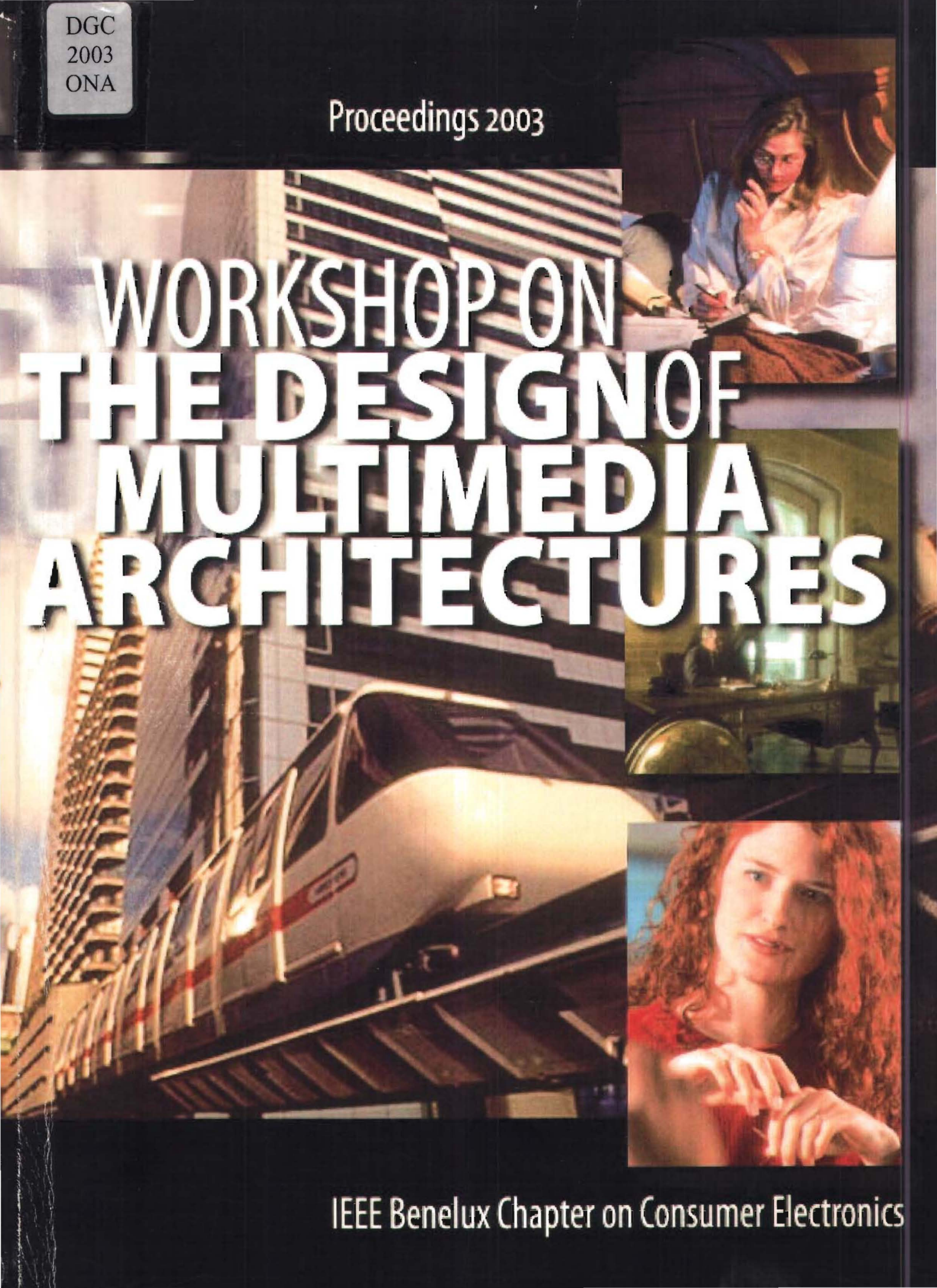
On the design of multimedia architectures : proceedings of a one-day workshop, Eindhoven, December 18, 2003

Proceedings 2003



WORKSHOP ON THE DESIGN OF MULTIMEDIA ARCHITECTURES

IEEE Benelux Chapter on Consumer Electronics

Peter H.N. de With (ed.)

On the Design of Multimedia Architectures

Proceedings of a one-day workshop

organized by the

IEEE Benelux Chapter on Consumer Electronics

December 18, 2003

Sponsors

Primary sponsor

IEEE

Supporting sponsor

Shannon Foundation

Workshop hosting

Embedded Systems INSTITUTE

On the design of multimedia architectures: proceedings of a one-day workshop, Eindhoven, December 18, 2003 / ed. by Peter H.N. de With. - Eindhoven : Technische Universiteit Eindhoven, 2003.
ISBN 90-386-0822-5
NUR 992
Subject headings : multimedia : proceedings / information systems ; modelling / computer design
CR Subject Classification (1998) : C.4, C.3, C.1, I.4, I.6

Contents

Preface

The IEEE Chapter on Consumer Electronics in the Benelux was founded in the late nineties to support events that are related to applications of Consumer Electronics. The CE domain is growing yearly, because computing, communication and storage devices are shrinking by means of continuous advances in technology.

The first workshop of the Benelux CE Section was devoted to multimedia video coding for Internet applications. The MPEG video compression standards have been a phenomenal success for the recording and digital distribution of video signals. The most important standard for digital moving video signals is beyond doubt the MPEG standard, of which the MPEG-2 video standard is most widely applied (e.g. DVD) and MPEG-4 is studied for e.g. portable applications of video systems. The widely accepted use of communication in computer networks is gradually becoming part of the consumer electronics area, leading to communicating consumer video over the Internet. This was the theme of the first workshop.

Despite the bust of the Internet and telecommunications bubble in the first years of this millennium, the technology has not stopped from innovating, although the pace of investments has decreased. An example of continuous technology improvement is the increased density of transistors in a chip, which enables advanced functionality inside e.g. portable systems and many other consumer products. This development poses system designers with the challenging problem of dealing with very complex and diverse architectures inside a single system. The complexity of many systems has grown so much, that virtually every CE manufacturer is outsourcing the design of particular system modules or subsystems. The system design owner should solve the problem of smooth integration and operation of the various subsystems. This complexity control problem occurs both in software and hardware design. The above considerations have resulted in the theme of the second workshop.

The first lecture of Dr. Martin Bolton deals with the growing complexity of DVD-based systems. Dr. Bolton is with ST Microelectronics, one of the leading semiconductor companies providing MPEG-based chip sets for various kinds of applications. The DVD system has become mature and DVD recording systems are affordable. The applications are further diversifying and the computer industry has embraced various RW formats for recording of data. The complexity of such systems is continuously augmenting due to all kinds of operations modes and a large number of interfaces. We are happy to hear from Mr. Bolton what the new challenges are in the race for ever increasing functionality.

We are particularly honored that Prof.dr. Yolande Berbers affiliated with the University of Leuven, Belgium, will address technologies in software architectures. Prof. Berbers is involved in multimedia system design and involved with mobile services. In her lecture, she will address the problems that occur when services lead to transactions between various mobile devices. She has been involved also in the design of real-time embedded software. With an increasing amount of mobile devices around a consumer, made by different manufacturers, the question for smooth and secure exchange of data is of vital importance.

The design of a new emerging software platform for TV systems is presented by Erik Moll. He is with Philips Digital Systems Lab in Eindhoven and was the principal architect of the so-called Multimedia Home Platform (MHP). This represents a software application layer that was standardized recently for digital TV systems. The MHP stack is a flexible software architecture that should enable the addition of new functions within the same framework. It even accept JAVA-programmed functions and besides enabling all digital TV applications including video, audio and data reception, processing and playback. At the time of the workshop, it is not sure whether MHP will be widely accepted in the consumer market. It surely represents an interesting software architecture, reflecting the state-of-the-art complexity of flexible multimedia systems.

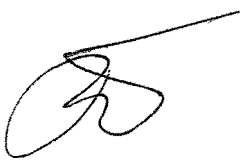
Mladen Berekovic presents the fourth lecture of this workshop. He is with one of the leading research centers for on-chip integration of advanced multimedia functions in Europe, the Microelectronics Center of the University of Hannover, Germany. The presentation of Mr. Berekovic addresses the design of a new chip that enables the execution of MPEG-4 coding, which is today one of the most advanced multimedia standards available. One of the problems of MPEG-4 chip design is that the standard allows object-oriented coding, which leads to a more dynamical behavior in architectures with respect to data exchange and computing and memory usage. Given the diverse nature of the coding tools within MPEG-4, the architecture aimed at consists of a heterogeneous set of embedded processors.

The poster session contains five interesting papers dealing with the design of subsystems within the development of larger projects. One poster deals with Quality of Service, which can improve the flexibility of assignments in consumer systems. A second poster addresses the use of an SIMD-VLIW architecture to evaluate its performance for an advanced new feature, called face recognition. Another poster discusses the design of a multi-channel MPEG system for surveillance applications. The last two posters deal with timing properties, but in different ways. One aspect studied is the prediction of real-time application in a large component-based architecture, whereas the second maps an MPEG-4 video decoder on a multiprocessor system, while attempting to derive the execution times of particular functions.

The final lecture is given by Dr. Egbert Jaspers and Prof.dr. Jef van Meerbergen from Philips Research and the University of Technology Eindhoven. In this lecture, they report on MPEG/H264 application studies and the related design of multimedia architectures. Since the amount of functions is growing and the question for more flexibility increases, the design paradigm for new chip architectures may change. Fast product development may be obtained with networks of identical or similar processors on a single chip, so that the system design further shifts to the software side. With this glimpse into the future, the workshop will be closed.

The IEEE Benelux Chapter on Consumer Electronics is happy to organize this workshop and offering the enclosed topics to a wide audience. The Chapter is part of the international IEEE CE Society. The Chapter gratefully acknowledges the Embedded Systems Institute (ESI) located at the premises of the University of Technology Eindhoven, for its supporting contributions to this workshop (e.g. registration, secretary support) and for co-hosting the day. The theme of the workshop and various issues in multimedia, such as hardware and software co-design also apply to embedded systems. The Chapter also acknowledges the supportive sponsoring of the Shannon Foundation, particularly for the proceedings.

These proceedings contain a mixture of slide copies and poster papers addressing the themes of the individual lectures and posters. This mixed approach was chosen to give maximum flexibility to the authors with minimum effort, thereby allowing the input of the latest material.



Peter H.N. de With

Board member IEEE Benelux Chapter on Consumer Electronics,
Professor Video Coding and Architectures, Electrical Engineering Faculty,
University of Technology Eindhoven, The Netherlands.

Program of "On the Design of Multimedia Architectures"

One-day workshop at the University of Technology Eindhoven, The Netherlands, on December 18, 2003. Organized by the IEEE Benelux Section on Consumer Electronics and hosted by the Embedded Systems Institute, Eindhoven, The Netherlands.

Organization committee
Prof.dr.ir. Peter H.N. de With (Eindhoven Univ. of Technology / LogicaCMG)
Prof.dr.ir. Kees A.S. Immink (Turing Machines, Rotterdam)
Dr.ir. Egbert Jaspers (Philips Research Labs, now with LogicaCMG)
Assisting program committee members:
Prof.dr.ir. Jef L. van Meerbergen (Philips Research Labs, Eindh. Univ. of Technology)
Dr.ir. Michel R.V. Chaudron (Eindhoven Univ. of Technology, Fac. Comp. Science)

Workshop Program

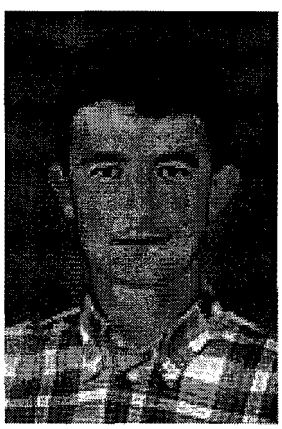
09.00-09.35 hrs.	Registration and coffee
09.35-09.45 hrs.	Opening workshop, Prof.dr.ir. Peter H.N. de With (LogicaCMG and Univ. of Technol. Eindhoven, NL)
09.45-10.30 hrs.	Dr. Martin Bolton (ST Microlectronics, Bristol, UK) "Flexible A/V codec architectures for DVD applications"
10.30-11.15 hrs.	Prof.dr. Yolande Berbers (Computer Science Dept., Cath. Univ., Leuven, Belgium) "SW development for embedded systems, based on components and contracts"
Break	
11.45-12.30 hrs.	Ir. Erik Moll (Philips Digital Systems Labs, Eindhoven, NL) "Multimedia Home Platform SW architecture"
Lunch	
13.45-14.30 hrs.	Dipl.-ing. Mladen Berekovic (University of Hannover, D) "System on Chip Architectures for MPEG-4 Video"
14.30-15.30 hrs.	Poster Session (incl break at end), including posters about QoS architectures, face processing in cameras, mapping of MPEG-4 decoders, RT aspects in CB architectures, etc. Poster session contents on next page
15.30-16.15 hrs.	Dr.ir. Egbert Jaspers and Prof.dr.ir. Jef van Meerbergen (Philips Research, Univ. of Technol. Eindhoven, NL) "On the Design of Multimedia Software and Future System Architectures"
16.15 hrs.	Closing, Prof.dr.ir. Kees A.S.Immink (Turing Machines, NL) Chairman of IEEE Benelux CE Section

Workshop Poster Contents

Contributors

Martin Bolton is the Manager of Video Architectures in the DVD Division of STMicroelectronics in Bristol. He has a B.SC degree in Electrical Engineering from Imperial College, London, UK, and a D.Phil. degree from the University of Sussex, UK. After earlier work on image generation for flight simulation, and a period on the staff at Bristol University, he has been involved in image compression architectures for over 15 years. Mr. Bolton is member of the Technical Program Committee of the Consumer Electronics Section of the IEEE.

