

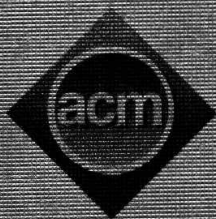
Lecture Notes in Computer Science

1685

P. Amestoy P. Berger M. Daydé I. Duff
V. Frayssé L. Giraud D. Ruiz (Eds.)

Euro-Par'99 Parallel Processing

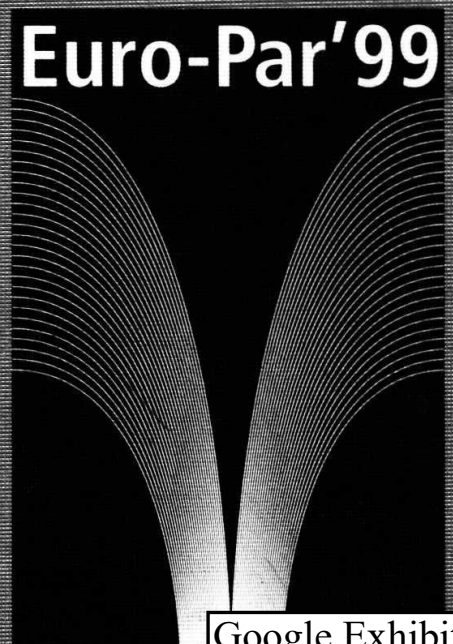
5th International Euro-Par Conference
Toulouse, France, August/September 1999
Proceedings



Euro-Par'99



Springer



Google Exhibit 1009
Google v. Singular

Patrick Amestoy Philippe Berger
Michel Daydé Iain Duff Valérie Frayssé
Luc Giraud Daniel Ruiz (Eds.)

Euro-Par'99

Parallel Processing

5th International Euro-Par Conference
Toulouse, France, August 31 – September 3, 1999
Proceedings



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Patrick Amestoy
Philippe Berger
Michel Daydé
Daniel Ruiz
ENSEEIH, 2, Rue Camichel, F-31071 Toulouse Cedex 7, France
E-mail: {amestoy,berger,dayde,ruiz}@enseeih.fr

Iain Duff
Valérie Frayssé
Luc Giraud
CERFACS, 42, Av. Gaspard Coriolis, F-31057 Toulouse Cedex 1, France
E-mail: {duff,fraysse,giraud}@cerfacs.fr

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Parallel processing : proceedings / Euro-Par '99, 5th International Euro-Par Conference, Toulouse, France, August 31 - September 3, 1999. Patrick Amestoy ... (ed.). [ACM ; IFIR]. - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 1999
(Lecture notes in computer science ; Vol. 1685)
ISBN 3-540-66443-2

CR Subject Classification (1998): C.1-4, D.1-4, F.1-2, G.1-2, E.1, H.2

ISSN 0302-9743

ISBN 3-540-66443-2 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1999
Printed in Germany

Typesetting: Camera-ready by author
SPIN: 10704397 06/3142 - 5 4 3 2 1 0 Printed on acid-free paper

Table of Contents

Invited Talks

Some Parallel Algorithms for Integer Factorisation	1
<i>Richard P. Brent</i>	
MERCATOR, the Mission	23
<i>Philippe Courtier</i>	
Adaptive Scheduling for Task Farming with Grid Middleware	30
<i>Henri Casanova, MyungHo Kim, James S. Plank, Jack J. Dongarra</i>	
Applying Human Factors to the Design of Performance Tools	44
<i>Cherri M. Pancake</i>	
Building the Teraflops/Petabytes Production Supercomputing Center	61
<i>Horst D. Simon, William T.C. Kramer, Robert F. Lucas</i>	
A Coming of Age for Beowulf-Class Computing	78
<i>Thomas Sterling, Daniel Savarese</i>	

Topic 01

Support Tools and Environments	89
<i>Frédéric Desprez</i>	
Systematic Debugging of Parallel Programs in DIWIDE Based on Collective Breakpoints and Macrosteps	90
<i>P. Kacsuk, R. Lovas, J. Kovács</i>	
Project Workspaces for Parallel Computing - The TRAPPER Approach ..	98
<i>Dino Ahr, Andreas Bäcker</i>	
PVMbuilder - A Tool for Parallel Programming	108
<i>Jan B. Pedersen, Alan Wagner</i>	
Message-Passing Specification in a CORBA Environment	113
<i>T. Es-sqalli, E. Fleury, E. Dillon, J. Guyard</i>	
Using Preemptive Thread Migration to Load-Balance Data-Parallel Applications	117
<i>Gabriel Antoniu, Christian Perez</i>	
FITS—A Light-Weight Integrated Programming Environment	125
<i>B. Chapman, F. Bodin, L. Hill, J. Merlin, G. Viland, F. Wollenweber</i>	

INTERLACE: An Interoperation and Linking Architecture for Computational Engines	135
<i>Matthew J. Sottile, Allen D. Malony</i>	
Multi-protocol Communications and High Speed Networks	139
<i>Benoît Planquelle, Jean-François Méhaut, Nathalie Revol</i>	
An Online Algorithm for Dimension-Bound Analysis	144
<i>Paul A.S. Ward</i>	
Correction of Monitor Intrusion for Testing Nondeterministic MPI-Programs.....	154
<i>D. Kranzlmüller, J. Chassin de Kergommeaux, Ch. Schaubschläger</i>	
Improving the Performance of Distributed Shared Memory Environments on Grid Multiprocessors	159
<i>Dimitris Dimitrellos, Constantine Halatsis</i>	
Topic 02	
Performance Evaluation and Prediction.....	163
<i>Jean-Marc Vincent</i>	
Performance Analysis of Wormhole Switching with Adaptive Routing in a Two-Dimensional Torus.....	165
<i>M. Colajanni, B. Ciciani, F. Quaglia</i>	
Message Passing Evaluation and Analysis on Cray T3E and SGI Origin 2000 Systems	173
<i>M. Prieto, D. Espadas, I.M. Llorente, F. Tirado</i>	
Performance Evaluation and Modeling of the Fujitsu AP3000 Message-Passing Libraries	183
<i>Juan Touriño, Ramón Doallo</i>	
Improving Communication Support for Parallel Applications	188
<i>Joerg Cordsen, Marco Dimas Gubitoso</i>	
A Performance Estimator for Parallel Programs.....	193
<i>Jeff Reeve</i>	
Min-Cut Methods for Mapping Dataflow Graphs	203
<i>Volker Elling, Karsten Schwan</i>	
Influence of Variable Time Operations in Static Instruction Scheduling....	213
<i>Patricia Borensztein, Cristina Barrado, Jesus Labarta</i>	
Evaluation of LH*LH for a Multicomputer Architecture*	217
<i>Andy D. Pimentel, Louis O. Hertzberger</i>	

Set Associative Cache Behavior Optimization	229
<i>Ramón Doallo, Basilio B. Fraguera, Emilio L. Zapata</i>	
A Performance Study of Modern Web Server Applications	239
<i>Ramesh Radhakrishnan, Lizy Kurian John</i>	
An Evaluation of High Performance Fortran Compilers Using the HPFBench Benchmark Suite	248
<i>Guohua Jin, Y. Charlie Hu</i>	
Performance Evaluation of Object Oriented Middleware	258
<i>László Böszörményi, Andreas Wickner, Harald Wolf</i>	
PopSPY: A PowerPC Instrumentation Tool for Multiprocessor Simulation .	262
<i>C. Limousin, A. Vartanian, J-L. Béchenec</i>	
Performance Evaluation and Benchmarking of Native Signal Processing ...	266
<i>Deependra Talla, Lizy Kurian John</i>	
Topic 03	
Scheduling and Load Balancing	271
<i>Jean-Marc Geib, Bruce Hendrickson, Pierre Manneback, Jean Roman</i>	
A Polynomial-Time Branching Procedure for the Multiprocessor Scheduling Problem	272
<i>Ricardo C. Corrêa, Afonso Ferreira</i>	
Optimal and Alternating-Direction Load Balancing Schemes	280
<i>Robert Elsässer, Andreas Frommer, Burkhard Monien, Robert Preis</i>	
Process Mapping Given by Processor and Network Dynamic Load Prediction	291
<i>Jean-Marie Garcia, David Gauchard, Thierry Monteil, Olivier Brun</i>	
Ordering Unsymmetric Matrices into Bordered Block Diagonal Form for Parallel Processing	295
<i>Y.F. Hu, K.C.F. Maguire, R.J. Blake</i>	
Dynamic Load Balancing for Ocean Circulation Model with Adaptive Meshing	303
<i>Eric Blayo, Laurent Debreu, Grégory Mounié, Denis Trystram</i>	
DRAMA: A Library for Parallel Dynamic Load Balancing of Finite Element Applications	313
<i>Bart Maerten, Dirk Roose, Achim Basermann, Jochen Fingberg, Guy Lonsdale</i>	
Job Scheduling in a Multi-layer Vision System	317
<i>M. Fikret Ercan, Ceyda Oğuz, Yu-Fai Fung</i>	

