

List of Figures

Figure 1-1: Table of direct influences of previous work, the technological and theoretical basis for the presented memory communication concept.	4
Figure 2-1: The array structure and nearest neighbor routing facilities of the Algotronix CAL1024 FPGA [Kea89] [Alg91a] [Alg91b] [OD95].	6
Figure 2-2: Overview on the architecture of the DP-FPGA.	7
Figure 2-3: The KressArray-1 architecture.	8
Figure 2-4: Basic cell of RaPiD-1.	9
Figure 2-5: The MATRIX architecture: (a) network switch, (b) nearest neighbor interconnect, and (c) length four bypass interconnect.	11
Figure 2-6: MATRIX basic function unit (BFU).	12
Figure 2-7: The Raw processor: (a) consists of an array of identical tiles. (b) Each tile contains instruction memory (IMEM), data memories (DMEM), an ALU, registers, configurable logic (CL), and a programmable switch with its associated instruction memory (SMEM).	13
Figure 2-8: The Pleiades heterogeneous architecture.	14
Figure 2-9: KressArray communication architecture by examples: (a) 4 reconfigurable nearest neighbor ports (rNN ports), (b) 8 rNN ports, (c) 10 rNN ports, (d) reconfigurable Data Path Unit (rDPU), use for routing only; (e) rDPU use for function and routing, (f) 2 back buses per row, (g) segmented single buses per column, (h) 2 per column, 3 per row, and (i) different function sets in alternating columns. Some vertical torus structure examples: (j) no shift vertical; (k) vertical shift right (to next column), (l) shift left (to previous column), and (m) double torus.	16
Figure 2-10: The MorphoSys architecture.	17
Figure 2-11: MorphoSys interconnection layers in the reconfigurable cell array.	18

Figure 2-12: MorphoSys reconfigurable cell (RC) architecture.	19
Figure 2-13: The CHESS array: (a) floorplan with embedded RAMs, and (b) detailed floorplan with nearest neighbor wiring.	20
Figure 2-14: The DReAM hardware structure: (a) architectural overview, and (b) the reconfigurable processing unit (RPU).	21
Figure 3-1: Processor-memory performance gap (taken from [PAC97]).	28
Figure 3-2: Different possibilities to attach data memory to a reconfigurable architecture: (a) external memory, (b) internal memory block, (c) reconfigurable architecture surrounded by internal memory, and (d) embedded memory.	30
Figure 3-3: Altera FLEX 10k CPLD family device block diagram with embedded array blocks (EAB).	32
Figure 3-4: The Actel ES FPGA: (a) overview on a 4-tile device, and (b) B8x8 utilities with embedded SRAM.	33
Figure 3-5: Xilinx Virtex series [Xil99]: (a) architecture overview, and (b) BlockRAM size of different devices.	34
Figure 4-1: A simple instruction sequencer (software-driven).	37
Figure 4-2: Overview on the Structured Memory Access (SMA) machine.	38
Figure 4-3: Memory Access Processor (MAP) of the Structured Memory Access (SMA) Machine.	39
Figure 4-4: The Move Control Unit (MCU) of the Map-oriented Machine 1	42
Figure 4-5: Generic address generation architectures.	44
Figure 4-6: The data sequencer of the Map-oriented Machine 2.	45
Figure 4-7: The JumpGenerator of the Map-oriented Machine 2.	45
Figure 4-8: Map-oriented Machine 2 Stepper Architecture.	46
Figure 4-9: The Single Step Control Unit of the Map-oriented Machine 2 for scan window generation.	47
Figure 4-10: The video signal processor (VSP) as a basis for the utilization of the address generation unit (AGU).	48
Figure 4-11: VSP address generation unit (AGU) addressing modes.	49