Yolande Berbers obtained her MSc degree in computer science from the K.U.Leuven in 1982. In 1987, she received a Ph.D. degree in computer science with a thesis in the area of distributed operating systems at the same university. Since 1990 she has been an associate professor in the department of computer science. She spent 5 months during 1985 and 1986 at the INRIA, France, involved in the Chorus project. She was an invited professor at the University of Inshasa (Zaire) in 1988, and at the Franco-Polish School of New Information and Communication Technologies (Poznan, Poland) in 1995 and 1996. She teaches advanced courses on real-time and embedded systems, and on computer architecture. Her research interests include software engineering for embedded software, middleware, real-time systems, component-oriented software development, distributed systems, environments for distributed and parallel applications, mobile agents. She is currently coordinator of CoDAMoS (Context-Driven Adaptation of Mobile Services), an ambient intelligence project with 3 other Belgian universities (UGent, VUB and LUC) and 18 industrial partners.

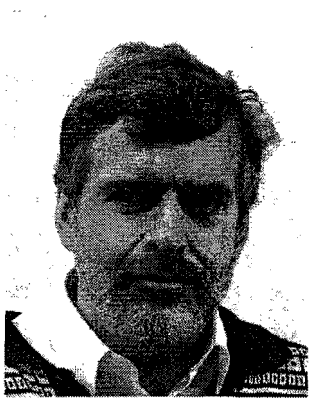


Erik Moll studied mathematics and computer science at the Eindhoven University of Technology. In 1985, he joined Philips Business Communication Systems (BCS) in Hilversum, The Netherlands, where he worked on SW development for voice & data communication systems. From 1989 onwards, he worked as a SW project-leader, e.g. migrating the complete software of a communication system from Chill to Ada and C++. He was involved a.o. in the definition of wireless communication systems (DECT) and on voice/data integration. In 1995, he joined Philips Research Labs to work on Computer graphics and UI Software Technology and worked on projects like a speech-controlled TV system and on 3D graphics with force-feedback UI for medical context. In 2000, he joined Philips Digital Systems Labs to work as Software Architect on Java in Consumer Electronic products. He conducted performance analysis of early DVB-MHP (the Multimedia Home Platform) implementations and became later the main Philips MHP architect. Mr. Moll has been a teacher for courses on SW programming and UI software technology at the Philips Centre of Technical Training and he was involved in on-site MHP trainings. He is a member of the advisory board of the Computer Science Dept. of the Polytechnical School of Utrecht, The Netherlands.

Mladen Berekovic (M'96) received the Dipl.-Ing. degree in electrical engineering from the University of Hannover, Germany, in 1995. Since then, he has been a Research Assistant with the Institute of Microelectronic Circuits and Systems of the University of Hannover. His current research interests include VLSI architectures for video signal processing, MPEG-4, System-on-Chip (SOC) designs, and simultaneously multi-threaded (SMT) processor architectures.

Egbert Jaspers was born in Nijmegen, the Netherlands, in 1969. He studied Electrical Engineering at the Venlo Polytechnical College, which resulted in the B.Sc. degree in 1993. Subsequently, he joined Philips Research Laboratories in Eindhoven, where he worked on video compression for digital HDTV recording. At the end of 1993, he left Philips for three years to pursue a M.Sc. degree at the Eindhoven University of Technology, from which he graduated in 1996. In the same year he joined the Philips Research Laboratories in Eindhoven as a Research Scientist in the Video Processing and Visual Perception Group. He participated in the design of several video-processing functions and gradually refocused his research to the architecture-related aspects of video processing for developing heterogeneous multiprocessor architectures for consumer systems. In 2000 he received a Chester Sall Award for the best papers of the IEEE CE Transactions in 1999. In April 2003 he obtained a Ph.D. degree at the University of Technology Eindhoven, for his work on Architecture Design of Video Processing Systems on a Chip. In November 2003, he joined LogicaCMG where he is deployed as a system architect consultant.

Jef van Meerbergen received the Electrical Engineering Degree and the Ph. D. degree from the Katholieke Universiteit Leuven, Belgium, in 1975 and 1980, respectively. In 1979 he joined the Philips Research Laboratories in Eindhoven, the Netherlands. He was engaged in the design of MOS digital circuits, domain-specific processors and general-purpose digital signal processors. He was the project leader of the Sigma-Pi project which delivered the first general purpose DSP within Philips. In 1985, he started working on application-driven high-level synthesis. Initially this work was targeted towards audio and telecom DSP applications. This work resulted in the AR|T system which is available from Frontier Design and which is used in projects like CD90, DAB, a UMTS turbodecoder etc. His current interests are in design methods, heterogeneous multiprocessor systems and reconfigurable architectures. Jef van Meerbergen is a Philips Research Fellow and a part-time professor at the Eindhoven University of Technology. He received the best paper award at the 1997 ED&TC conference.



Peter H.N. de With graduated (M.Sc.) in electrical engineering from the University of Technology Eindhoven, The Netherlands in 1984 and received his Ph.D. degree from the University of Technology Delft, The Netherlands, in 1992. He joined Philips Research Labs Eindhoven in 1984, where he became a member of the Magnetic Recording Systems Dept. From 1985 to 1993 he was involved in several European research projects on SDTV and HDTV recording. In the early nineties, he was the principal video coding expert in the standardization of DV systems. In 1994, he joined the TV Systems Dept., where he was leading the design of advanced programmable video architectures. In 1996, he became senior TV systems architect and in 1997, he was appointed as full-time professor at the University of Mannheim, Germany, in the faculty of Computer Engineering. In 2000, he joined CMG Eindhoven (now LogicaCMG) as a principal consultant and he became professor at the University of Technology Eindhoven, (Electrical Engineering), where he leads multimedia video system design. He has written numerous papers and holds over 40 patents. In 1995 and 2000, he co-authored papers that received the IEEE CES Transactions Papers Award. In 1996, he received a company invention award and in 1997, Philips received the ITVA Award for DV standard contributions. Mr. De With is a senior member of the IEEE, program committee member of the CES and ICIP, Chairman of the Benelux Information Theory Working Group, Scientific Board member of LogicaCMG and ASCI.

"Flexible A/V codec architectures for DVD applications"

Dr. Martin Bolton

ST Microlectronics, Bristol, UK

FLEXIBLE AUDIO/VIDEO CODEC ARCHITECTURES FOR DVD APPLICATIONS

Martin Bolton
STMicroelectronics, Bristol

Outline

- Motivation
- The VLIW Processor
- Codec Architecture
- Decoder Architecture
- SOC to SAC

What do we need?

- Now:
  - A/V codec cells supporting MPEG-2 & MPEG-4 for consumer applications, e.g. DVD recorder, PVR
- In the future:
  - Extensions to support emerging standards without major rework of the design

Beyond MPEG-2

- New Video Standards:
  - Windows Media Video
  - Real Video
  - H.264
  - AVS
- High Definition decoding is now becoming more important

Hardware vs. Software

- (Mainly) hardware architecture:
  - small, low power
  - long design cycle, inflexible
- (Mainly) software architecture:
  - flexible
  - processor choice is constrained in consumer applications
- Challenge is to find the best hybrid solution

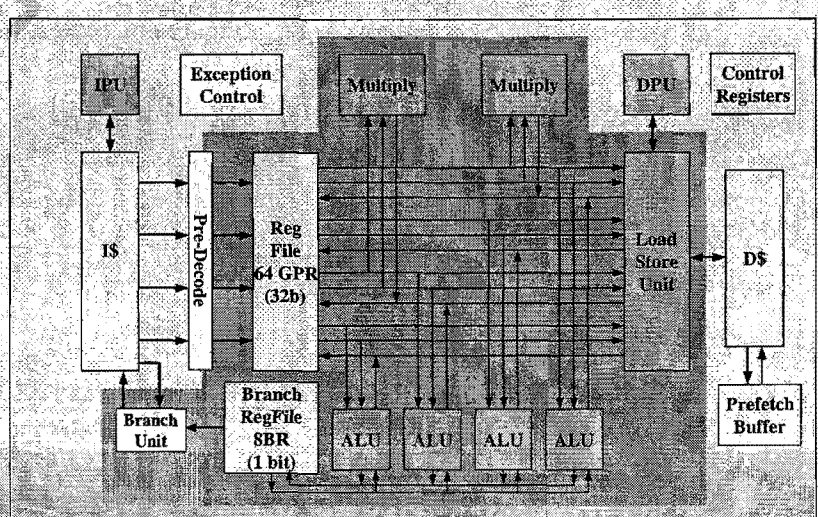
Our Approach

- VLIW processor + acceleration for selected functions
- Guiding principles:
  - Keep VLIW small, fast and simple
  - Low architectural complexity
  - Keep flexibility where it is needed

VLIW Processor

- ST220: a member of the "Lx" Processor Family
- A Joint *HP Labs* and *STMicroelectronics* Design
  - Technology platform for System-On-Chip (SOC) VLIW cores
- Lx is an "architecture framework"
  - Customizable and Scalable, high performance VLIW
  - Includes hardware and toolchain
  - Presented as one compatible architecture family to the user

Processor Architecture



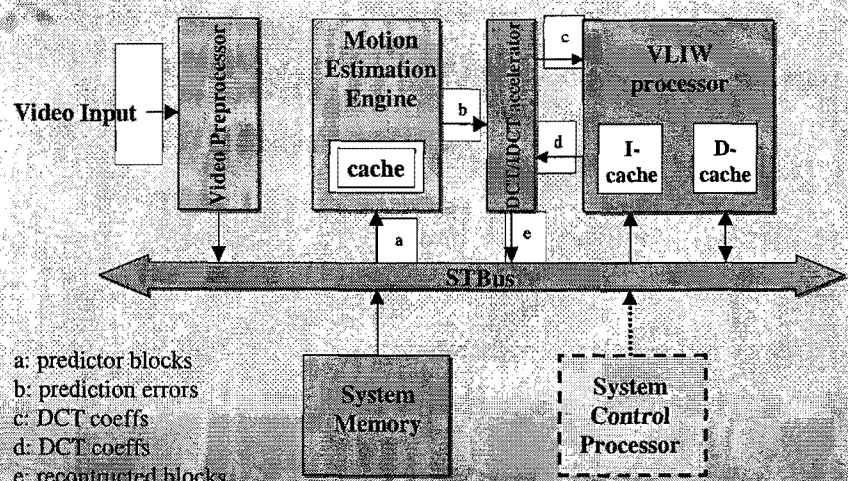
ST220 – main features

- 4 issue processor with 32 kByte I-cache and 32 kByte D-cache
- 64 32-bit registers
- 2 multipliers, 4 integer, 1 load/store, 1 branch
- Clock: 400MHz
- Size in 0.13μ technology:
  - core: 1.6 mm^2
  - caches: 3.2 mm^2

Streaming Data Interface

- Data ports for streaming media application
- Attached to load/store unit
- Direct access to/from video processing hardware
- Reduces SOC bus traffic
- Reduces cache pollution and control complexity
- 32 bits wide