A New Algorithm for Multi-objective Graph Partitioning	322
<i>Kirk Schloegel, George Karypis, Vipin Kumar</i>	
Scheduling Iterative Programs onto LogP Machine	332
<i>Welf Löwe, Wolf Zimmermann</i>	
Scheduling Arbitrary Task Graphs on LogP Machines	340
<i>Cristina Boeres, Aline Nascimento, Vinod E.F. Rebello</i>	
Scheduling with Communication Delays and On-Line Disturbances	350
<i>Aziz Moukrim, Eric Sanlaville, Frédéric Guinand</i>	
Scheduling User-Level Threads on Distributed Shared-Memory Multiprocessors	358
<i>Eleftherios D. Polychronopoulos, Theodore S. Papatheodorou</i>	
Using Duplication for the Multiprocessor Scheduling Problem with Hierarchical Communications	369
<i>Evrpidis Bampis, Rodolphe Giroudeau, Jean-Claude König</i>	
Topic 04	
Compilers for High Performance Systems	373
<i>Barbara Chapman</i>	
Storage Mapping Optimization for Parallel Programs	375
<i>Albert Cohen, Vincent Lefebvre</i>	
Array SSA for Explicitly Parallel Programs	383
<i>Jean-François Collard</i>	
Parallel Data-Flow Analysis of Explicitly Parallel Programs	391
<i>Jens Knoop</i>	
Localization of Data Transfer in Processor Arrays	401
<i>Dirk Fimmel, Renate Merker</i>	
Scheduling Structured Systems	409
<i>Jason B. Crop, Doran K. Wilde</i>	
Compiling Data Parallel Tasks for Coordinated Execution	413
<i>Erwin Laure, Matthew Haines, Piyush Mehrotra, Hans Zima</i>	
Flexible Data Distribution in PGHPF	418
<i>Mark Leair, Douglas Miles, Vincent Schuster, Michael Wolfe</i>	
On Automatic Parallelization of Irregular Reductions on Scalable Shared Memory Systems	422
<i>E. Gutiérrez, O. Plata, E.L. Zapata</i>	

I/O-Conscious Tiling Strategy for Disk-Resident Data	430
<i>Mahmut Kandemir, Alok Choudhary, J. Ramanujam</i>	
Post-Scheduling Optimization of Parallel Programs	440
<i>Stephen Shafer, Kanad Ghose</i>	
Piecewise Execution of Nested Parallel Programs - A Thread-Based Approach	445
<i>W. Pfannenstiel</i>	
Topic 05	
Parallel and Distributed Databases	449
<i>Burkhard Freitag, Kader Hameurlain</i>	
Distributed Database Checkpointing	450
<i>Roberto Baldoni, Francesco Quaglia, Michel Raynal</i>	
A Generalized Transaction Theory for Database and Non-database Tasks .	459
<i>Armin Feßler, Hans-Jörg Schek</i>	
On Disk Allocation of Intermediate Query Results in Parallel Database Systems	469
<i>Holger Märtens</i>	
Highly Concurrent Locking in Shared Memory Database Systems	477
<i>Christian Jacobi, Cédric Lichtenau</i>	
Parallel Processing of Multiple Text Queries on Hypercube Interconnection Networks	482
<i>Basilis Mamalis, Paul Spirakis, Basil Tampakas</i>	
Topic 06 + 20	
Fault Avoidance and Fault Removal in Real-Time Systems & Fault-Tolerant Computing	487
<i>Gilles Motet, David Powell</i>	
Quality of Service Management in Distributed Asynchronous Real-Time Systems	489
<i>Binoy Ravindran</i>	
Multiprocessor Scheduling of Real-Time Tasks with Resource Requirements	497
<i>Costas Mourlas</i>	
Designing Multiprocessor/Distributed Real-Time Systems Using the ASSERTS Toolkit	505
<i>Kanad Ghose, Sudhir Aggarwal, Abhrajit Ghosh, David Goldman, Peter Sulatycke, Pavel Vasek, David R. Vogel</i>	

UML Framework for the Design of Real-Time Robot Controllers	511
<i>L. Carroll, B. Tondou, C. Baron, J.C. Geffroy</i>	

Software Implemented Fault Tolerance in Hypercube	515
<i>D.R. Avresky, S. Geoghegan</i>	

Managing Fault Tolerance Transparently Using CORBA Services	519
<i>René Meier, Paddy Nixon</i>	

Topic 07

Theory and Models for Parallel Computation	523
<i>Michel Cosnard</i>	

Parallel Algorithms for Grounded Range Search and Applications	525
<i>Michael G. Lamoureaux, Andrew Rau-Chaplin</i>	

Multi-level Cooperative Search: A New Paradigm for Combinatorial Optimization and an Application to Graph Partitioning.....	533
<i>Michel Toulouse, Krishnaiyan Thulasiraman, Fred Glover</i>	

A Quantitative Measure of Portability with Application to Bandwidth-Latency Models for Parallel Computing.....	543
<i>Gianfranco Bilardi, Andrea Pietracaprina, Geppino Pucci</i>	

A Cost Model for Asynchronous and Structured Message Passing	552
<i>Emmanuel Melin, Bruno Raffin, Xavier Rebeuf, Bernard Viot</i>	

A Parallel Simulation of Cellular Automata by Spatial Machines	557
<i>Bruno Martin</i>	

Topic 08

High-Performance Computing and Applications	561
<i>Wolfgang Gentzsch</i>	

Null Messages Cancellation Through Load Balancing in Distributed Simulations	562
<i>Azzedine Boukerche, Sajal K. Das</i>	

Efficient Load-Balancing and Communication Overlap in Parallel Shear-Warp Algorithm on a Cluster of PCs.....	570
<i>Frédérique Chaussuier, Frédéric Desprez, Michel Loi</i>	

A Hierarchical Approach for Parallelization of a Global Optimization Method for Protein Structure Prediction	578
<i>S. Crivelli, T. Head-Gordon, R. Byrd, E. Eskow, R. Schnabel</i>	

Parallelization of a Compositional Simulator with a Galerkin Coarse/Fine Method	586
<i>Geir Åge Øye, Hilde Reme</i>	

Some Investigations of Domain Decomposition Techniques in Parallel CFD	595
<i>F. Chalot, G. Chevalier, Q.V. Dinh, L. Giraud</i>	
A Parallel Ocean Model for High Resolution Studies	603
<i>Marc Guyon, Gurvan Madec, François-Xavier Roux, Maurice Imbard</i>	
Nonoverlapping Domain Decomposition Applied to a Computational Fluid Mechanics Code	608
<i>Paulo B. Vasconcelos, Filomena D. d'Almeida</i>	
A PC Cluster with Application-Quality MPI	613
<i>M. Gołębiewski, A. Basermann, M. Baum, R. Hempel, H. Ritzdorf, J.L. Träff</i>	
Using Network of Workstations to Support a Web-Based Visualization Service	624
<i>Wilfrid Lefer, Jean-Marc Pierson</i>	
High-Speed LANs: New Environments for Parallel and Distributed Applications	633
<i>Patrick Geoffray, Laurent Lefèvre, CongDuc Pham, Loïc Prylli, Olivier Reymann, Bernard Tourancheau, Roland Westrelin</i>	
Consequences of Modern Hardware Design for Numerical Simulations and Their Realization in FEAST	643
<i>Ch. Becker, S. Kilian, S. Turek, the FEAST Group</i>	
A Structured SADT Approach to the Support of a Parallel Adaptive 3D CFD Code	651
<i>Jonathan Nash, Martin Berzins, Paul Selwood</i>	
A Parallel Algorithm for 3D Geometry Transformations in OpenGL	659
<i>J. Sébot Julien, A. Vartanian, J-L. Béchenec, N. Drach-Temam</i>	
Parallel Implementation in a Industrial Framework of Statistical Tolerancing Analysis in Microelectronics	663
<i>Salvatore Rinaudo, Francesco Moschella, Marcello A. Anile</i>	
Interaction Between Data Parallel Compilation and Data Transfer and Storage Cost Minimization for Multimedia Applications	668
<i>Chidamber Kulkarni, Koen Danckaert, Francky Catthoor, Manish Gupta</i>	
Parallel Numerical Simulation of a Marine Host-Parasite System	677
<i>Michel Langlais, Guillaume Latu, Jean Roman, Patrick Silan</i>	
Parallel Methods of Training for Multilayer Neural Network	686
<i>El Mostafa Daoudi, El Miloud Jaâra</i>	