Figure 4-12: VSP address generation unit (AGU) sub modes of neighborhood search.	49
Figure 4-13: The VSP address generation unit (AGU) architecture.	50
Figure 4-14: The Data Sequencer of the Map-oriented Machine 3.	52
Figure 4-15: Block diagram and operation illustration of the MoM-3 generic address generator.	52
Figure 4-16: Handle position generator of the MoM-3 generic address generator. ...	53
Figure 4-17: Memory address generator of the MoM-3 generic address generator. .	54
Figure 4-18: The Texas Instruments TMS320C54x DSP architectural overview.	58
Figure 4-19: TMS320C54x direct addressing block diagram.	60
Figure 4-20: TMS320C54x indirect addressing block diagram for dual data-memory operands.	61
Figure 4-21: Adopt target architecture styles for custom processors: (a) example application with index function, (b) custom address calculation unit (cACU), and (c) incremental address generation unit (iAGU).	63
Figure 4-22: Intersil HSP45240 address sequencer block diagram.	66
Figure 4-23: Intersil HSP45240 sequence generator block.	68
Figure 5-1: The inherent advantage of reconfigurable systems in connection with the memory interface load: the time of instruction fetch (graphic taken from [Har00]).	73
Figure 5-2: Classification of configurable computers by data sequencing mechanism.	75
Figure 5-3: The PRISM-II hardware platform.	76
Figure 5-4: The Riley-2 architecture.	77
Figure 5-5: EPLD based transient recorder for video signals by L. Larson: (a) overview, and (b) controller EPLD.	78
Figure 5-6: RaPiD local memory with address generator.	79
Figure 5-7: The PAR-1: (a) machine overview, (b) sequencing and interface module (SIM), and (c) datapath module (DM) including local memory.	80
Figure 5-8: The REACT hardware architecture.	81
Figure 5-9: The CHAMP timing and control unit.	82

Figure 5-10: Block diagram of the Map-oriented Machine 1.	83
Figure 5-11: Block diagram of the Map-oriented Machine 2.	84
Figure 5-12: The MoM-3 machine architecture.	85
Figure 5-13: Different scan window sizes and an address sequence for a linear scan pattern	85
Figure 5-14: MoM-3 Scan Pattern Examples:	
(a) Single Steps,	
(b) Linear Scans,	
(c) Video Scans,	
(d) Zig-zag Scan,	
(e) Spiral Scan,	
(f) Curve Following (Data Dependent), and	
(g) Lee Routing (Data Dependent).	86
Figure 6-1: The basic Xputer architecture.	90
Figure 6-2: The 2-stage address generation.	92
Figure 6-3: Basic data sequencer architecture consisting of a two stage address generator pipeline.	94
Figure 6-4: The scan window generator implementation.	96
Figure 6-5: A video scan example:	
(a) scan steps	
(b) (c) with x- and y-components.	98
Figure 6-6: Illustration of the slider model [HBH97a] and generated scan sequence for x-component of the video scan in figure 6-5 at page 98.	100
Figure 6-7: Pseudo code description of the slider model for $Floor < B0 < L0 < Ceiling$	101
Figure 6-8: Video scan example with possible slider model parameters:	
(a) all parameters are handle positions,	
(b) Limit is no handle position,	
(c) Floor is no handle position, and	
(d) Ceiling is no handle position.	103
Figure 6-9: Stepper used in the Handle Position Generator to generate one slider (see figure 6-6 at page 100) of the slider model for one coordinate (x or y).	110
Figure 6-10: The escape unit. Different escape conditions:	
(a) $End > Initial\ position$,	
(b) $End < Initial\ position$, and	
(c) hardware implementation of the escape unit.	111
Figure 6-11: 1-dimensional generic address generator (1d-GAG).	112

Figure 6-12: The 2-dimensional video scan generator with three steppers per dimension, which implement the sliders of the slider model (figure 6-6 at page 100).	114
Figure 6-13: Window stack with video scan parameters and stack window movement illustrated.	115
Figure 6-14: Run length coding scan for a 24x16 pixel image in JPEG image compression method.	117
Figure 6-15: Inner scan of the JPEG zig-zag enumeration: (a) scan steps, and (b) generated by two meshed scans.	117
Figure 6-16: Meshed scan 1 of the JPEG zig-zag enumeration (see figure e6-15 at page 117): (a) scan steps of video scan 1, and (b) the covered area in the data memory, and (c) scan steps of video scan 2, and (d) the covered area in the data memory.	118
Figure 6-17: Meshed scan 2 of the JPEG zig-zag enumeration (see figure e6-15 at page 117): (a) scan steps of video scan 1, and (b) the covered area in the data memory, and (c) scan steps of video scan 2, and (d) the covered area in the data memory.	119
Figure 6-18: Stack based implementation of the JPEG zig-zag enumeration.	120
Figure 6-19: Memory unit with integrated scan window generator.	125
Figure 6-20: Memory unit without scan window generator.	126
Figure 6-21: The KressArray implementation of a stepper (see also figure e6-9 at page 110 and figure 6-10 at page 111).	127
Figure 6-22: The basic KressArray implementation of the data sequencer. For details on the stepper implementation see figure 6-21 at page 127.	129
Figure 6-23: KressArray mappings generated by the MA-DPSS (see [HHH99a] or [HHH00]) of the data sequencer shown in figure 6-22 at page 129: (a) using an array with one directed nearest neighbor (NN) port, and (b) using an array with one bidirectional nearest neighbor (NN) port.	131
Figure 6-24: KressArray mappings generated by the MA-DPSS (see [HHH99a] or [HHH00]) of the data sequencer shown in figure 6-22 at page 129: (a) using an array with two bidirectional nearest neighbor (NN) ports, and (b) using an array with three nearest neighbor (NN) ports.	132