Partitioning of Functions



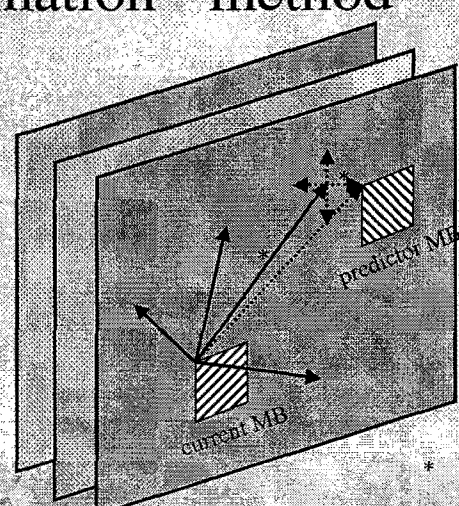
a: predictor blocks
b: prediction errors
c: DCT coeffs
d: DCT coeffs
e: reconstructed blocks

Motion Estimation - method

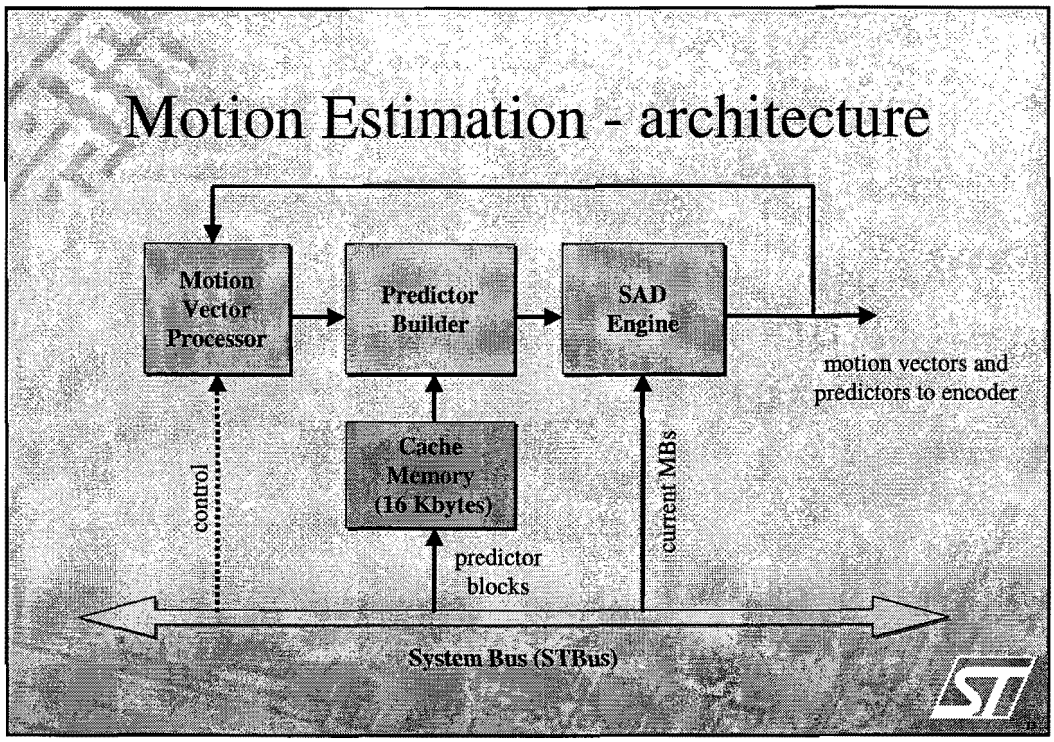
- A 2-step process

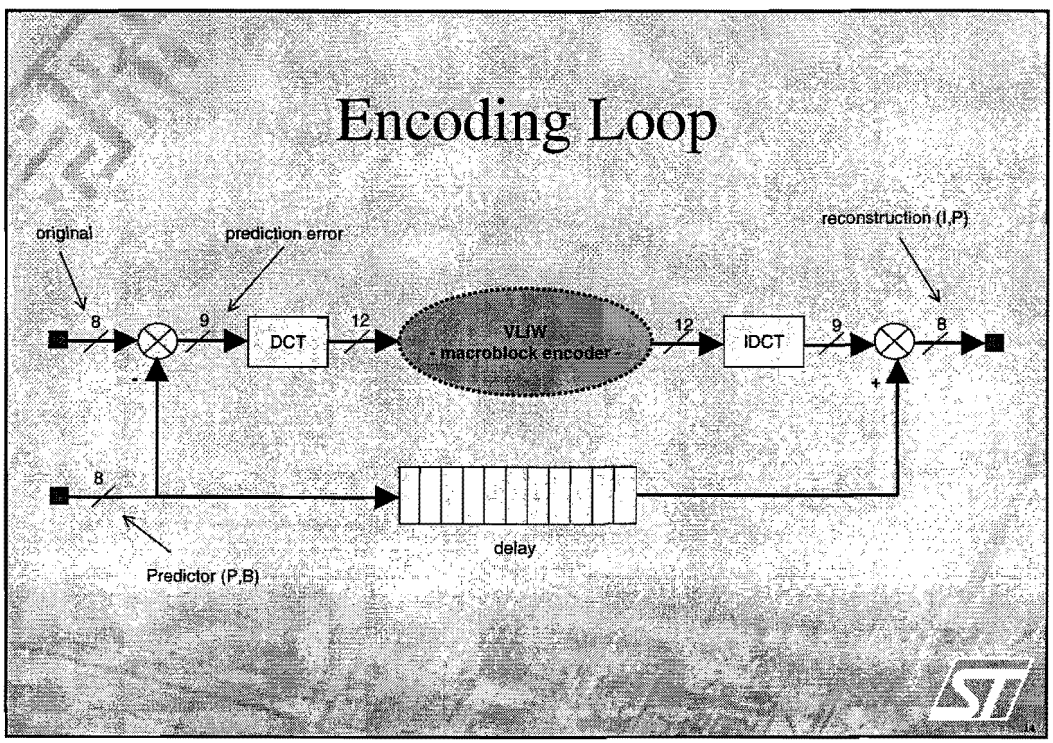
re-used motion vectors
"delta" updates
final motion vector

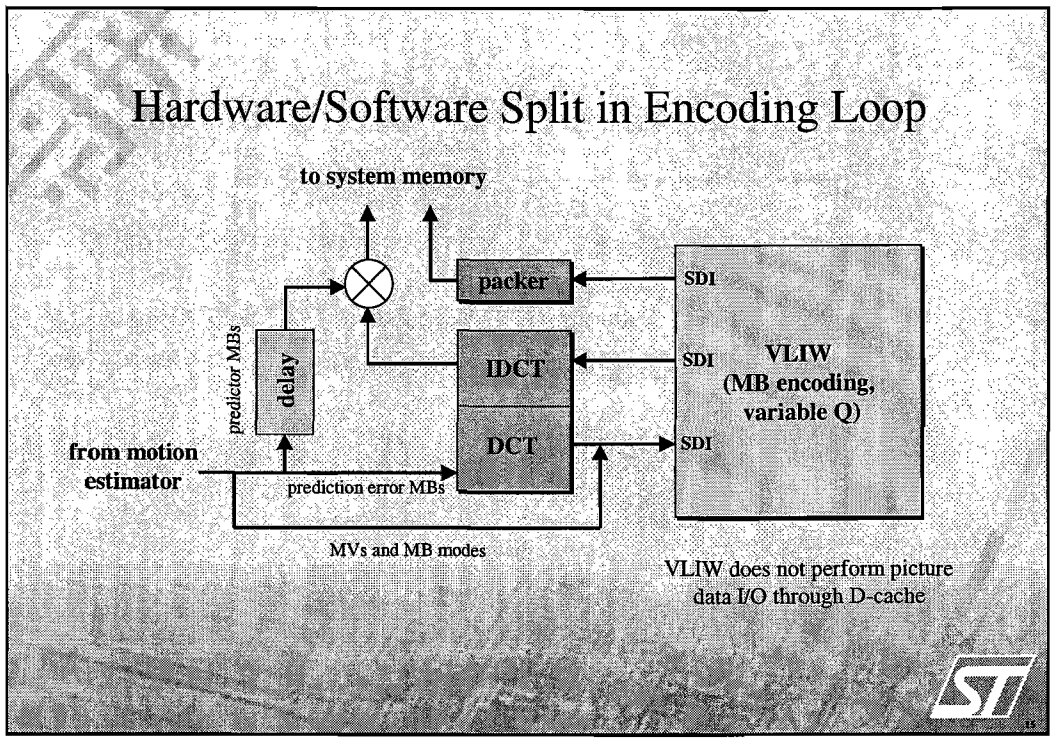
- Tracks true motion
- Complexity is search window-independent

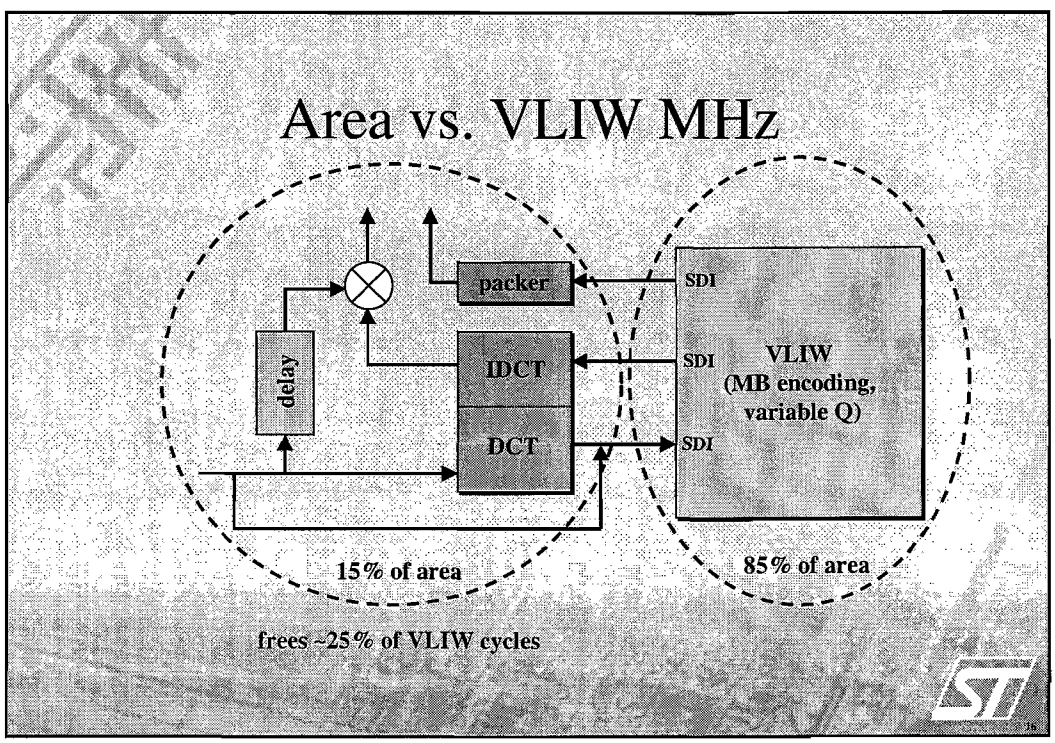


* winner


Motion Estimation - architecture
Motion Vector Processor
Predictor Builder
SAD Engine
motion vectors and predictors to encoder
Cache Memory (16 Kbytes)
control
current MBs
predictor blocks
System Bus (STBus)


Encoding Loop
original
prediction error
reconstruction (I,P)
8
9
DCT
12
VLIW
macroblock encoder
12
IDCT
9
8
-
+
8
delay
Predictor (P,B)


Hardware/Software Split in Encoding Loop
to system memory
packer
SDI
SDI
SDI
VLIW
(MB encoding,
variable Q)
IDCT
DCT
delay
predictor MBs
from motion
estimator
prediction error MBs
MVs and MB modes
VLIW does not perform picture
data I/O through D-cache


Area vs. VLIW MHz
packer
SDI
SDI
SDI
VLIW
(MB encoding,
variable Q)
IDCT
DCT
delay
15% of area
85% of area
frees ~25% of VLIW cycles