Partitioning of Vector-Topological Data for Parallel GIS Operations: Assessment and Performance Analysis	691
<i>Terence M. Sloan, Michael J. Mineter, Steve Dowers, Connor Mulholland, Gordon Darling, Bruce M. Gittings</i>	

Topic 09

Parallel Computer Architecture - What Is Its Future?	695
--	-----

Chris Jesshope

The Algebraic Path Problem Revisited	698
--	-----

Sanjay Rajopadhye, Claude Tadonki, Tanguy Risset

Vector ISA Extension for Sparse Matrix-Vector Multiplication	708
--	-----

Stamatis Vassiliadis, Sorin Cotofana, Pyrrhos Stathis

A Study of a Simultaneous Multithreaded Processor Implementation	716
--	-----

Dominik Madoń, Eduardo Sánchez, Stefan Monnier

The MorphoSys Parallel Reconfigurable System	727
--	-----

*Guangming Lu, Hartej Singh, Ming-hau Lee, Nader Bagherzadeh,
Fadi Kurdahi, Eliseu M.C. Filho*

A Graph-Oriented Task Manager for Small Multiprocessor Systems	735
--	-----

*Xavier Verians, Jean-Didier Legat, Jean-Jacques Quisquater,
Benoit Macq*

Implementing Snoop-Coherence Protocol for Future SMP Architectures ...	745
--	-----

Wissam Hlayhel, Jacques Collet, Laurent Fesquet

An Adaptive Limited Pointers Directory Scheme for Cache Coherence of Scalable Multiprocessors	753
--	-----

Cheol Ho Park, Jong Hyuk Choi, Kyu Ho Park, Daeyeon Park

Two Schemes to Improve the Performance of a Sort-Last 3D Parallel Rendering Machine with Texture Caches	757
--	-----

Alexis Vartanian, Jean-Luc Béchenec, Nathalie Drach-Temam

ManArray Processor Interconnection Network: An Introduction	761
---	-----

Gerald G. Pechanek, Stamatis Vassiliadis, Nikos Pitsianis

Topic 10

Distributed Systems and Algorithms	767
--	-----

Gérard Padiou, André Schiper

A Cooperation Service for CORBA Objects. From the Model to the Applications	769
--	-----

Khalil Drira, Frédéric Gouëzec, Michel Diaz

Symphony: Managing Virtual Servers in the Global Village	777
<i>Roy Friedman, Assaf Schuster, Ayal Itzkovitz, Eli Biham, Erez Hadad, Vladislav Kalinovsky, Sergey Kleyman, Roman Vitenberg</i>	
Épidaure: A Java Distributed Tool for Building DAI Applications	785
<i>Djamel Fezzani, Jocelyn Desbiens</i>	
A Client/Broker/Server Substrate with 50 μ s Round-Trip Overhead	790
<i>Olivier Richard, Franck Cappello</i>	
Universal Constructs in Distributed Computations	795
<i>Ajay D. Kshemkalyani, Mukesh Singhal</i>	
Illustrating the Use of Vector Clocks in Property Detection: An Example and a Counter-Example	806
<i>Michel Raynal</i>	
A Node Count-Independent Logical Clock for Scaling Lazy Release Consistency Protocol	815
<i>Luciana Bezerra Arantes, Bertil Folliot, Pierre Sens</i>	
Mutual Exclusion Between Neighboring Nodes in an Arbitrary System Graph Tree That Stabilizes Using Read/Write Atomicity	823
<i>Gheorghe Antonoiu, Pradip K. Srimani</i>	
Topic 11	
Parallel Programming: Models, Methods and Languages	831
<i>Luc Bougé, Bill McColl, Mamoun Filali, Henk Sips</i>	
Exploiting Advanced Task Parallelism in High Performance Fortran via a Task Library*	833
<i>Thomas Brandes</i>	
A Run-Time System for Dynamic Grain Packing	845
<i>João Luís Sobral, Alberto José Proença</i>	
Optimising Skeletal-Stream Parallelism on a BSP Computer	853
<i>Andrea Zavanella</i>	
Parallel Programming by Transformation	858
<i>Noel Winstanley</i>	
Condensed Graphs: A Multi-level, Parallel, Intermediate Representation ..	866
<i>John P. Morrison, Niall J. Dalton</i>	
A Skeleton for Parallel Dynamic Programming	877
<i>D. Morales, F. Almeida, F. Garcia, J. Gonzalez, J. Roda, C. Rodriguez</i>	

Programming Effort vs. Performance with a Hybrid Programming Model for Distributed Memory Parallel Architectures	888
<i>Andreas Rodman, Mats Brorsson</i>	
DAOS — Scalable And-Or Parallelism	899
<i>Luís Fernando Castro, Vítor Santos Costa, Cláudio F.R. Geyer, Fernando Silva, Patrícia Kayser Vargas, Manuel E. Correia</i>	
Write Detection in Home-Based Software DSMs	909
<i>Weiwu Hu, Weisong Shi, Zhimin Tang</i>	
D'Caml: Native Support for Distributed ML Programming in Heterogeneous Environment	914
<i>Ken Wakita, Takashi Asano, Masataka Sassa</i>	
ParBlocks - A New Methodology for Specifying Concurrent Method Executions in Opus	925
<i>Erwin Laure</i>	
Static Parallelization of Functional Programs: Elimination of Higher-Order Functions & Optimized Inlining	930
<i>Christoph A. Herrmann, Jan Laitenberger, Christian Lengauer, Christian Schaller</i>	
A Library to Implement Neural Networks on MIMD Machines	935
<i>Yann Boniface, Frédéric Alexandre, Stéphane Vialle</i>	
Topic 12	
Architectures and Algorithms for Vision and Other Senses	939
<i>Alain Ayache, Virginio Cantoni, Concettina Guerra, Pieter Jonker</i>	
LUX: An Heterogeneous Function Composition Parallel Computer for Graphics	940
<i>Stéphane Mancini, Renaud Pacalet</i>	
A Parallel Accelerator Architecture for Multimedia Video Compression ...	950
<i>Bertil Schmidt, Manfred Schimmeler</i>	
A Parallel Architecture for Stereoscopic Processing	961
<i>Milton Romero, Bruno Ciciani</i>	
A Robust Neural Network Based Object Recognition System and Its SIMD Implementation	969
<i>Alfredo Petrosino, Giuseppe Salvi</i>	
Multimedia Extensions and Sub-word Parallelism in Image Processing: Preliminary Results	977
<i>Marco Ferretti, Davide Rizzo</i>	

