measurement board has four operating modes that are selected by decoding signals *A* and *D*. These modes are explained in Table C.2. The measurement board works

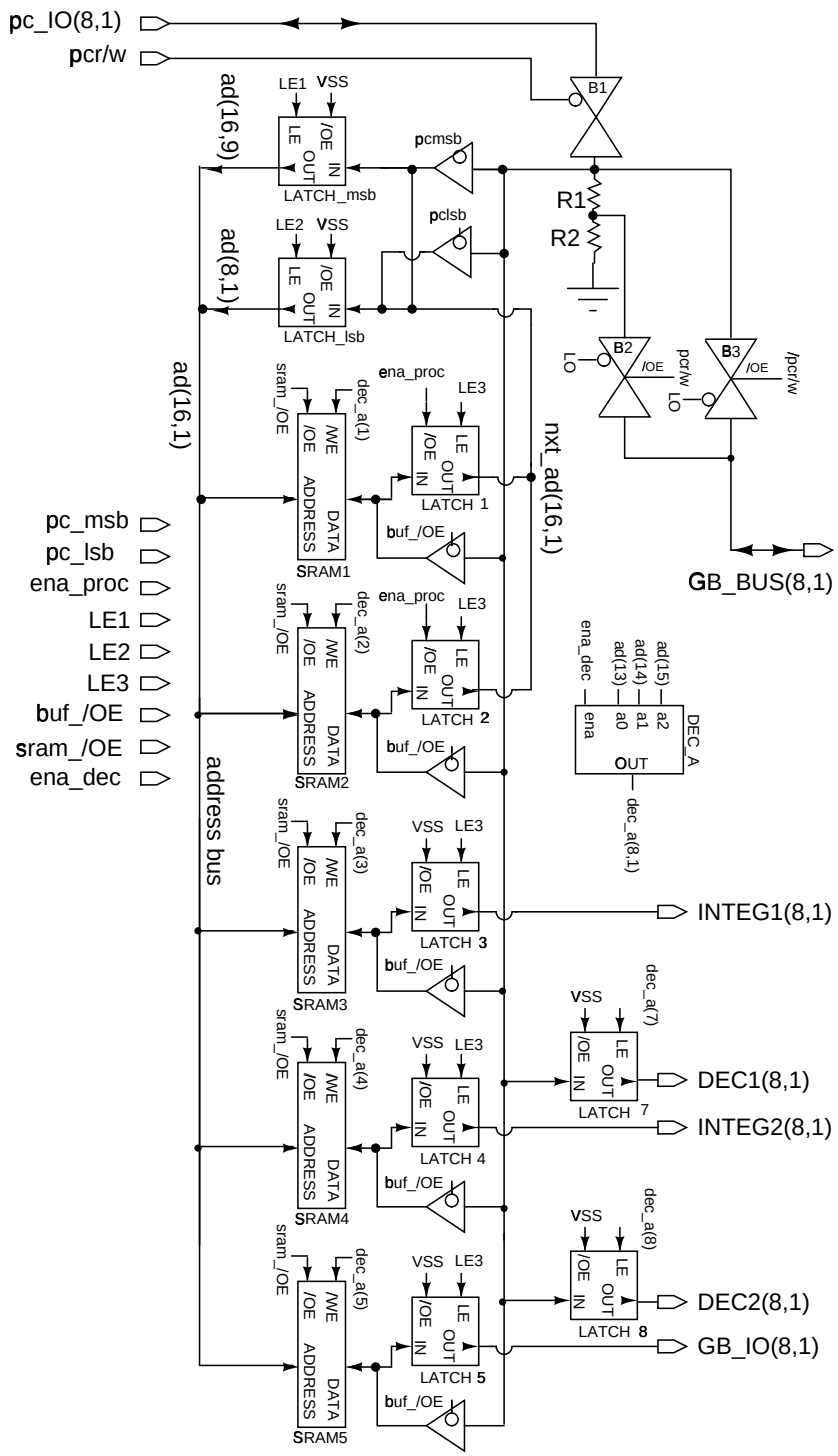

Figure C.1 Block diagram of the measurement board without control logic.