Audio/Video Encoding

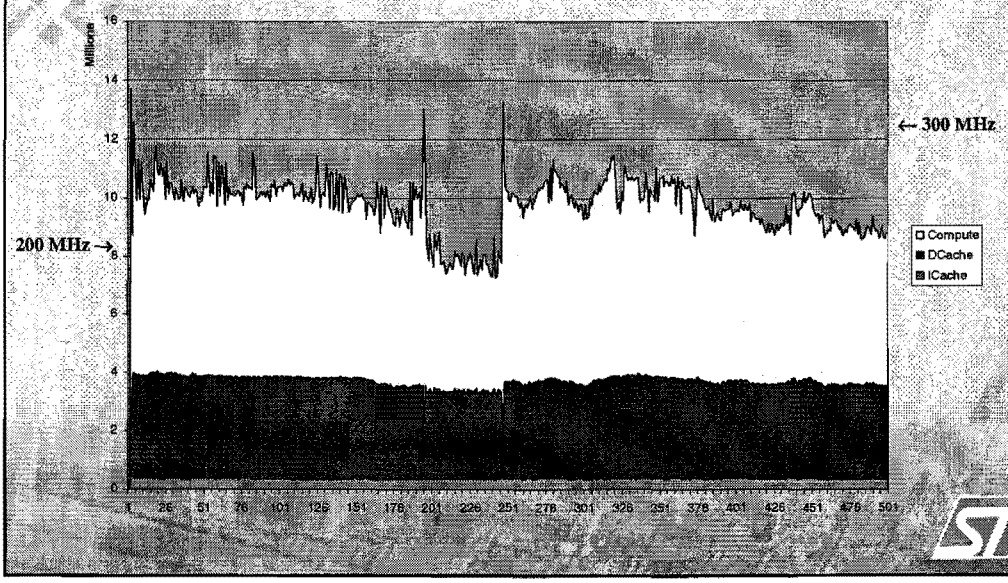
- MPEG-2 MP@ML video encoding consumes ~50% of available VLIW cycles
  - hence there is more than enough spare capacity for audio encoding

  ...with no increase in codec cell area
- Uses real-time kernel (OS21)

Use as a Decoder

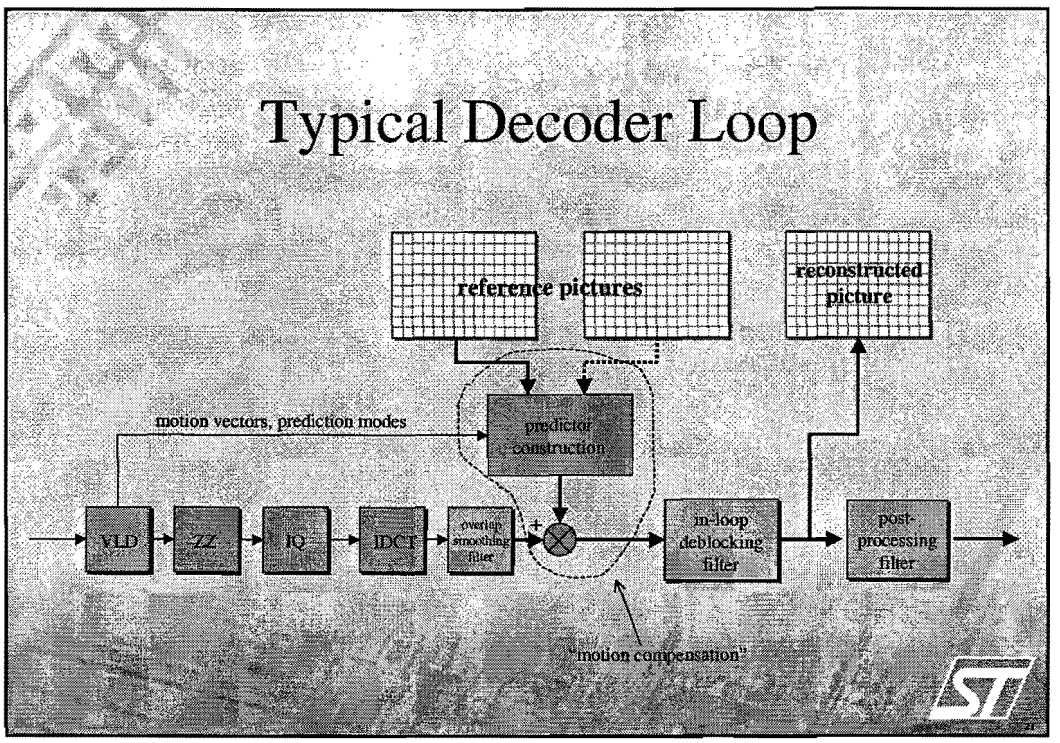
- VLIW processor is capable of real-time video decoding without acceleration
  - example: MPEG-4 (SP tools), D1 picture size, 1.5 Mbit/s requires 250 MHz in SOC environment
- MHz can be reduced by using motion estimation processor for motion compensation part of the decoder

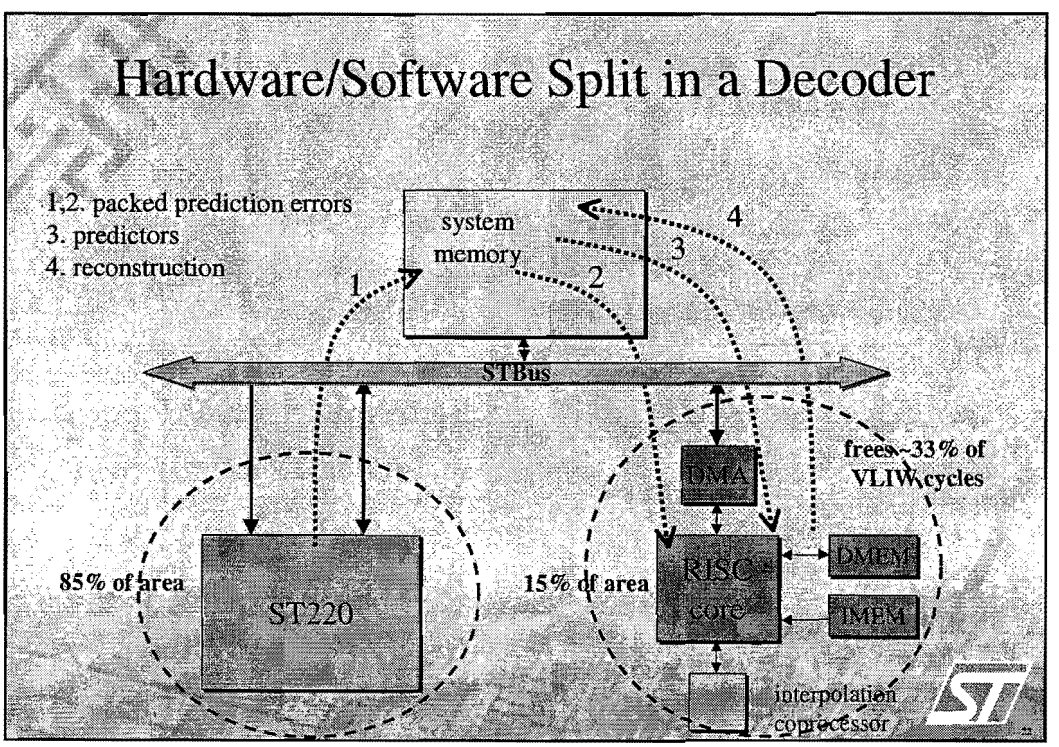
Processor loading for MPEG-4 decode



Accelerating Decode

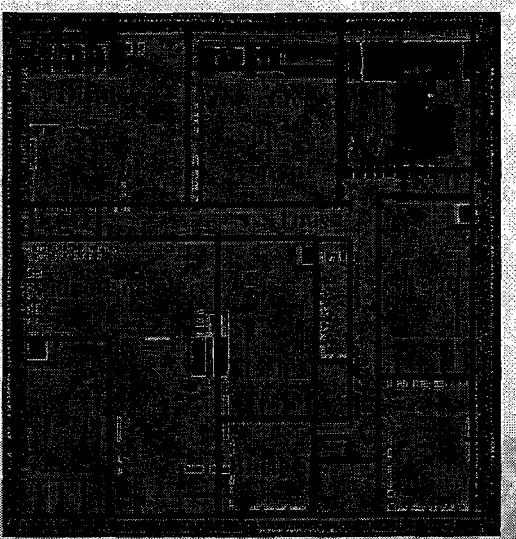
- Supporting multiple standards demands a flexible solution
- An implementation in a general-purpose processor(s) will not be economical except for the simpler requirements
- The challenge is to find the best hard/soft spilt
  - there are many possible solutions; here is one…


Typical Decoder Loop
reference pictures
reconstructed picture
motion vectors, prediction modes
predictor construction
VLD
ZZ
IQ
IDCT
overlap smoothing filter
+
in-loop deblocking filter
post-processing filter
"motion compensation"

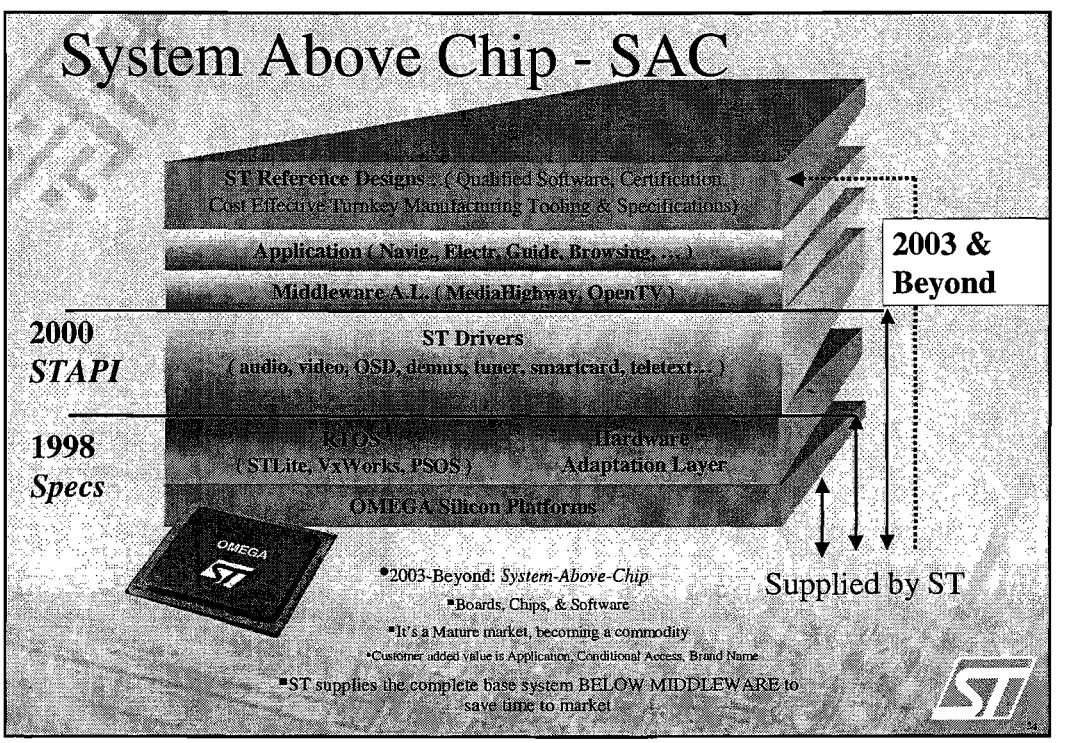

Hardware/Software Split in a Decoder
1,2. packed prediction errors
3. predictors
4. reconstruction
system memory
1
2
3
4
STBus
DMA
frees ~33% of VLIW cycles
85% of area
ST220
15% of area
RISC core
DMEM
IMEM
interpolation coprocessor

SOC Layout Example

- DVD recorder chip prototype
- 43 million transistors
- 3x ST200



System Above Chip - SAC



- 2003-Beyond: *System-Above-Chip*
  - Boards, Chips, & Software
  - It's a Mature market, becoming a commodity
  - Customer added value is Application, Conditional Access, Brand Name
  - ST supplies the complete base system BELOW MIDDLEWARE to save time to market

Conclusions

- We must reduce silicon cost, while increasing product flexibility
  - the challenge is to find ways of satisfying these contradictory requirements
- The key is a good balance between software and hardware
  - there is no single best solution to this

"SW development for embedded systems, based on components and contracts"

Prof.dr. Yolande Berbers

Computer Science Dept., Cath. University Leuven, Belgium

DistriNet Research Group

Software development for embedded systems based on components and contracts

Yolande Berbers
DistriNet
Department of Computer Science
KULeuven, Leuven, Belgium

embedded systems

DistriNet Research Group

Context and problem statement

- Embedded Systems
  - functionality and complexity: ever increasing (communication, multi-media, ...)
  - quality requirements are growing (ISO 900x)
  - time-to-market is getting ever smaller

The current software technology cannot always fulfill these higher requirements
- ⇦ too late
- ⇦ too expensive
- ⇦ too many errors

embedded systems

Context and problem statement

- Information systems
  - even bigger increase in functionality and complexity
  - comparable increase in quality requirements
  - focus on methodological aspects of software production
  - modern methodologies have proven benefits
    - clear increase of productivity and quality improvement