Vanishing Point Detection in the Hough Transform Space	987
<i>Andrea Matessi, Luca Lombardi</i>	
Parallel Structure in an Integrated Speech-Recognition Network	995
<i>M. Fleury, A.C. Downton, A.F. Clark</i>	
3D Optoelectronic Fix Point Unit and Its Advantages Processing 3D Data	1005
<i>B. Kasche, D. Fey, T. Höhn, W. Erhard</i>	
Parallel Wavelet Transforms on Multiprocessors	1013
<i>Manfred Feil, Rade Kutil, Andreas Uhl</i>	
Vector Quantization-Fractal Image Coding Algorithm Based on Delaunay Triangulation	1018
<i>Zahia Brahimi, Karima Ait Saadi, Noria Baraka</i>	
Topic 13+19	
Numerical Algorithms for Linear and Nonlinear Algebra	1023
mpC + ScaLAPACK = Efficient Solving Linear Algebra Problems on Heterogeneous Networks	1024
<i>Alexey Kalinov, Alexey Lastovetsky</i>	
Parallel Subdomain-Based Preconditioner for the Schur Complement	1032
<i>Luiz M. Carvalho, Luc Giraud</i>	
A Preconditioner for Improved Fermion Actions	1040
<i>Wolfgang Bietenholz, Norbert Eicker, Andreas Frommer, Thomas Lippert, Björn Medeke, Klaus Schilling</i>	
Application of a Class of Preconditioners to Large Scale Linear Programming Problems	1044
<i>Venansius Baryamureeba, Trond Steihaug, Yin Zhang</i>	
Estimating Computer Performance for Parallel Sparse <i>QR</i> Factorisation .	1049
<i>David J. Miron, Patrick M. Lenders</i>	
A Mapping and Scheduling Algorithm for Parallel Sparse Fan-In Numerical Factorization	1059
<i>Pascal Hénon, Pierre Ramet, Jean Roman</i>	
Scheduling of Algorithms Based on Elimination Trees on NUMA Systems	1068
<i>María J. Martín, Inmaculada Pardines, Francisco F. Rivera</i>	
Block-Striped Partitioning and Neville Elimination	1073
<i>P. Alonso, R. Cortina, J. Ranilla</i>	

A Comparison of Parallel Solvers for Diagonally Dominant and General Narrow-Banded Linear Systems II	1078
<i>Peter Arbenz, Andrew Cleary, Jack Dongarra, Markus Hegland</i>	
Using Pentangular Factorizations for the Reduction to Banded Form	1088
<i>B. Groß, B. Lang</i>	
Experience with a Recursive Perturbation Based Algorithm for Symmetric Indefinite Linear Systems	1096
<i>Anshul Gupta, Fred Gustavson, Alexander Karaivanov, Jerzy Wasniewski, Plamen Yalamov</i>	
Parallel Cyclic Wavefront Algorithms for Solving Semidefinite Lyapunov Equations	1104
<i>José M. Claver, Vicente Hernández, Enrique S. Quintana-Ortí</i>	
Parallel Constrained Optimization via Distribution of Variables	1112
<i>Claudia A. Sagastizábal, Mikhail V. Solodov</i>	
Solving Stable Stein Equations on Distributed Memory Computers	1120
<i>Peter Benner, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí</i>	
Convergence Acceleration for the Euler Equations Using a Parallel Semi-Toeplitz Preconditioner	1124
<i>Andreas Kähäri, Samuel Sundberg</i>	
A Stable and Efficient Parallel Block Gram-Schmidt Algorithm	1128
<i>Denis Vanderstraeten</i>	
On the Extension of the Code GAM for Parallel Computing	1136
<i>Felice Iavernaro, Francesca Mazzia</i>	
PAMIHR. A Parallel FORTRAN Program for Multidimensional Quadrature on Distributed Memory Architectures	1144
<i>G. Laccetti, M. Lapegna</i>	
Stability Issues of the Wang's Partitioning Algorithm for Banded and Tridiagonal Linear Systems	1149
<i>Velisar Pavlov, Plamen Yalamov</i>	
Topic 14	
Emerging Topics in Advanced Computing in Europe	1153
<i>Renato Campo, Luc Giraud</i>	
The HPF+ Project: Supporting HPF for Advanced Industrial Applications	1155
<i>Siegfried Benkner, Guy Lonsdale, Hans Zima</i>	

TIRAN: Flexible and Portable Fault Tolerance Solutions for Cost Effective Dependable Applications	1166
<i>O. Botti, V. De Florio, G. Deconinck, F. Cassinari, S. Donatelli, A. Bobbio, A. Klein, H. Kufner, R. Lauwereins, E. Thurner, E. Verhulst</i>	
OCEANS – Optimising Compilers for Embedded Applications	1171
<i>Michel Barreteau, François Bodin, Zbigniew Chamski, Henri-Pierre Charles, Christine Eisenbeis, John Gurd, Jan Hoogerbrugge, Ping Hu, William Jalby, Toru Kisuki, Peter M.W. Knijnenburg, Paul van der Mark, Andy Nisbet, Michael F.P. O'Boyle, Erven Rohou, André Seznec, Elena A. Stöhr, Menno Treffers, Harry A.G. Wijshoff</i>	
Cray T3E Performances of a Parallel Code for a Stochastic Dynamic Assets and Liabilities Management Model	1176
<i>G. Zanghirati, F. Cocco, F. Taddei, G. Paruolo</i>	
Parametric Simulation of Multi-body Systems on Networks of Heterogeneous Computers	1187
<i>Javier G. Izaguirre, José M. Jiménez, Unai Martín, Bruno Thomas, Alberto Larzábal, Luis M. Matey</i>	
Parallel Data Mining in the HYPERBANK Project	1195
<i>S. Fotis, J. A. Keane, R. I. Scott</i>	
High Performance Computing for Optimum Design of Multi-body Systems	1199
<i>José M. Jiménez, Nassouh A. Chehayeb, Javier G. Izaguirre, Beidi Hamma, Yan Thiaudière</i>	
Topic 15	
Routing and Communication in Interconnection Networks	1203
Optimizing Message Delivery in Asynchronous Distributed Applications ..	1204
<i>Girindra D. Sharma, Nael B. Abu-Ghazaleh, Umesh Kumar V. Rajasekaran, Philip A. Wilsey</i>	
Circuit-Switched Broadcasting in Multi-port Multi-dimensional Torus Networks*	1209
<i>San-Yuan Wang, Yu-Chee Tseng, Sze-Yao Ni, Jang-Ping Sheu</i>	
Impact of the Head-of-Line Blocking on Parallel Computer Networks: Hardware to Applications	1222
<i>V. Puente, J.A. Gregorio, C. Izu, R. Beivide</i>	
Interval Routing on Layered Cross Product of Trees and Cycles	1231
<i>R. Kráľovič, B. Rován, P. Ružička</i>	

Topic 16**Instruction-Level Parallelism and Uniprocessor Architecture . . . 1241**
*Pascal Sainrat, Mateo Valero*Design Considerations of High Performance Data Cache with Prefetching 1243
*Chi-Hung Chi, Jun-Li Yuan*Annotated Memory References: A Mechanism for Informed Cache
Management 1251
*Alvin R. Lebeck, David R. Raymond, Chia-Lin Yang,
Mithuna S. Thottethodi*Understanding and Improving Register Assignment 1255
*Cindy Norris, James B. Fenwick, Jr.*Compiler-Directed Reordering of Data by Cyclic Graph Coloring 1260
*Daniela Genius, Sylvain Lelait*Code Cloning Tracing: A "Pay per Trace" Approach 1265
*Thierry Lafage, André Seznec, Erven Rohou, François Bodin*Execution-Based Scheduling for VLIW Architectures 1269
*Kemal Ebcioglu, Erik R. Altman, Sumedh Sathaye, Michael Gschwind*Decoupling Recovery Mechanism for Data Speculation from Dynamic
Instruction Scheduling Structure 1281
*Toshinori Sato*Implementation of Hybrid Context Based Value Predictors Using Value
Sequence Classification 1291
*Luis Piñuel, Rafael A. Moreno, Francisco Tirado*Heterogeneous Clustered Processors: Organization and Design 1296
*Francesco Pessolano*An Architecture Framework for Introducing Predicated Execution into
Embedded Microprocessors 1301
*Daniel A. Connors, Jean-Michel Puiatti, David I. August,
Kevin M. Crozier, Wen-mei W. Hwu*Multi-stage Cascaded Prediction 1312
*Karel Driesen, Urs Hölzle*Mispredicted Path Cache Effects 1322
*Jonathan Combs, Candice Bechem Combs, John Paul Shen***Topic 17****Concurrent and Distributed Programming with Objects 1333**
Patrick Sallé, Marc Pantel