Figure 6-25: Routing requirements needed to map the data sequencer shown in figure 6-22 at page 129 to KressArrays with different fabrics. 133

Figure 6-26: Handle position sequences of linear scan examples:
 (a) parallel to the y-axis,
 (b) diagonal, and
 (c) parallel to the x-axis. 134

Figure 6-27: Handle position sequences of rectangular video scan examples:
 (a) scan line parallel to the y-axis, and
 (b) scan line parallel to the x-axis. 135

Figure 6-28: Application specific data sequencers:
 (a) to generate a linear scan parallel to the y-axis as pictured at figure 6-26a at page 134, and
 (b) to generate a diagonal scan as shown at figure 6-26b at page 134. 136

Figure 6-29: Data sequencer for a rectangular video scan which is parallel to the x- and y-axis (see figure 6-27a at page 135). The operators numbers are used in the layout in figure 6-21 at page 114. 137

Figure 6-30: Handle position sequence of a triangular video scan example. 138

Figure 6-31: Handle position sequence of a parallelogram video scan example. ... 139

Figure 6-32: Handle position sequence of a trapezium video scan example. 139

Figure 6-33: Application specific data sequencer to generate triangle (see figure 6-30 at page 138), trapezium (see figure 6-32 at page 139), and parallelogram (see figure 6-31 at page 139) video scans, which are parallel to the y-axis. 140

Figure 6-34: Area requirements of the presented application specific data sequencers. 141

Figure 7-1: Memories of different dimension:
 (a) 1-dimensional, and
 (b) 2-dimensional. 147

Figure 7-2: Row major mapping (see [HB85] or [PH90]) of 2-dimensional memory to 1-dimensional memory. 148

Figure 7-3: Row major mapping with parallel memory banks. 150

Figure 7-4: Dynamic assignment of scan window positions to memory banks during scan window movement:
 (a) before scan window movement, and
 (b) new assignment after scan window movement. 151

Figure 7-5: Illustration of row major address mapping for 2-dimensional array coherence:	
(a) memory mapping of a 2-dimensional example	
(b) to one linear memory	
(d) without re-mapping by the memory mapper, and	
(c) to one linear memory	
(e) with re-mapping by the memory mapper.	152
Figure 7-6: Illustration of the speed-up figures for the concurrent data access examples shown in table 7-1.	154
Figure 7-7: (a) scan pattern example,	
(b) (c) (d) scan steps with scan window overlapping, and	
(e) scan step without scan window overlapping.	155
Figure 7-8: Illustration of the speed-up figures for the scan window overlap optimization shown in table 7-2 at page 156.	157
Figure 7-9: Burst accesses inside a scan window.	158
Figure 7-10: Bank interleave addressing scheme of the Map-oriented Machine with Parallel Data Access (MoM-PDA, see appendix D.1 at page 262). ...	159
Figure 7-11: Illustration of the speed-up figures for the burst access optimization shown in table 7-3 at page 159.	160
Figure 7-12: Summary of hardware level access optimizations.	161
Figure 7-13: Scan pattern example:	
(a) before inner scan line loop unrolling, and	
(b) after inner scan line loop unrolling with unrolling factor 3.	163
Figure 7-14: Scan pattern example:	
(a) before scan line unrolling, and	
(b) after scan line unrolling with unrolling factor 2.	163
Figure 7-15: Scan window example with	
(a) burst accesses inside,	
(b) burst accesses after loop unrolling, and	
(c) burst accesses after loop unrolling and modification of the storage scheme.	164
Figure 7-16: Number of memory cycles of the loop unrolling application example in figure 7-15 at page 164.	165
Figure 7-17: Low level storage map modification example:	
(a) scan window before modification,	
(b) scan window after low level storage map modification, and	
(c) example scan pattern.	166
Figure 7-18: Number of memory cycles of the storage scheme optimization example in figure 7-17 at page 166.	167