Our approach:
adaptation of successful methodologies for the specific requirements of embedded software:

Focus on non-functional constraints
(scarce resources)

embedded systems

Our approach

Innovative method for <u>component-based</u> software-development for Embedded Systems, supported by

- a notation for component modeling,
  with emphasis on resource constraints
  supported by a development environment
- a component architecture, methodology
  and run-time component system

General methodology: case driven
research is generic, conceptual basic research.
All developed software is to be considered as
proof-of-concept.

embedded systems

DistriNet Research Group

Overview

- Who we are
  - DistriNet
  - Cross-cutting issues
- SEESCOA methodology
- CCOM design tool
- Draco Middleware
- Case: camera surveillance
- Current and future work

embedded systems

DistriNet Research Group

DistriNet

- GOAL: development of open, distributed object support platforms for advanced applications, using state of the art software technology
  - always application driven
  - often conducted in close collaboration with industry
- focus on following application areas (-> task forces)
  - embedded and real-time applications
  - network services
  - internet applications
  - multi agent systems
  - security solutions
- 32 researchers

embedded systems

DistriNet Research Group

Cross-cutting issues

- *Software Reuse and Separation of Concerns:* development of system software that is structured in a modular way.
- *Conceptual models* that are dedicated to the development of particular kind of systems.
- Dynamic configuration and integration to build systems that are *dynamically customizable to application-specific needs,* possibly by integrating other components or services.
- *Quality of Service (QoS):* QoS architectures for embedded software systems, network protocols and middleware.

embedded systems

DistriNet Research Group

Overview

- Who we are
- SEESCOA methodology:
  - SEESCOA: Software Engineering for Embedded Systems using a Component-Oriented Approach
  - Core concepts
    - Components
    - Ports
    - Connectors
    - Contracts
- CCOM design tool
- Draco Middleware
- Case: camera surveillance
- Current and future work

embedded systems

DistriNet Research Group

SEESCOA Component

- Advantages of Components:
  Faster time-to-market (through reuse), Robustness, Maintainability, Reconfigurable (possibly at runtime)
- Challenges when used in embedded systems:
  - Reuse components in diff. HW/SW settings: not trivial
  - Non-functional behaviour of components
    - Resource usage, QoS (timing, bandwidth use, ...)
- SEESCOA Component:
  - Reusable, documented entity, used to build software systems
  - has particular function within a component system
  - are composed (glued) using interfaces specified on 4 levels
  - are loosely coupled to other components
- Component Blueprint vs. Component Instance
  - Blueprint: design time description
  - Instance: run-time instantiation of blueprint, can have state
- Built in-house, rather than COTS

embedded systems

DistriNet Research Group

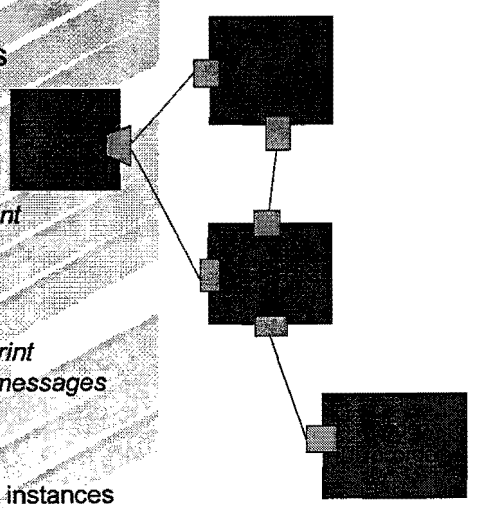
SEESCOA Component

- Interface specification: 4 levels
  - *Syntactic level:*
    - *Signature of operations (asynchronous, bi-directional)*
  - *Semantic level:*
    - *Specification of pre- and postconditions*
    - *Has not been formalized yet*
  - *Synchronization level:*
    - *Specification of the interaction protocol*
    - *Is done by means of extended MSC's*
  - *Quality of service level:*
    - *Specification of non-functional properties:*
      time complexity, memory usage, ...
      Through contracts

embedded systems

SEESCOA component, port, connector

- ❖ blueprints: static entities
- ❖ instances: dynamic entities



- Component
  - Component Instance
    - instantiation of a reusable *Component Blueprint*
    - exists at runtime
- Port
  - Port Instance
    - instantiation of a *Port Blueprint*
    - used for sending/receiving *messages*
    - has a specification
- Connector
  - Interconnects two or more port instances
  - Only compatible port instances can be connected
    - Their interfaces have to match on all levels

embedded systems

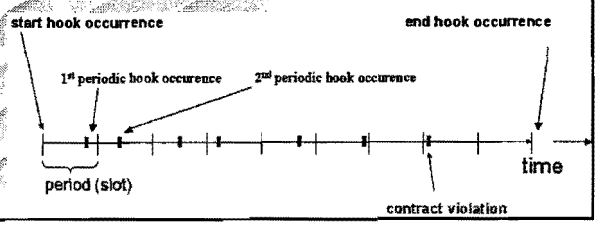
SEESCOA contracts

- Contract
  - imposes a (non-functional) constraint on the design
  - is attached to one or more participants
    - component instance
    - port instance
    - connector
  - currently the following types of contracts have been defined
    - Timing contracts
      - Deadline Timing contract: imposes a deadline on the occurrence of a particular event
      - Periodicity Timing contract: imposes periodicity constraint on occurrence of particular event
      - ✓ *Event* = sending, receiving or end of processing of message
      - ✓ Timing contracts are described using *hooks* on *extended MSC's*
    - Bandwidth contracts

embedded systems

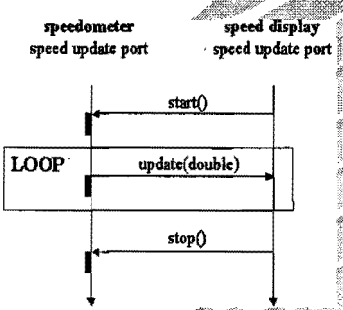
SEESCOA timing contracts

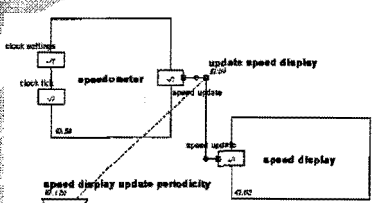
- A DeadlineTC is a template with **3** parameters:
  - **start hook occurrence**: contract becomes active
  - **end hook occurrence**: contract becomes inactive
  - **deadline**: maximum allowed time between end and start hook
- A PeriodicityTC is a template with **4** parameters
  - **start hook occurrence**: contract becomes active
  - **end hook occurrence**: contract becomes inactive
  - **periodic hook**: hook that has to occur periodically
  - **period**: period length. Every occurrence has to occur within associated period (not earlier, not later)



Periodicity Timing Contract

- Example: *speed is displayed at 2 Hertz*

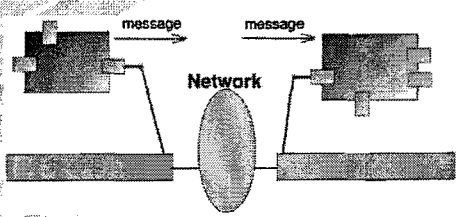




start hook occurrence: $start()_{send}[1]$
end hook occurrence: $stop()_{send}[1]$
periodic hook: $update()_{eca}$
period: 500

SEESCOA bandwidth contracts

- Why:
  - Distributed components consume bandwidth
  - Bandwidth feasibility should be checked at
    - design-time (component composition)
    - run-time (component deployment)
    - Run-time (while application is running)

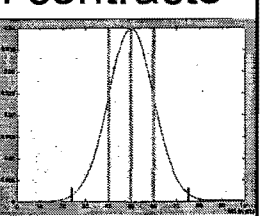


- Where:
  - On each component's ports
    - these produce messages

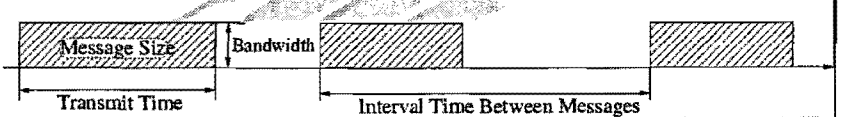
embedded systems

SEESCOA bandwidth contracts

- How
  - Statistical data-flow charact. (port's output)
  - Data-flow requirements (port's input)



- Data analysis
  - No low-level data (no packets, no time slots, ...)
  - Seen from the designer's point of view
    - Interval Time Between Messages (ITMB)
    - Message Size (MS)



systems

DistriNet Research Group

Overview

- Who we are
- SEESCOA methodology
- CCOM design tool for application building
  - Blueprint model
  - Instance model
  - Scenario model
  - Code generation
- Draco Middleware
- Case: camera surveillance
- Current and future work

embedded systems

DistriNet Research Group

Application Building

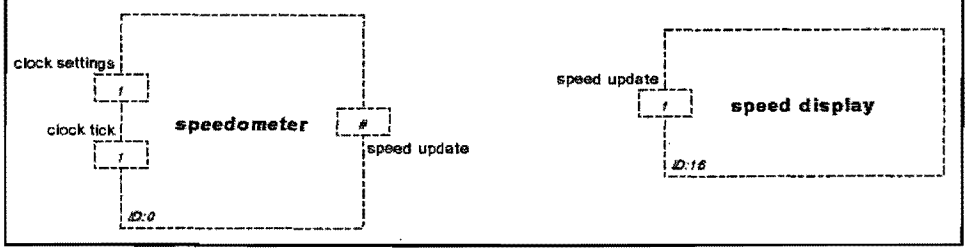
- three models for application building
  - ➢ adds extra views on the application
  - ➢ helps designer decompose design in coherent parts
  - ➢ makes use of the basic tool concepts
  - ▪ Blueprint Model
  - ▪ Instance Model
  - ▪ Scenario Model

embedded systems

DistriNet Research Group

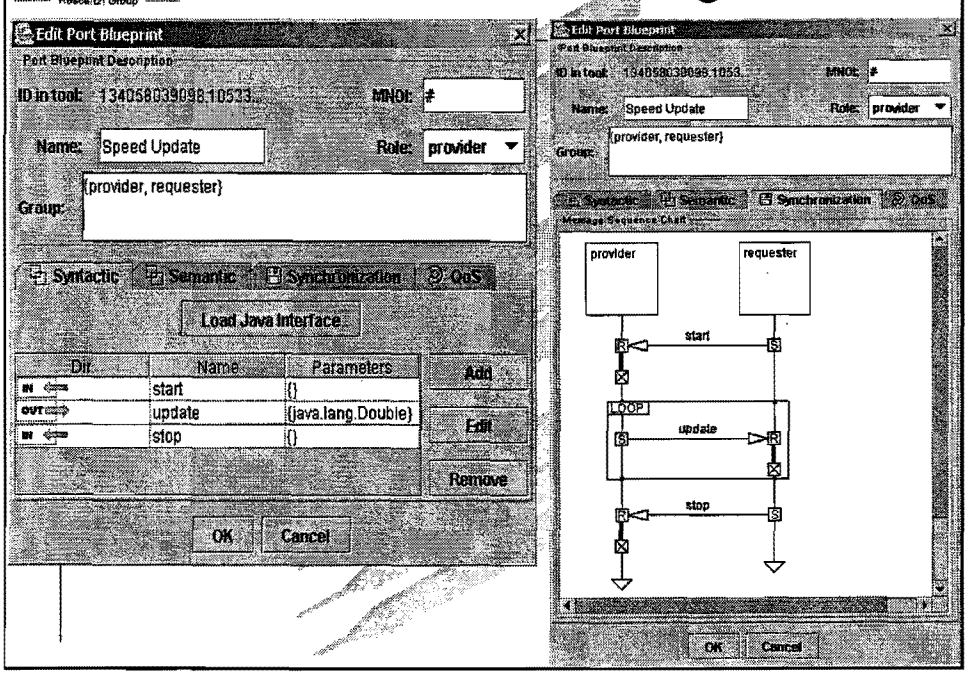
Blueprints

- Definition of a component blueprint
- Definition of a port blueprint
  - Attached to a component blueprint
  - Has an MNOI: 1 or more or undefined (#)
- Blueprint Model
  - **set of closely related component blueprints** that are **used** in (a part of) the application