Non-regular Process Types	1334
<i>Franz Puntigam</i>	
Decision Procedure for Temporal Logic of Concurrent Objects	1344
<i>Jean-Paul Bahsoun, Rami El-Baïda, Hugues-Olivier Yar</i>	
Aliasing Models for Object Migration.....	1353
<i>Uwe Nestmann, Hans Hüttel, Josva Kleist, Massimo Merro</i>	
Dynamic Extension of CORBA Servers	1369
<i>Marco Catunda, Noemi Rodriguez, Roberto Ierusalimschy</i>	
On the Concurrent Object Model of UML.....	1377
<i>Iulian Ober, Ileana Stan</i>	
Object Oriented Design for Reusable Parallel Linear Algebra Software ...	1385
<i>Eric Noulard, Nahid Emad</i>	
Topic 18	
Global Environment Modelling	1393
<i>Michel Déqué</i>	
The Parallelization of the Princeton Ocean Model.....	1395
<i>L.A. Boukas, N.Th. Mimikou, N.M. Missirlis, G.L. Mellor, A. Lascaratos, G. Korres</i>	
Modular Fortran 90 Implementation of a Parallel Atmospheric General Circulation Model.....	1403
<i>William Sawyer, Lawrence Takacs, Andrea Molod, Robert Lucchesi</i>	
Implementation of the Limited-Area Numerical Weather Prediction Model Aladin in Distributed Memory	1411
<i>Claude Fischer, Jean-François Estrade, Jure Jerman</i>	
Parallelization of the French Meteorological Mesoscale Model MésosNH ...	1417
<i>Patrick Jabouille, Ronan Guivarch, Philippe Kloos, Didier Gazen, Nicolas Gicquel, Luc Giraud, Nicole Asencio, Veronique Ducrocq, Juan Escobar, Jean-Luc Redelsperger, Joël Stein, Jean-Pierre Pinty</i>	
The PALM Project: MPMD Paradigm for an Oceanic Data Assimilation Software	1423
<i>A. Fouilloux, A. Piacentini</i>	
A Parallel Distributed Fast 3D Poisson Solver for Mésos-NH	1431
<i>Luc Giraud, Ronan Guivarch, Joël Stein</i>	
Porting a Limited Area Numerical Weather Forecasting Model on a Scalable Shared Memory Parallel Computer	1435
<i>Roberto Ansaloni, Paolo Malfetti, Tiziana Paccagnella</i>	

Topic 22**High-Performance Data Mining and Knowledge Discovery 1439***David Skillicorn, Domenico Talia*Mining of Association Rules in Very Large Databases: A Structured
Parallel Approach 1441*P. Becuzzi, M. Coppola, M. Vanneschi*Parallel k/h -Means Clustering for Large Data Sets 1451*Kilian Stoffel, Abdelkader Belkoniene*Performance Analysis for Parallel Generalized Association Rule Mining on
a Large Scale PC Cluster 1455*Takahiko Shintani, Masato Oguchi, Masaru Kitsuregawa*Inducing Load Balancing and Efficient Data Distribution Prior to
Association Rule Discovery in a Parallel Environment 1460*Anna M. Manning, John A. Keane***Topic 23****Symbolic Computation 1465***Mike Dewar*Parallelism in ALDOR — The Communication Library II^{it} for Parallel,
Distributed Computation 1466*Thierry Gautier, Niklaus Mannhart*

A Library for Parallel Modular Arithmetic 1476

*David Power, Russell Bradford*Performance Evaluation of Or-Parallel Logic Programming Systems on
Distributed Shared-Memory Architectures 1484*Vanusa Menditi Calegario, Inês de Castro Dutra*A Parallel Symbolic Computation Environment: Structures and
Mechanics 1492*Mantşika Matooane, Arthur Norman***Index of Authors 1497**

A Parallel Accelerator Architecture for Multimedia Video Compression

Bertil Schmidt and Manfred Schimmler

Institut für Datenverarbeitungsanlagen, TU Braunschweig,
Hans-Sommer-Str. 66, 38106 Braunschweig, Germany
{bschmidt, masch}@ida.ing.tu-bs.de

Abstract. This paper describes a parallel architecture for a variety of algorithms for video compression. It has been designed to meet the requirements of encoding and decoding according to the ITU-T standard H.263. The architecture is an implementation of the instruction systolic array (ISA) model which combines the simplicity of systolic arrays with the flexibility of a programmable parallel computer. Although the parallel accelerator unit is implemented on no more than 9 mm² of silicon it suffices to meet the compression rate necessary to send a compressed video stream through a standard ISDN terminal interface.

1 Introduction

Video compression for multimedia systems is a computationally intensive task. The desired compression factor determines the performance requirements of the underlying hardware. Several authors have suggested dedicated VLSI implementations [7] of video compression algorithms. Because of their regularity, high throughput rate, small silicon area, and low power consumption, systolic arrays have been proven as a good candidate structure for these dedicated solutions [2], [3]. Their disadvantage is the lack of flexibility with respect to the implementation of different algorithms and different problem sizes, i.e. each algorithm and each problem size requires an individual hardware solution. ISAs have been developed in order to combine the speed and simplicity of systolic arrays with flexible programmability [5]. Originally, the main application field of ISAs was supposed to be scientific computing. However, in the mid 90s the suitability of the ISA architecture for other applications was recognized, e.g. [6], [8]-[10]. In this paper we illustrate how an ISA architecture can solve all computationally intensive tasks of a multimedia video compression application efficiently. This ISA has been developed in order to compress a source data rate of 15 frames per second (fps) in CIF format according to the ITU-T standard H.263 [1] for a transmission with the given data rate of two ISDN channels.

This paper is organized as follows. The concept of the ISA is explained in Section 2. Section 3 gives an introduction to video compression with H.263. Section 4 presents the new video accelerator architecture. It is documented how the ISA is integrated on an accelerator for videophone applications. The parallel ISA algorithms

for video compression are explained in Section 5 and their performance is evaluated in Section 6. The outlook to further research topics concludes the paper in Section 7.

2 Principle of the ISA

The ISA is a quadratic array of identical processors, each connected to its four direct neighbours by data wires. The array is synchronized by a global clock. The processors are controlled by instructions, row selectors and column selectors. The instructions are input in the upper left corner of the processor array, and from there they move step by step in horizontal and vertical direction through the array. This guarantees that within each diagonal of the array the same instruction is active during each clock cycle. In clock cycle $k+1$ processor $(i+1, j)$ and $(i, j+1)$ execute the instruction that has been executed by processor (i, j) in clock cycle k . The selectors also move systolically through the array: the row selectors horizontally from left to right, column selectors vertically from top to bottom. Selectors mask the execution of the instructions within the processors, i.e. an instruction is executed if and only if both selector bits, currently in that processor, are equal to one. Otherwise, a no-operation is executed.

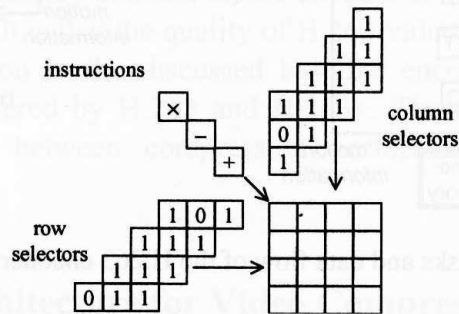


Fig. 1: Control flow in an ISA

Every processor has read and write access to its own memory. Besides that, it has a designated *communication register (C-register)* that can also be read by the four neighbour processors. Within each clock phase reading access is always performed before writing access. Thus, two adjacent processors can exchange data within a single clock cycle in which both processors overwrite the contents of their own communication register with the contents of the communication register of its neighbour. This convention avoids read/write conflicts and also creates the possibility to broadcast information across a whole row or column with one single instruction. This property can be exploited for an efficient calculation of row broadcasts, row ringshifts, and row sums which are the key-operations in many algorithms described in Section 5.

3 Multimedia Video Compression

The high amount of visual data associated with typical multimedia services establishes the need for efficient data compression schemes in order to facilitate transmission and storage applications. Several international standards have been introduced for video compression targeting different application fields. Communication applications (e.g. videophone, teleteaching) are covered by the ITU-T standard H.263. The H.263 codec codes video frames using a discrete cosine transform (DCT) on blocks of size 8×8 pixels. An initial frame is coded and transmitted as an independent frame. Subsequent frames, which are modelled as changing slowly due to small motions of objects in a scene, are coded efficiently in the inter mode using motion compensation (MC) in which the displacement of groups of pixels from their position in the previous frame are transmitted together with the DCT-coded difference between the predicted and original images. Fig. 2 shows a block diagram of the codec.