Figure 7-19: High level storage scheme modification: exchange of x and y: (a) before modification, and (b) after modification.	169
Figure 7-20: Number of memory cycles of one scan window access for the application example in figure 7-19 at page 169.	170
Figure 7-21: Data map compression example: (a) before modification with wasted data memory, and (b) after compression (c) with transformation matrix A.	171
Figure 7-22: Application example: (a) before data map transformation, and (b) after modification.	172
Figure 7-23: Required resources of the transformed example application of figure 7-22b at page 172 in percent of the original application (figure 7-22a at page 172).	174
Figure 7-24: Algorithm to find the optimum burst accesses.	176
Figure 7-25: Algorithm to find the Enhanced Scheduling.	178
Figure 7-26: Supplemented hardware architecture, which supports optimum scheduling of data.	179
Figure 7-27: The optimization flow.	180
Figure 8-1: The linear filter application: (a) calculation of a new pixel value under application of a linear filter (two scan windows), (b) the result may be stored also at the upper left location, (c) then the result can also be written into the memory space of the original image without overwriting data needed for the following computation steps (one scan window).	185
Figure 8-2: Mapping of the merged buffer linear filter implementation to the 2- dimensional memory: (a) scan pattern, and (b) initial scan window design.	186
Figure 8-3: Scan window input with the XMDS Task Designer.	187
Figure 8-4: Scan pattern input with the XMDS Task Designer.	187
Figure 8-5: Hardware level scan window optimizations for the merged buffer linear filter: (a) parallel data access only, (b) burst access only, and (c) scan window overlap only.	190

-
- Figure 8-6: Comparison of the different scan window types for the merged buffer linear filter application optimized on hardware level: initial scan window (see section 8.1.3 at page 189), with parallel memory bank optimization only (see page 189), with burst access optimization only (see page 190), with scan window overlapping only (see page 190), and under application of all hardware level access optimizations (see page 191): the first scan line scan window and the inner scan line scan window.192
- Figure 8-7: Illustration of scan line unrolling:
(a) scan windows at the same position of three subsequent scan lines,
(b) scan line parallelization of the merged buffer linear filter application, generated from the merged buffer linear filter application (see figure 8-2 at page 186) by scan line unrolling, and
(c) its compound scan window.194
- Figure 8-8: Illustration of inner scan line loop unrolling:
(a) scan windows of two subsequent positions in a scan line,
(b) the parallelized merged buffer linear filter application, generated from the step parallelization of the merged buffer linear filter application (see figure 8-7 at page 194) by inner scan line loop unrolling, and
(c) its compound scan window.196
- Figure 8-9: Mapping of the parallelized merged buffer linear filter application:
(a) after modification of the storage map by the exchange of x and y, and
(b) its detailed scan window.198
- Figure 8-10: Application of the transformation matrix with the XMDS Task Designer. 199
- Figure 8-11: Scan window implementation of the parallelized merged buffer linear filter application after modification of the storage map and the access optimization process:
(a) for the first scan line position, and
(b) for inner scan line positions: read bursts of overlapping scan window area omitted.200
- Figure 8-12: Number of memory cycles of the merged buffer linear filter application for different levels of memory access optimization.202
- Figure 8-13: Speed-up figures of the merged buffer linear filter application for different levels of memory access optimization. The figures are based on the memory cycles given in figure 8-12 at page 202.203
- Figure 8-14: Application Statistics of the initial linear filter implementation generated by the XMDS Application Analyzer.204
- Figure 8-15: Application Statistics of the final (optimized) linear filter design generated by the XMDS Application Analyzer.205