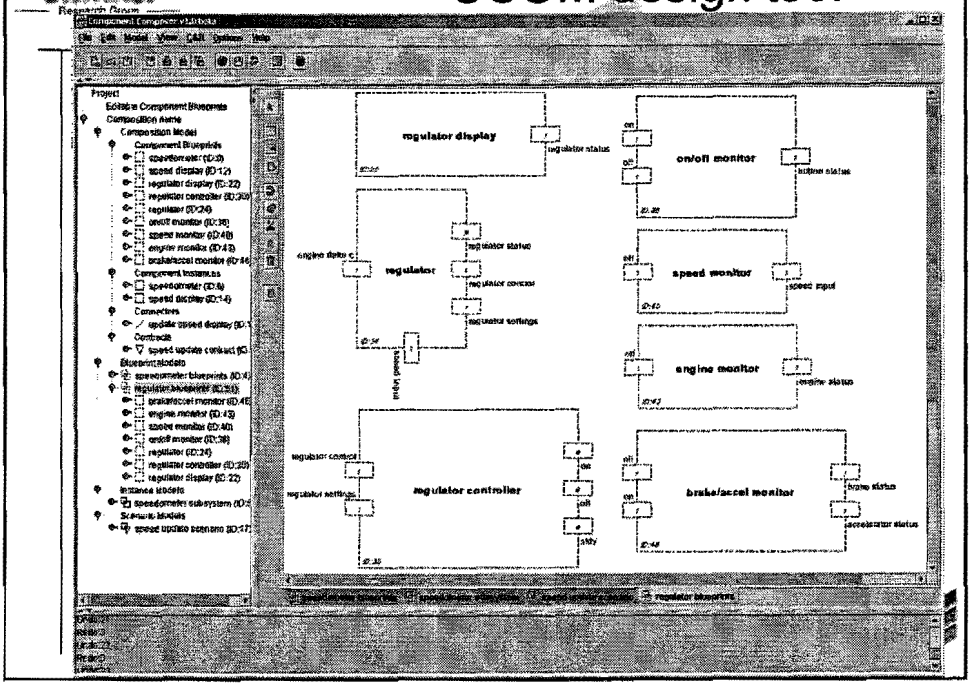
DistriNet Research Group

CCOM design tool



DistriNet Research Group

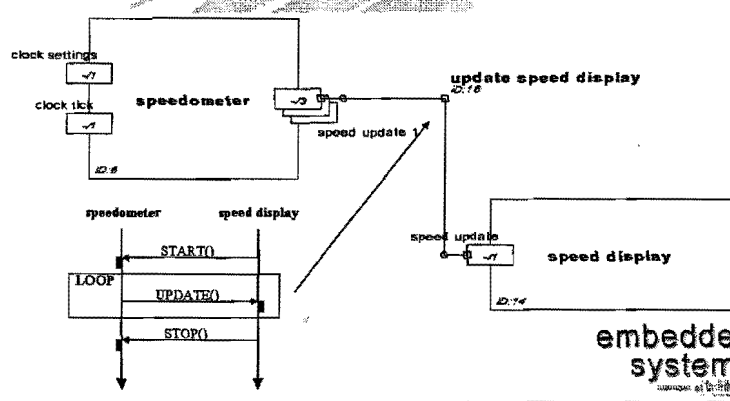
CCOM design tool



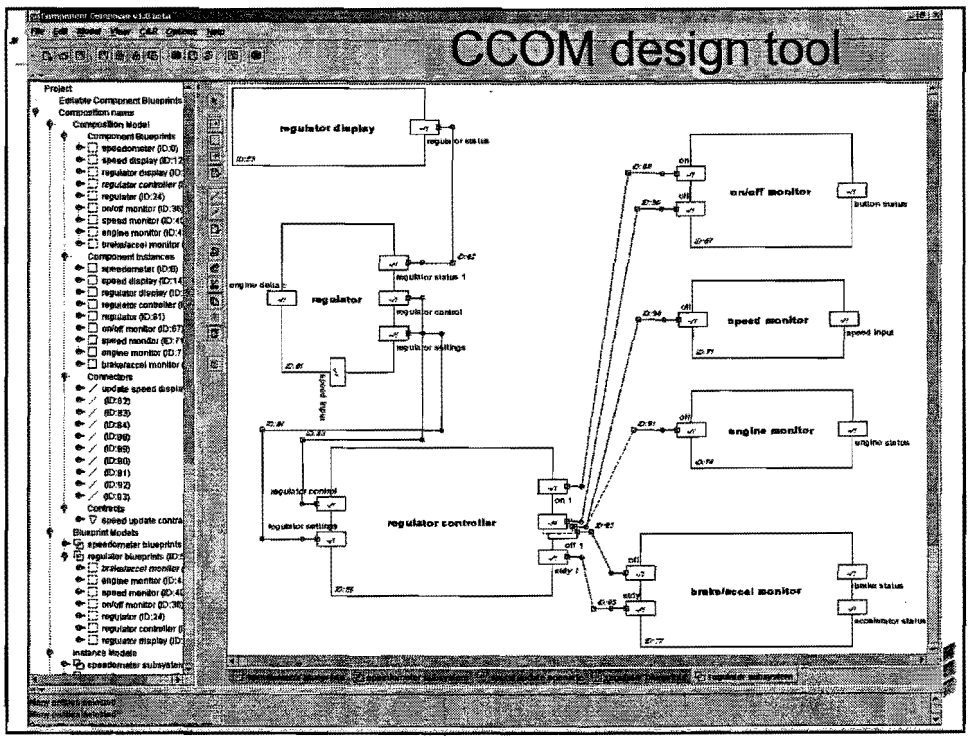
DistriNet Research Group

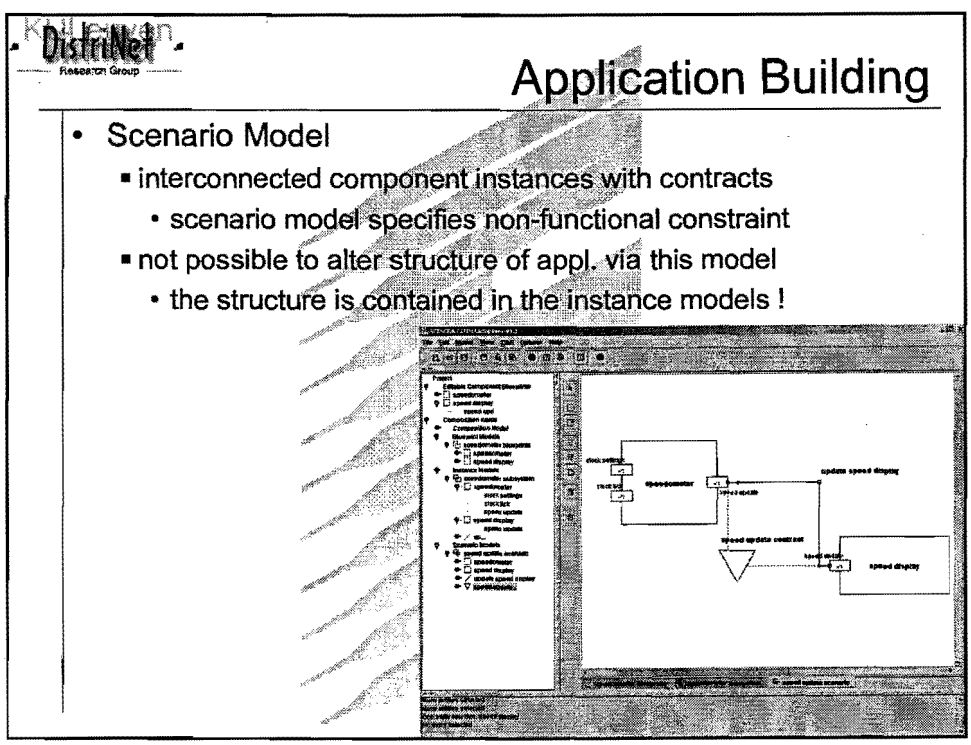
Application Building

- Instance Model: represents **runtime situation** of application
  - **Set of interconnected component instances**
  - Application can consist of more than 1 instance model
  - Component instance can occur in more than one model
  - Only compatible port instances can be connected
    - their interfaces have to match on all levels



embedded systems

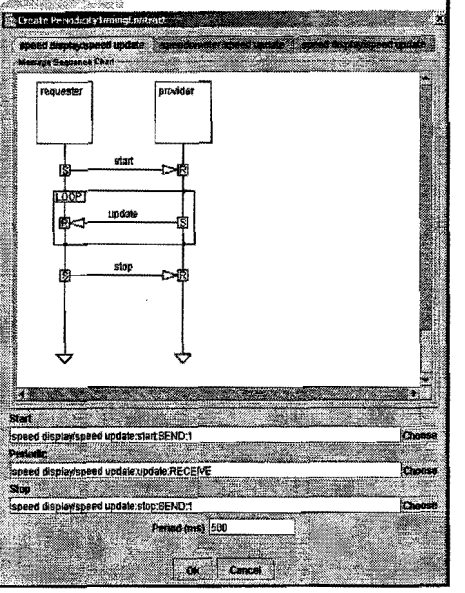

CCOM design tool
regulator display
regulator
regulator controller
on/off monitor
speed monitor
engine monitor
brake/accel monitor


DistriNet
Research Group
Application Building
• Scenario Model
▪ interconnected component instances with contracts
• scenario model specifies non-functional constraint
▪ not possible to alter structure of appl. via this model
• the structure is contained in the instance models !

DistriNet Research Group

CCOM design tool

- CCOM Scenario model
  - Contracts are defined
  - by filling in templates



DistriNet Research Group

CCOM Code generation

- CCOM generates skeleton (.comp file)
  - Programmer fills in methods with functionality
  - Syntax used is "seescoa component syntax"
    - Some new keywords for components, ports, messages, contracts, ...
- Our propocessor transforms .comp in .java files
- Eventually, every component is one .jar file
- .jar file read by Draco who creates component

embedded systems

DistriNet Research Group

Overview

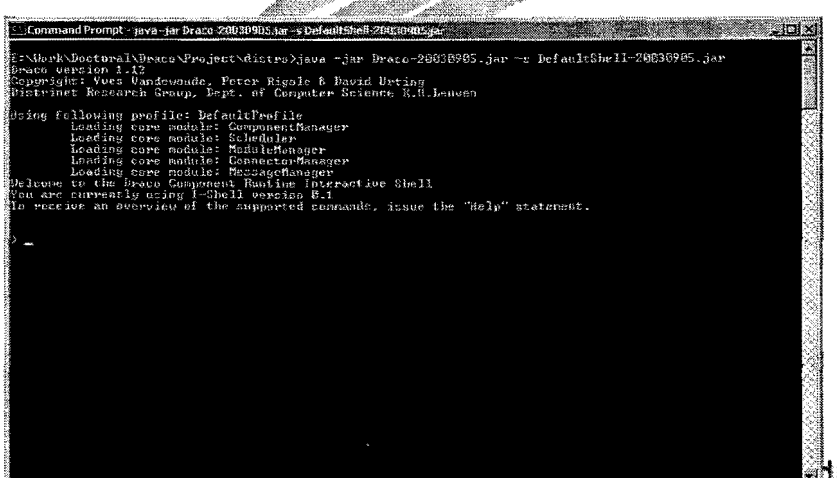
- Who we are
- SEESCOA methodology
- CCOM design tool
- Draco Middleware:
  - Some screen shots
  - Draco architectuur
    - Core Modules, Messageflow, Extension Modules
  - extension modules
    - distribution, contract monitoring, live updates
- Case: camera surveillance
- Current and future work

embedded systems

DistriNet Research Group

At Runtime

- Start Draco



DistriNet Research Group

At Runtime

- Load Components

```
Command Prompt - java -jar Draco-20030905.jar -s DefaultShell-20030905.jar

E:\Work\Doctoral\Draco\Project\distro>java -jar Draco-20030905.jar -s DefaultShell-20030905.jar
Draco version 1.12
Copyright: Yves Vandewoude, Peter Rigole & David Urting
Distrinet Research Group, Dept. of Computer Science K.U.Leuven

Using following profile: DefaultProfile
        Loading core module: ComponentManager
        Loading core module: Scheduler
        Loading core module: ModuleManager
        Loading core module: ConnectorManager
        Loading core module: MessageManager
Welcome to the Draco Component Runtime Interactive Shell
You are currently using I-Shell version 0.1
To receive an overview of the supported commands, issue the "Help" statement.

> Load component camera from CameraComponent.jar

Componentinstance camera created.
This component has following ports:
MultiCastPorts:
        camera/Out      (maximum 10 simultanious connections)
PortGroups:
        camera/Control  (maximum 1 instances)

> Load component display from ImageViewer.jar

Componentinstance display created.
This component has following ports:
MultiCastPorts:
PortGroups:
        display/Input   (maximum 1 instances)
        display/Control (maximum 1 instances)

> _
```