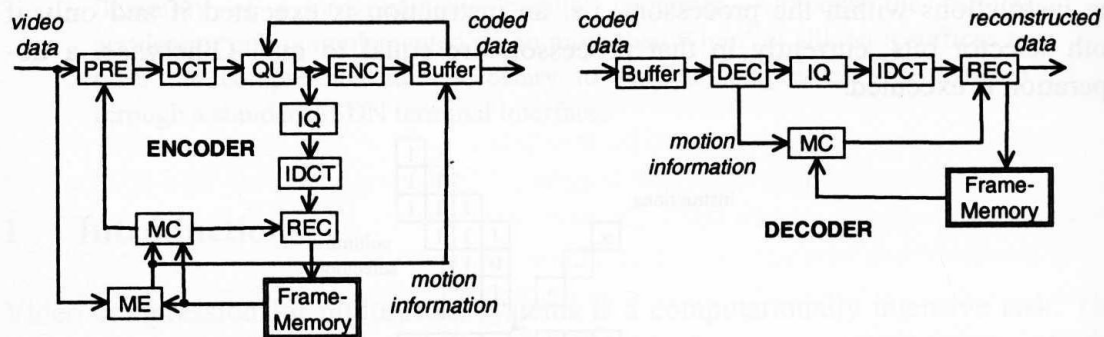


Fig. 2: Tasks and data flow of the H.263 encoder and decoder

The first step in the interframe coder is to calculate a motion vector for the current 16×16 pixels macroblock (MB) in ME (motion estimation). The motion vector is obtained by minimizing a cost function measuring the mismatch between a candidate MB in the previous frame and the current MB. Although several cost measures have been introduced [7], the most widely used one is the sum-of-absolute-differences (SAD) defined by $SAD = \sum_{k=0 \dots 15} \sum_{l=0 \dots 15} |c_{ij}(k,l) - r_{i-u,j-v}(k,l)|$, where $c_{ij}(k,l)$ represents the pixel (k,l) of a 16×16 MB from the current picture at the spatial location (i,j) , and $r_{i-u,j-v}(k,l)$ represents the pixel (k,l) of a candidate MB from a reference picture at the spatial location (i,j) displaced by the vector (u,v) . To find the MB producing the minimum mismatch error, we need to calculate SAD at several locations within a search window. The simplest, but the most compute-intensive search method, known as the full search or exhaustive search method, evaluates SAD at every possible pixel location in the search area. To lower the computational complexity, several algorithms that restrict the search area to a few have been proposed [3]. In baseline H.263, one motion vector per MB is allowed for motion compensation. Both, horizontal and vertical components of the motion vectors may be of half pixel

accuracy, but their values may lie only in the ± 15 range, limiting the search window used in ME.

The predicted MB represented by the calculated motion vector is loaded from the frame memory into MC (motion compensation). If the motion vector is of half pixel accuracy this operation requires an interpolation. The motion compensated prediction error is computed by the difference between the predicted and original MB in PRE (prediction). The resulting difference MB is transformed using a DCT of each 8×8 block, quantization by an adaptive quantizer (QU), entropy encoded using a variable-length coder (ENC), and buffered for a transmission over a fixed rate channel. The quantizer step size is calculated by evaluation of the buffer occupancy. After the quantization process, the original MB is reconstructed by the corresponding inverse operations (IQ, IDCT, REC) and stored in the frame memory. Of course, the reconstructed and original MB are not equal because of the performed lossy quantization. Since the reconstructed MB is available to encoder and decoder, it is used for prediction of the next frame.

The intra/inter mode selection is made at the MB level. If a MB does not change significantly with respect to the reference picture, an encoder can also choose not to encode it, and the decoder will simply repeat the MB located at the subject MB's spatial location. Only the processing instructions and capabilities of the decoder are standardized. The only necessary demand on the encoder is to produce a syntactically correct bitstream. The result is that the quality of H.263 video depends on the encoder implementation. In addition to the discussed baseline encoder algorithms, several negotiable options are offered by H.263 and H.263+. These optional modes allow developers to trade off between compression performance and computational complexity.

4 Accelerator Architecture for Video Compression

The analysis of the different video coding tasks and their processing efforts leads to a fixpoint processor architecture. The processor needs a small local memory for the storage and fast supply of local image data. The wordlength of the data items should be chosen to 8 bits because of the 8 bit input pixels and must also be able to process longer operands, e.g. DCT requires intermediate operands of length up to 24 bit. In order to achieve the real time requirements of the H.263 video codec several of these processors have to work in parallel. Thus, the particular processor has to be optimized with respect to its chip area and power consumption. Figure 3 depicts the processor architecture for the implementation of the H.263 video codec.

Due to the limited chip area the processor has to be very compact. In particular, it implies the choice of a bit-serial data organization. The size of the local memory is 32 internal registers plus 2 communication registers (C-registers). The word length of data items is 8 bits. Furthermore, each processor has two special constant registers 0 and -1. In addition to the registers, there are a zero flag and a negative flag that control the processing units depending on the state of the processor.

Two operand-multiplexers choose two registers of the own local memory or of the C-registers of the neighbours. The operands are propagated to the units required for execution of the current instruction. This units can be the multiplier and the arithmetic, logical, and conditional unit (ALCU). The corresponding instruction set consists of 24 instructions. The result of the execution is given to the result bus and from there they are written into the local registers or the C-registers or both.

During the execution of one instruction the processor receives the next instruction together with a column selector bit from the upper neighbour and a row selector bit from the left neighbour. The selectors are interpreted. If both are 1 then the execution of the instruction is prepared. If one of the selectors is 0 then the execution of a no-operation is initialized. Instruction and column selector are propagated bit serially to the lower neighbour and the row selector to the right neighbour during the execution of the instruction.

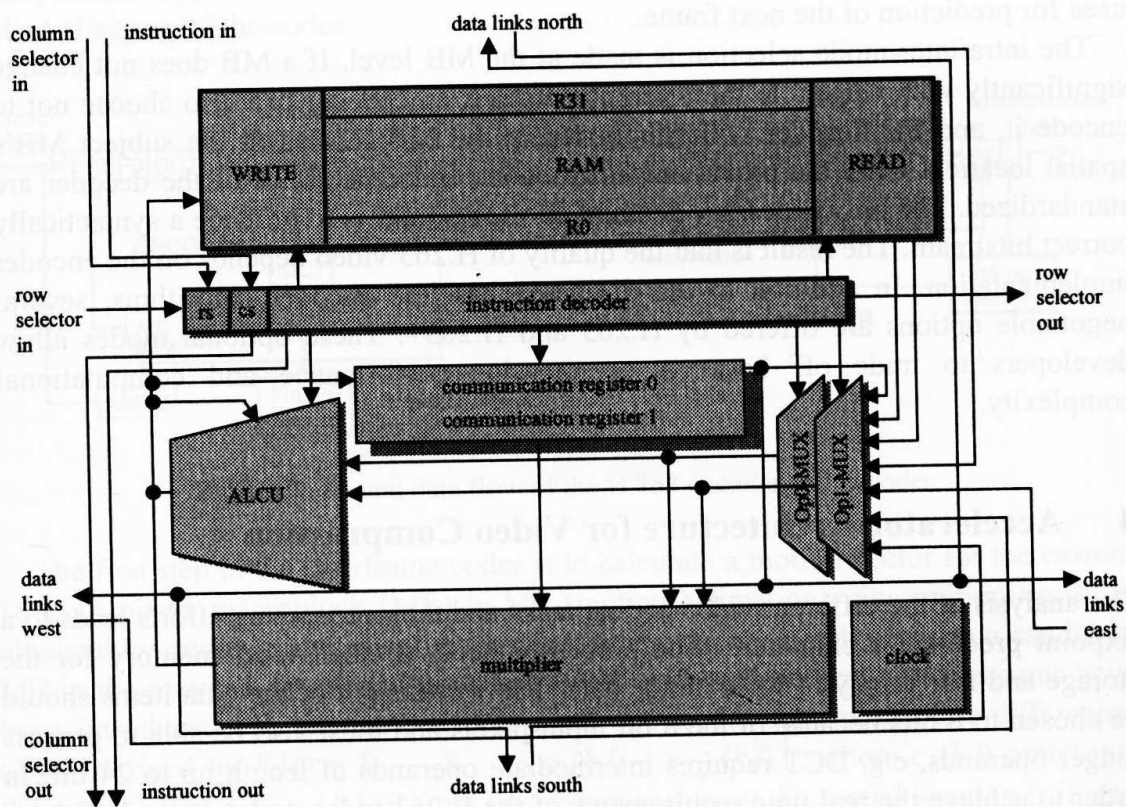


Fig. 3: Architecture of the array processor

The fine grained pipelined execution units are laid out for bitwise 300 MHz true single phase clock. Table 1 shows the areas of the different units of the processor. The full-custom design has been made with a 0.25μ digital CMOS process. An 12×8 ISA of the described processors has been structured in order to provide real-time processing for an H.263 codec at CIF resolution (352×288 pixels) and 15 fps. The resulting silicon area of the processor array is 3.7 mm^2 . At a bitwise clock frequency of 300 MHz and a word format of 8 bits the theoretical peak performance of the array

is 3.6 GIPS. The corresponding estimated power consumption is 120 mW. In order to exploit the computation capabilities of this unit, it is necessary to provide data and control information at an extremely high speed. Therefore, a cascaded memory concept is implemented on-chip that forms a fast input and output environment for the parallel processing unit (see Figure 4).