Figure 8-16: Required clock cycles of the initial and final (optimized) filter implementation.	206
Figure 8-17: The parallelized linear filter implementation used for the KressArray (a) input image (b) with scan window, and (c) filtered result picture (d) with scan window.	208
Figure 8-18: Linear filter implementation for the KressArray-3 using DPUs of conventional functionality. Cursive numbers indicate the operator and memory port numbers in the layout, pictured in figure e8-21 at page 214. 210	
Figure 8-19: Linear filter implementation to compute p0new. Cursive numbers indicate the operator numbers in the layouts, pictured in figure 8-21 at page 214 and figure 8-23 at page 216.	211
Figure 8-20: Linear filter implementation to compute p3new. Cursive numbers indicate the operator numbers in the layouts, pictured in figure 8-21 at page 214 and figure 8-23 at page 216.	212
Figure 8-21: KressArray-3 mapping result for the linear filter implementation using DPUs of conventional functionality (see figure 8-18 at page 210). ...	214
Figure 8-22: Linear filter implementation for the KressArray-3 with multi-function DPUs. Cursive numbers indicate the operator and memory port numbers in the layout, pictured in figure 8-23 at page 216.	215
Figure 8-23: KressArray-3 mapping result for the linear filter implementation using multi-functional DPUs (see figure 8-22 at page 215).	216
Figure 8-24: The linear filter mapping results.	218
Figure A-1: Overview of the XMDS.	226
Figure A-2: The XMDS Chooser window.	230
Figure A-3: The XMDS Task Designer window for scan window design.	232
Figure A-4: The XMDS Task Designer window for video scan design.	233
Figure A-5: The XMDS Task Designer window for complex scan design.	234
Figure A-6: The XMDS Application Composer window.	235
Figure A-7: The XMDS High Level Language Interface window.	236
Figure A-8: The XMDS Application Analyzer window.	237
Figure A-9: The XMDS Memory Access Simulator window.	238
Figure A-10: The XMDS KressArray Configuration Generator window.	239
Figure C-1: MDRAM block diagram (reprinted from [Sie96]).	254

Figure C-2: Timing of ID register reprogramming command sequence.	256
Figure C-3: Address and data paths for read/write operations in the MDRAM.	257
Figure C-4: Sample timing of a burst operation.	258
Figure D-1: The MoM-PDA machine overview.	262
Figure D-2: The address generation datapath of the MoM-PDA data sequencer. ..	264
Figure D-3: The 1-dimensional Generic Address Generator of the MoM-PDA data sequencer.	265
Figure D-4: The MoM-PDA Base / Limit stepper implementation.	267
Figure D-5: The end detection unit of the MoM-PDA Base- and Limit steppers. .	268
Figure D-6: The Address stepper of the MoM-PDA data sequencer.	270
Figure D-7: The Scan Window Generator of the MoM-PDA data sequencer.	271
Figure D-8: The organization of the Offset Memory of the MoM-PDA Scan Window Generator.	272
Figure D-9: The offset Generator of the MoM-PDA Scan Window Generator.	273
Figure D-10: Data of several applications mapped to the parallel memory banks. ..	274
Figure D-11: Context switcher implementation of the MoM-PDA data sequencer.	275
Figure D-12: The memory mapper of the MoM-PDA data sequencer.	276
Figure D-13: Schematic of the Burst Control Unit / MDRAM bus system of the MoM- PDA.	278
Figure D-14: Structure of the MoM-PDA Burst Control Unit.	279
Figure D-15: The task manager of the MoM-PDA data sequencer.	281
Figure D-16: XC6216 PCI Board from Virtual Computer Corporation.	283
Figure D-17: Hot Works Board Architecture.	284
Figure D-18: MoM-PDA prototype board.	285
Figure D-19: Altera FLEX 10k100 device with adaptor board.	286
Figure D-20: SIEMENS MDRAM device.	287
Figure D-21: Floor plan of the Burst Control Unit on the Xilinx XC4013e chip.	288
Figure D-22: Xilinx XC6216 FPGA: (a) device, and (b) die photograph [MV97].	289
Figure D-23: Timing of an rALU-interrupted read burst.	289

List of Figures

Figure D-24:Timing of an rALU-interrupted write burst.	290
Figure D-25:The MoM-PDA rALU organization.	291
Figure D-26:MDRAM interface diagram.	292
Figure D-27:The KressArray emulator.	293