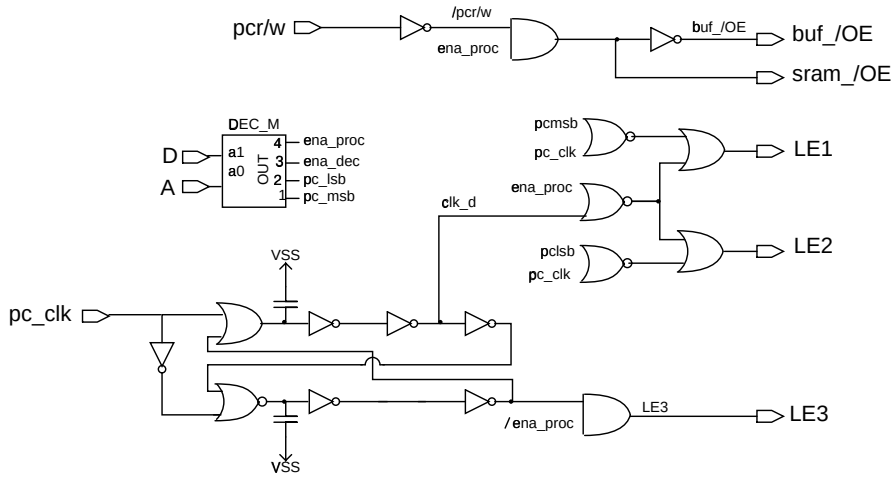

Figure C.2 Control logic of the measurement board.

so that first mode bit *pc_msb* is activated and *msb* address is written to *LATCH_msb*. Similarly, *lsb* address is written to *LATCH_lsb*. Then the operating mode is switched to *ena_dec*. In this mode, address $ad(16,1)$ is provided to the SRAMs and decoder *DEC_A*. The outputs of *DEC_A* identify which SRAM or latch is written. Table C.3 shows the addresses of the latches and SRAMs. Each SRAM has a 12-bit address space (bits $ad(12,1)$) and can therefore store 4096 eight-bit words. The SRAMs are grouped as shown in Table C.4. SRAM 1 stores the next *msb* addresses $nxt\_ad(16,9)$, SRAM 2 stores the next *lsb* addresses $nxt\_ad(8,1)$ and SRAMs 3-5 store the data of the control chip inputs $INTEG1(8,1)$, $INTEG2(8,1)$ and $GB\_IO(8,1)$. By storing these data lines to the SRAMs, the measurement board can be programmed.

After the SRAMs and latches are written, the operating mode is switched to processing mode (*ena_proc* is activated). When latch enable signal *LE3* is HI (*LE1* and *LE2* are LO), the output data of the SRAMs is stored to latches 1-5. Then *LE1, LE2* and *LE3* are toggled and the output data of latches 1 and 2 (next address $nxt\_ad(16,1)$) is stored to *LATCH_msb* and *LATCH_lsb*. Therefore, when the latch enable signals are toggled, the SRAMs are read from addresses that are programmed to SRAMs 1 and 2.

C.2 Carrying out the Measurements

Since the measurement board can store control sequences of the chip inputs $INTEG1(8,1)$, $INTEG2(8,1)$ and $GB\_IO(8,1)$, the I/O bottleneck of the parallel port is somewhat re-

D	*A*	active mode signal	explanation of the operating mode
LO	LO	*pc_msb*	PC writes the *msb* address to *LATCH_lsb*
LO	HI	*pc_lsb*	PC writes the *lsb* address to *LATCH_lsb*
HI	HI	*ena_dec*	decoder *DEC_A* is enabled, AD(15,13) are used as decoder inputs and the decoder outputs activate one of latches 7 or 8 or SRAMs
HI	LO	*ena_proc*	processing mode

Table C.2 Measurement board operating modes.

$ad(15,13)$	active SRAM or latch
0x00	SRAM1 (*msb*)
0x10	SRAM2 (*lsb*)
0x20	SRAM3
0x30	SRAM4
0x40	SRAM5
0x50	-
0x60	latch7
0x70	latch8

Table C.3 Programming the SRAMs.

laxed. Data for chip inputs $DEC1(8,1)$ and $DEC2(8,1)$ is stored in the PC because decoder input signals are easy to produce by programming with C language in the PC. The downside is that changing $DEC1(8,1)$ or $DEC2(8,1)$ takes a relatively long time. However, the speed is sufficient for functional measurements.

The speed and power measurements were carried out so that first the data to be processed was loaded into the chip, then the control of chip inputs $INTEG1(8,1)$, $INTEG2(8,1)$ and $DEC2(8,1)$ was given for a pattern generator. The pattern generator provided the fast switching signals for the chip and the integration sequence was completed. Then the control of chip inputs $INTEG1(8,1)$, $INTEG2(8,1)$ and $DEC2(8,1)$ was given back for the measurement board, and data read from the chip to the PC.

SRAM #	1	2	3	4	5
data line	$nxt\_ad(16,9)$	$nxt\_ad(8,1)$	$INTEG1$	$INTEG2$	GB_IO

Table C.4 Contents of one control line in a program.