For the fast exchange of data with the processor array each processor has two memory banks. Each memory bank contains 8 interface registers. One of these banks is always assigned to the corresponding processor, the other to the internal RAM by means of a fast data channel. The exchange of data between ISA and the internal RAM is done by bank switching. Both memory banks can be active at the same time, i.e. data transfer between ISA and internal RAM can be done concurrently to the execution of an ISA program. The internal RAM can also communicate bidirectionally with the external SDRAM.

The data transfer is controlled by an on-chip controller. The controller is started by the sequential standard RISC core (e.g. Hitachi SH4) and operates autonomously afterwards. It receives instructions from an instruction queue. The controller supplies the processor array with instructions and selectors that are stored in the ISA program memory. The instruction queue and the ISA program memory are located in the external SRAM, separated from the data. This part of the SDRAM is exclusively available to the controller by means of a fast channel. The internal RAM is organized as single ported static RAM. Its size of 8 KByte is mainly determined by the H.263 implementation (see Section 5). The layout of the complete video accelerator with a 0.25 μ digital CMOS process requires an area of 8.1 mm² as depicted in Table 2.

Table 1: Area of the individual processor.

RAM	12.144 μm^2
Multiplier	6.398 μm^2
Read-write-logic	2.874 μm^2
ALCU	5.125 μm^2
Instruction decoder	4.011 μm^2
Addressing	1.190 μm^2
Routing	1.674 μm^2
Operand multiplexers	2.133 μm^2
Clock	2.613 μm^2
Sum	38.135 μm^2

Table 2: Area of the accelerator.

ISA	3.7 mm ²
Controller	1.1 mm ²
Internal RAM	3.3 mm ²
Sum	8.1 mm²

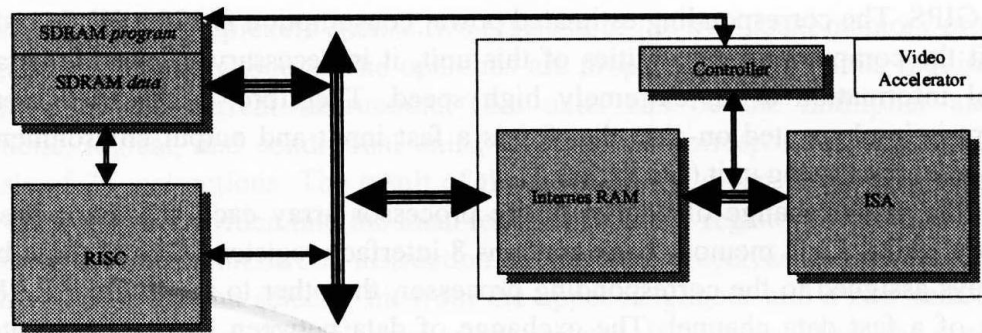


Fig. 4: Video accelerator architecture and data paths to the environment.

5 Parallelization of Video Compression Algorithms

The tasks of the hybrid coding scheme can be subdivided into low level, medium level, and high level tasks. Low level tasks typically operate on pixel data. They are highly regular and offer a large potential of data parallelism. Examples for low level tasks are full search motion estimation, DCT, and prediction. These tasks require a main percentage of the overall processing effort. Medium level tasks like quantization or entropy coding operate on the results of the low level tasks. Computation requirements are significantly lower than for low level tasks. The high level tasks comprise the bitstream handling and several control tasks, primary on encoder side, like quantizer control and decisions for coding strategy. High level tasks show an irregular control flow in conjunction with low computational rates. For control tasks a certain amount of flexibility with respect to modifications of the algorithms is advantageous. The proposed coprocessor architecture for multimedia comprise a flexible general-purpose RISC core with a less flexible accelerator adapted towards a specific type of processing. Thus, the RISC core performs the high level tasks of lower computational requirements, whereas the video accelerator executes the computation intensive but regular low level tasks and a part of the medium level tasks. The parallelization of the H.263 coding tasks on the proposed accelerator architecture is described in the following.

ME: In the 2D array structure of a 4×8 ISA the calculation of one SAD can be efficiently parallelized as follows: Assume, the current MB data $c(k,l)$ and the reference frame candidate MB data $r(k,l)$, $k,l = 0, \dots, 15$, are stored in the processor array such that processor (i,j) , $i = 0, \dots, 3$, $j = 0, \dots, 7$, holds $c(4 \cdot i, 2 \cdot j), \dots, c(4 \cdot i + 3, 2 \cdot j + 1)$ and $r(4 \cdot i, 2 \cdot j), \dots, r(4 \cdot i + 3, 2 \cdot j + 1)$ in 16 internal registers. Now each node (i,j) computes the partial sum $\sum_{m=0, \dots, 3, n=0, \dots, 1} |c(4 \cdot i + m, 2 \cdot j + n) - r(4 \cdot i + m, 2 \cdot j + n)|$ within 30 instructions. The result is written into the two C-registers $C1$ (most significant byte) and $C0$ (least significant byte). The partial sums are added along each processor row within 2 instructions using two efficient row sum operations (see Section 2). The first operation adds up the values of $C0$ and the second operation adds up the values of $C1$ and the carry-bit of the previous addition. The total sum (SAD) for the candidate MB

is completed by the corresponding two column sum operations in the rightmost processor column and the result is written into *C0* and *C1* of the lower right processor. This SAD is compared in the lower right processor to the stored SAD resulting from the previous comparison. The smaller one as well as the corresponding motion vector is kept for future comparisons.

The preceding computation procedure is repeated until all possible candidate MBs are compared and the final motion vector is obtained. To speed up the computation the calculated SAD value is compared to a preselected threshold value. The search process terminates if the calculated SAD value is smaller than the threshold. Because the processor array of the video accelerator is of size 12×8 three different SAD values for the current MB can be calculated in parallel by a simple replication of the program for the 4×8 array.

Shifting of the candidate MB within the search area depends on the chosen search strategy. In order to avoid delay times in the systolic architecture it is necessary to know these locations in advance, and thus the next candidate MB data can be preloaded from the internal RAM into the ISA during the processing of the current MB. The ± 15 pixels 2-step search with subsequent half pixel refinement is suitable and leads to a very good coding quality [3]. In the first step, it evaluates the motion vector candidates with distance two between two nearby search points within the ± 15 pixels search area. Thus, 225 SAD values of previously known locations have to be calculated in the worst case. In the second step, the SAD values of the 8 neighbours of the calculated minimum point are evaluated. A subsequent half pixel search around the full pixel accurate displacement vector evaluates the motion vector with half-pixel accuracy. This involves the additional evaluation of 8 candidate blocks per motion vector. The locations of candidate points in the second iteration step and the half pixel search are data dependent. Thus, the data locations to be used at the next step are transferred to the controller when the current step is completed. At the beginning of the new step, the controller loads the corresponding reference data into the ISA. MBs of adjacent candidate vectors overlap quite significantly. In order to reduce memory accesses (bandwidth) only the non-overlapping data have to be loaded into the processor array. Since SAD values for 3 horizontally neighboured motion vectors are computed in the processor array in parallel only 2×20 pixels have to be loaded when the search area moves vertically up in the first step of the 2-step search. Search areas of adjacent MBs also overlap. Thus, only the non-overlapped data of size 48×16 pixels is loaded from the external SDRAM to the internal RAM, which can be performed concurrently to the motion vector computation of the current MB.