DistriNet Research Group

At Runtime

- Connect Components

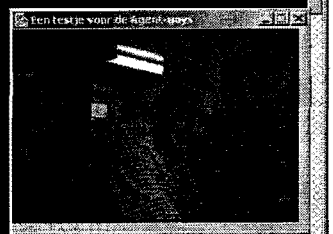
```
Command Prompt - java -jar Draco-20030905.jar -s DefaultShell-20030905.jar
        Loading core module: ModuleManager
        Loading core module: ConnectorManager
        Loading core module: MessageManager
Welcome to the Draco Component Runtime Interactive Shell
You are currently using I-Shell version 0.1
To receive an overview of the supported commands, issue the "Help" statement.

> Load component camera from CameraComponent.jar

Componentinstance camera created.
This component has following ports:
MultiCastPorts:
        camera/Out      (maximum 10 simultanious connections)
PortGroups:
        camera/Control  (maximum 1 instances)

> Load component display from ImageViewer.jar

Componentinstance display created.
This component has following ports:
MultiCastPorts:
PortGroups:
        display/Input   (maximum 1 instances)
        display/Control (maximum 1 instances)

> Connect camera/Out with display/Input

Ports successfully connected!

> Connect display/Control with Shell

Ports successfully connected!

> Connect camera/Control with Shell

Ports successfully connected!

>
```

DistriNet Research Group

At Runtime

- Start Components

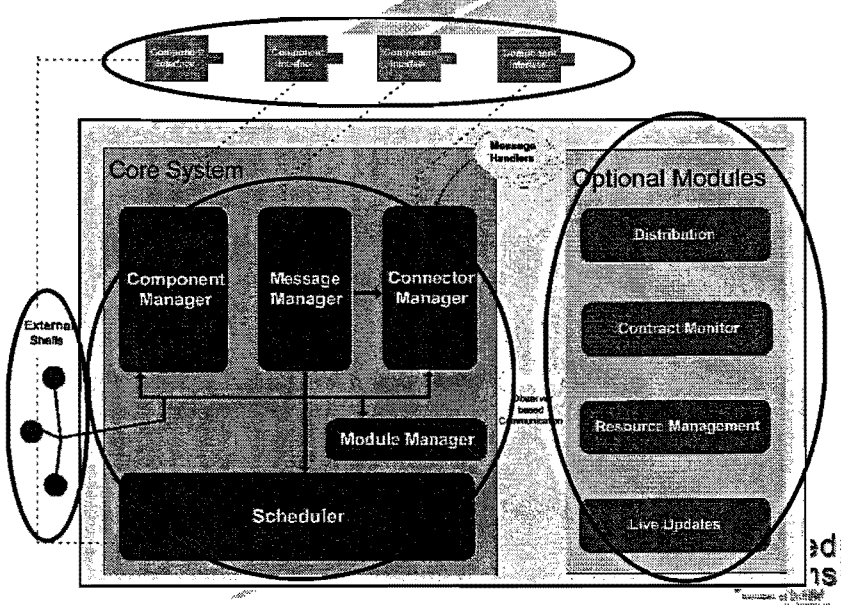
```
Component instance camera created.
This component has following ports:
MultiCastPorts:
        camera/Out      (maximum 10 simultanious connections)
PortGroups:
        camera/Control  (maximum 1 instances)

> Load component display from ImageViewer.jar

Component instance display created.
This component has following ports:
MultiCastPorts:
PortGroups:
        display/Input   (maximum 1 instances)
        display/Control (maximum 1 instances)

> Connect camera/Out with display/Input

Ports successfully connected!

> Connect display/Control with Shell

Ports successfully connected!

> Connect camera/Control with Shell

Ports successfully connected!

> Send Start[width:320|height:240|title:Een testje voor de Agent-guys] to display/Control

> Send Configure[resolution:320x240|id:0] to camera/Control

> Send StartDelay:0] to camera/Control

Delay set to: 0
```



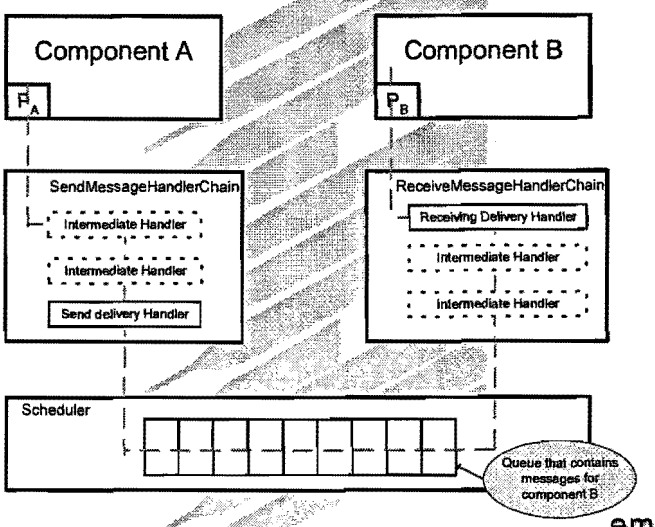
systems

DistriNet Research Group

Draco Architectuur



Draco Message Delivery: Concept



Draco Extension Modules

- Extension modules allow for new functionality
  - Distribution
  - Contract negotiation
  - Dynamic Updating
- Can be loaded (en removed) dynamically
- Hook in in Draco through
  - Message Handlers that intercept messages
  - Register themselves with core modules
    - via Publish-Subscribe mechanism
  - No Control mechanism:
    - a module has fully access to internals of Draco
    - each module is trusted

Draco Timing Contract Monitor

- Monitor consists of 3 subsystems
  - Event collection/timestamping
  - Contract monitoring
  - Violation reporting
- Is an optional part of Draco
- Monitors deadline and periodicity constraints
- New monitor types can be added easily
- Contract violation
  - Offline report
  - Online violation feedback
    - To application
    - To middleware (e.g. scheduler)

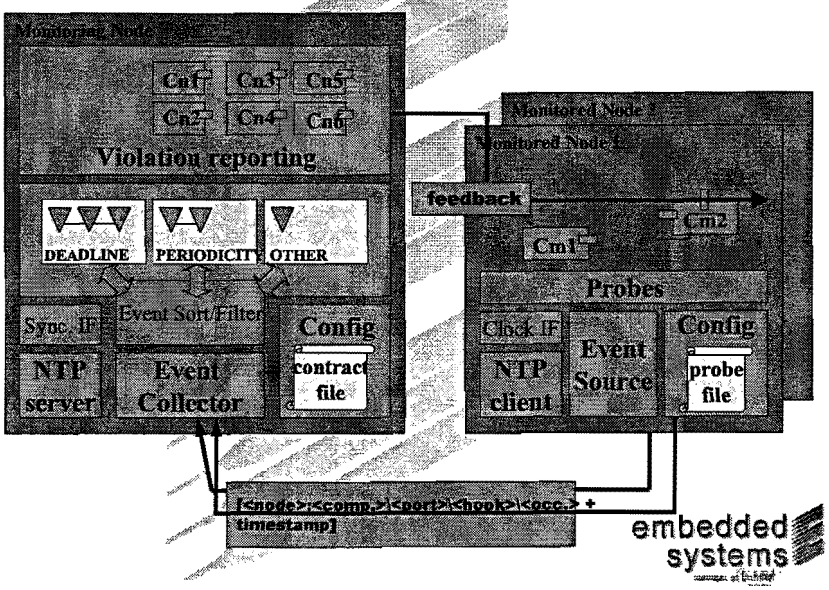
embedded systems

Draco Contract Monitor

- Distributed Monitoring Challenges
  - Managing resource utilization
    - Extra overhead due to exchange of timing messages
  - Clock synchronization on various nodes for correct timestamps
- Draco Approach
  - Monitoring functionality split in
    - Monitoring Node (1): contract monitoring, violation report
    - Monitored Nodes (n): event collection / timestamping
  - Clock synchronization: NTP (Network Time Protocol)
  - Uses contract file (on monitoring node)
  - Uses probe files (on monitored nodes)
  - Violations made explicit through feedback ports (on contracts)

embedded systems

Draco Contract Monitor



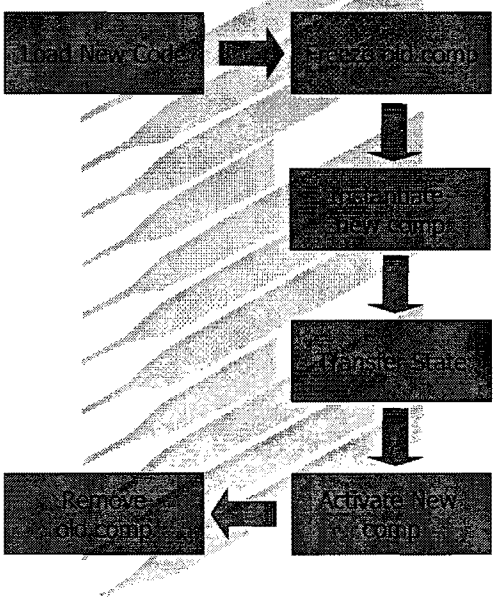
embedded systems

Draco Live updates

- What is live update ?
  - Changing (part of) code of an application
  - while application is running
- Why live updates ?
  - Fixing bugs
  - Adding features or functionality
  - Adaptability of devices to environment
    - *Communication (establishing new protocol)*
    - *Peripherals*
    - ...

embedded systems

Live updates: steps in updating



embedded systems

Overview

- Who we are
- SEESCOA methodology
- CCOM design tool
- Draco Middleware
- Case: camera surveillance
  - Functionality
  - Reuse
- Current and future work

embedded systems

DistriNet Research Group

Test case: camera surveillance

- Functionality
  - Distributed surveillance
  - Motion detection
  - Recording and retrieving images
  - Intelligent zooming
  - Platform independent user interface visualization
- Objectives of test case
  - Proof of concept for
    - design methodology and CCOM tool
    - Draco runtime system with real life example
  - Cross domain showcase example
    - communication
    - signal processing
    - imaging