MC: The complete reference frame MB corresponding to the calculated motion vector has to be loaded into the ISA. In contrast to the other processing tasks ME is only performed on the luminance data of the MB. For MC and the preceding tasks another mapping of MB data onto the ISA is used. Each processor holds a 2×2 subarray. Thus, each of the six 8×8 blocks of the MB is mapped onto a 4×4 processor subarray of the 12×8 ISA. If the motion vector is of half pixel accuracy linear interpolation is processed in the processor array.

PRE/REC: The motion compensated MB and the original MB are simply subtracted (prediction) resp. added (reconstructed). Only 4 subtraction resp. additions have to be performed within each processor in parallel.

DCT/IDCT: The DCT of an 8×8 block $x_{i,j}$, $i,j=0,\dots,7$, can be expressed as

$$y_{k,l} = \sum_{i=0}^7 c_{i,k} \left[\sum_{j=0}^7 x_{i,j} \cdot c_{j,l} \right], \text{ where } c_{j,l} = \cos((2j+1)l\pi/16) \quad (1)$$

The core operation in DCT computation is the multiply accumulation (MAC).

$$y_{2l} = \sum_{j=0}^3 (x_j + x_{7-j}) \cdot c_{j,2l}; \quad y_{2l+1} = \sum_{j=0}^3 (x_j - x_{7-j}) \cdot c_{j,2l+1} \quad \text{for } l = 0, \dots, 3 \quad (2)$$

Therefore, the multiplier of the introduced ISA processor has been designed to perform MAC efficiently, i.e. simultaneously with multiplication, a third operand can be added to the result without additional delay. As already indicated by the brackets in (1), the 2D transform can be decomposed into cascade of two subsequent 1D transforms applied horizontally and vertically on each row, resp. column. Additional efficiency can be gained, if we split the 1D transform of a sequence x_j , $j = 0, \dots, 7$, into even and odd numbered frequency samples:

This algorithm is applied for the 8×8 DCT within each 4×4 processor subarray. The input values for the 1D DCT are initially permuted within each processor row of length 4, such that each processor j , $0 \leq j \leq 3$, holds x_j and x_{7-j} and the addition $x_j + x_{7-j}$ and the subtraction $x_j - x_{7-j}$ can be performed in parallel. Now each processor j , $0 \leq j \leq 3$, computes y_{2j} and y_{2j+1} according to (2) in 4 stages. The computation in each stage requires two MAC operations per processor. Permutation of the values $x_j + x_{7-j}$ and $x_j - x_{7-j}$ between every two stages involves routing of data, which is performed efficiently within each processor row by ringshifting. The coefficient values are precomputed and loaded into the processor array when needed. This load operation takes only a small number of instruction cycles since the values can be broadcasted along the columns of the ISA. The 1D DCT along the processor columns is computed by the *reflected ISA program* for the rows. The IDCT is implemented in a similar way.

QU/IQ: The division operation for quantization is evaluated by multiplication with the reciprocal value in each processor in parallel. The quantizer step size is controlled by loading the corresponding coefficients into the processor array.

ENC/DEC: Due its irregularity the variable length en/decoding and entropy en/decoding is not suitable for an efficient parallelization on the ISA. Thus, the sequential RISC performs these operations. Because of their low computational requirements, their runtime on the RISC is dominated by the processing time of the next/previous MB on the video accelerator.

6 Performance Evaluation

For the runtime determination of the parallel programs described in Section 5 on the introduced accelerator architecture the number of required instructions is multiplied with the corresponding clock cycle time of 26.7 ns. Additionally, the data transfer

between ISA and internal RAM is considered with a throughput of 150 MByte/s. The results are shown Table 3. Note that the times for ME are worst cases, i.e. calculated SAD values are never smaller than the preselected threshold.

Table 3: Worst case runtime on the video accelerator for encoding and decoding per MB. It includes computing time on the ISA and data transfer time between ISA and internal RAM. Data transfer is divided into concurrent transfer, i.e. the transfer time is dominated by the computing time on the ISA, and not concurrent transfer, i.e. during the transfer time no computations are performed on the ISA.

Task		Runtime video accelerator	Data transfer ISA ↔ internal RAM	
			Concurrent to ISA	Not concurrent to ISA
En-coder	ME step 1	103 μs	Load 4080 Bytes	Load 640 Bytes
	step 2	6 μs	Load 36 Bytes	Load 288 Bytes
	half pixel	8.5 μs	Load 18 Bytes	Load 17 Bytes
	MC	5 μs	Load 128 Bytes	Load 451 Bytes
	PRE	0.5 μs		
	DCT and QU	10 μs		
	IQ and IDCT	10 μs	Store 384 Bytes	
	REC	3 μs		Store 384 Bytes
De-coder	IQ and IDCT	10 μs	Load 451 Bytes	Load 384 Bytes
	MC	2 μs		
	REC	3 μs		Store 384 Bytes
Sum per MB		161 μs	5097 Bytes	2584 Bytes
Sum per CIF		64 ms	2.02 Mbytes	1.0 MBytes

The amount of data transfer between external SDRAM and internal RAM per MB is

Encoder: current MB (384 Bytes), overlapping search area (768 Bytes), chrominance blocks of motion compensated MB incl. border pixels (162 Bytes), coded MB (384 Bytes), reconstructed MB (384 Bytes).

Decoder: current MB (384 Bytes), motion compensated MB incl. border pixels (451 Bytes), reconstructed MB (384 Bytes).

The resulting total amount of data transfer of 3301 Bytes per MB, resp. 1.3 MBytes per CIF frame, can be transferred concurrently to the accelerator activities with a reasonable throughput of the SDRAM bus, e.g. 100 MByte/s.

7 Conclusions

In this paper we have presented an ISA architecture for algorithms required for video compression. The accelerator unit has been implemented on an area of 8.1 mm² of silicon using a 0.25 μ digital CMOS process. It is capable of on-line encoding and decoding of a video stream of 15 fps in CIF format with the H.263. The global architecture of the accelerator unit has been discussed as well as the detailed implementation of the single processing element of the 12 $\times$ 8 array. It has been

shown how the new architecture can be programmed for applications as motion estimation, motion compensation, DCT, and quantization. Apart from its performance figures, the most promising property of the accelerator unit is its flexibility. It would be interesting to study the performance for the new architecture for applications like [8], [9], [10].

References

1. Video Coding for low bitrate communication, ITU-T Draft Recommendation H.263, Geneva, May (1996)
2. Chang, Y.-T., Wang, C.-L.: New Systolic Array Implementation of the 2-D Discrete Cosine Transform and Its Inverse, *IEEE Trans. Circ. Syst. Vid. Tech.* 5 (2) (1995) 150-157
3. Cheng, S.-C., Hang, H.-M.: A Comparison of Block-Matching Algorithms Mapped to Systolic-Array Implementation, *IEEE Trans. Circ. Syst. Video Tech.* 7 (5) (1997) 741-757
4. Kunde, M., et al.: The Instruction Systolic Array and its Relation to Other Models of Parallel Computers, *Parallel Computing* 7 (1988) 25-39
5. Lang, H.-W.: The Instruction Systolic Array, a parallel architecture for VLSI, *Integration, the VLSI Journal* 4 (1986) 65-74
6. Lang, H.-W., Maaß, R., Schimmler, M.: The Instruction Systolic Array - Implementation of a Low-Cost Parallel Architecture as Add-On Board for Personal Computers, *Proc. HPCN 94*, LNCS 797, Springer Verlag (1994) 487-488.
7. Pirsch, P., Stolberg, H.-J.: VLSI Implementations of Image and Video Multimedia Processing Systems, *IEEE Trans. Circ. Syst. Video Tech.*, 8 (7) (1998) 878-891
8. Schimmler, M., Lang, H.-W.: The Instruction Systolic Array in Image Processing Applications, *Proc. Europto 96*, SPIE 2784 (1996) 136-144
9. Schmidt, B., Schimmler, M., Schröder, H.: Long Operand Arithmetic on Instruction Systolic Computer Architectures and Its Application to RSA cryptography, *Proc. EuroPar'98*, LNCS 1470, Springer Verlag (1998) 916-922
10. Schmidt, B., Schimmler, M., Schröder, H.: The Instruction Systolic Array in Tomographic Image Reconstruction Applications, *Proc. PART'98*, Springer Verlag (1998) 343-354