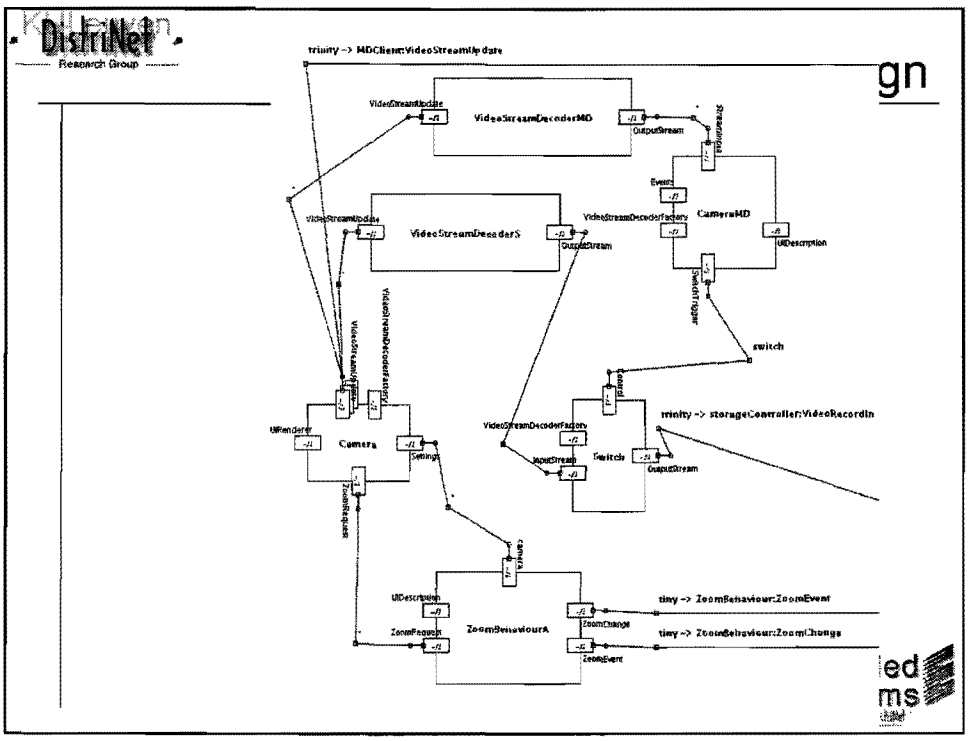
embedded systems

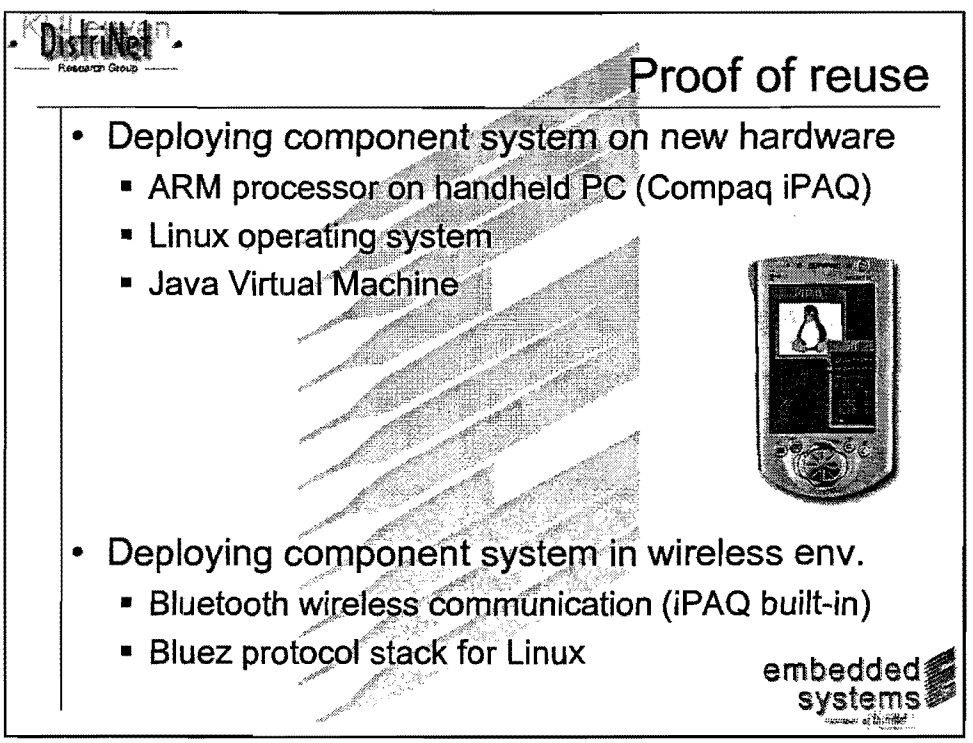
DistriNet Research Group

Test case: camera surveillance

- Hardware platform: Camera with local intelligence
  - Sony DFW-VL500 with FireWire
  - PC/104 with
    - National Semiconductor MediaGX @ 200MHz
    - 32MB RAM
    - 16MB Flash
- Software platform:
  - Linux
  - Java

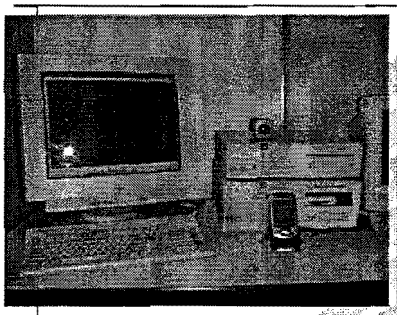
embedded systems

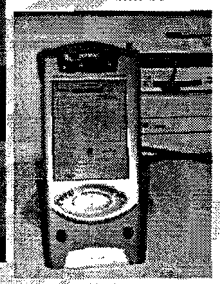

DistriNet
Research Group
gn
VideoStreamDecoderMD
VideoStreamDecoderS
CameraMD
Camera
Switch
ZoomBehaviourA
ed
ms

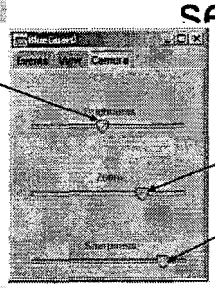

DistriNet
Research Group
Proof of reuse
• Deploying component system on new hardware
▪ ARM processor on handheld PC (Compaq iPAQ)
▪ Linux operating system
▪ Java Virtual Machine
• Deploying component system in wireless env.
▪ Bluetooth wireless communication (iPAQ built-in)
▪ Bluez protocol stack for Linux
embedded
systems

DistriNet Research Group

setup



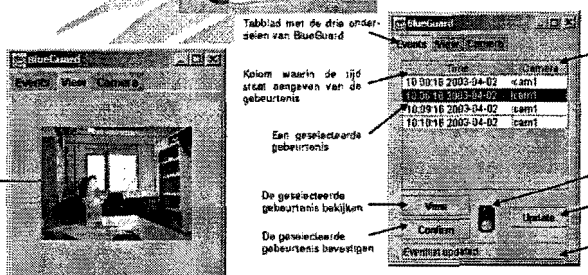




De regelaar voor de helderheid

De regelaar voor de zoom instelling

De regelaar voor de scherpstelling



Tabblad met de drie onderdelen van BlueGuard

Kolom waarin de tijd staat aangegeven van de gebeurtenis

Kolom waarin de camera staat aangegeven waarop de gebeurtenis is geregistreerd

Een geselecteerde gebeurtenis

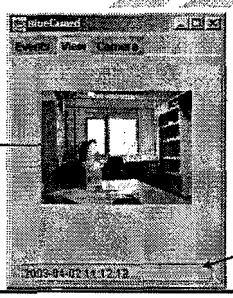
De status van de bluetooth verbinding

De geselecteerde gebeurtenis bekijken

Actualiseren van de gebeurtenissen lijst

De geselecteerde gebeurtenis bevestigen

Tekstvlak met status van de BlueGuard applicatie



Een cameraopname van de laatste geselecteerde gebeurtenis

De datum van de getoonde cameraopname

systems

DistriNet Research Group

Overview

- Who we are
- SEESCOA methodology
- CCOM design tool
- Draco Middleware
- Case: camera surveillance
- Current and future work

embedded systems

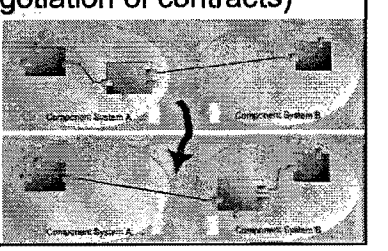
Current and future work

- Contracts
  - New types of contracts: memory
    - Both definition and monitoring
    - Can be attached to components and to connectors
  - Monitoring of Bandwidth contracts
- Run-time evolution
  - State transfer
    - Related to other research domains
      - Database scheme evolution
      - Process migration & distributed computing
  - Concurrent updates

embedded systems

Current and future work

- Adapting SEESCOA and Draco for Aml
  - CoDAMoS project
  - Environment where services are downloaded
    - Very dynamic environment
  - Context plays important role
  - Violation of contracts reported to application
- Managing layer for applications
  - Take actions
    - When services are loaded (negotiation of contracts)
    - When contracts are violated
    - E.g. bandwidth contract
      - Possible relocation of comp.
      - Possible renegotiation of contracts



DistriNet
Research Group

Thank you !

Questions ?

Yolande.Berbers@cs.kuleuven.ac.be

embedded systems

"Multimedia Home Platform SW architecture"

Ir. Erik Moll Philips

Philips Digital Systems Labs, Eindhoven, NL

"Putting Java™ in your TV"

Erik Moll
December 2003

erik.moll@philips.com

PHILIPS

Contents

- Short overview of the Multimedia Home Platform
- The Philips MHP SW Architecture
- Robustness problems and solutions
- Q&A

A 15 minute overview of MHP

Erik Moll
December 2003

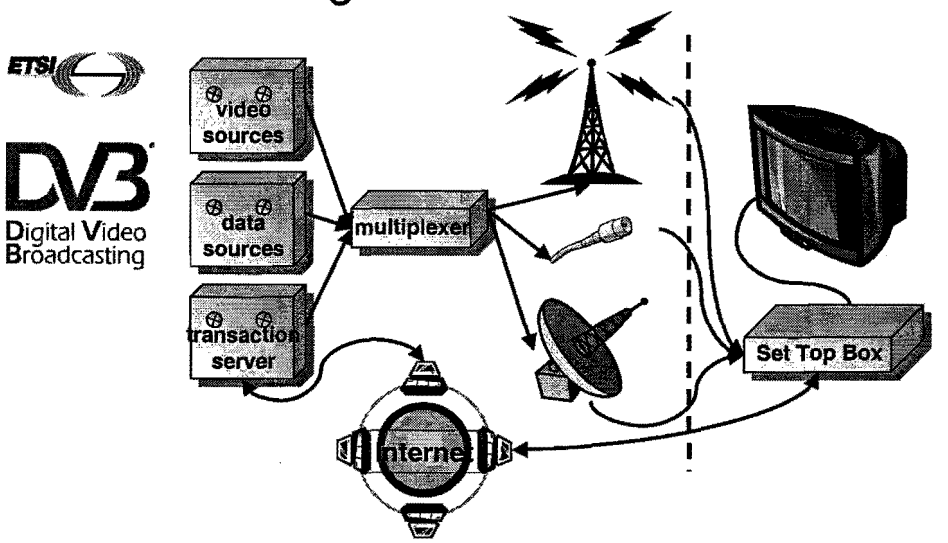
PHILIPS

The Multimedia Home Platform (DVB-MHP)

- A European standard for interactive digital television
  - Based on DVB and Java
  - An open system in the living room
  - Deployed in a growing number of EU and other countries
- Used as basis for US (OCAP) and Japanese (ARIB) equivalents
  - The GEM (Globally Executable MHP) defines the common core.

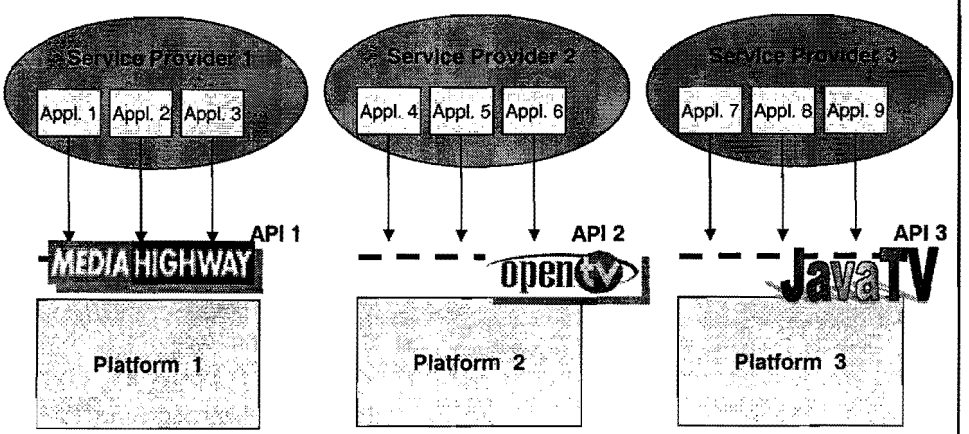
PHILIPS

DVB-MHP Digital TV Context



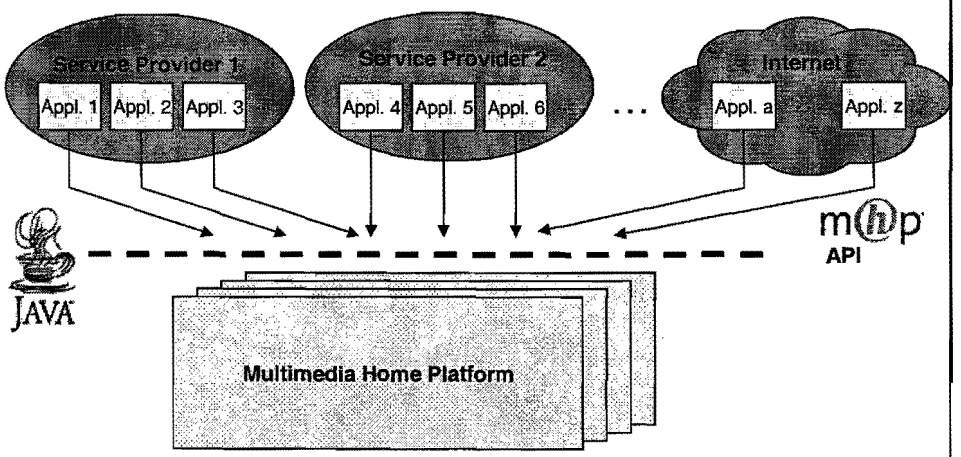
PHILIPS

Today's "Vertical" Markets



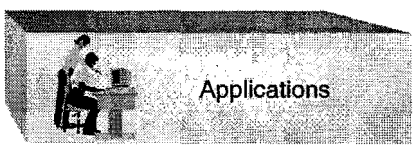
PHILIPS

Future "Horizontal" Markets

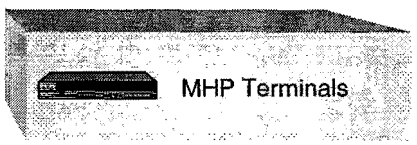


PHILIPS

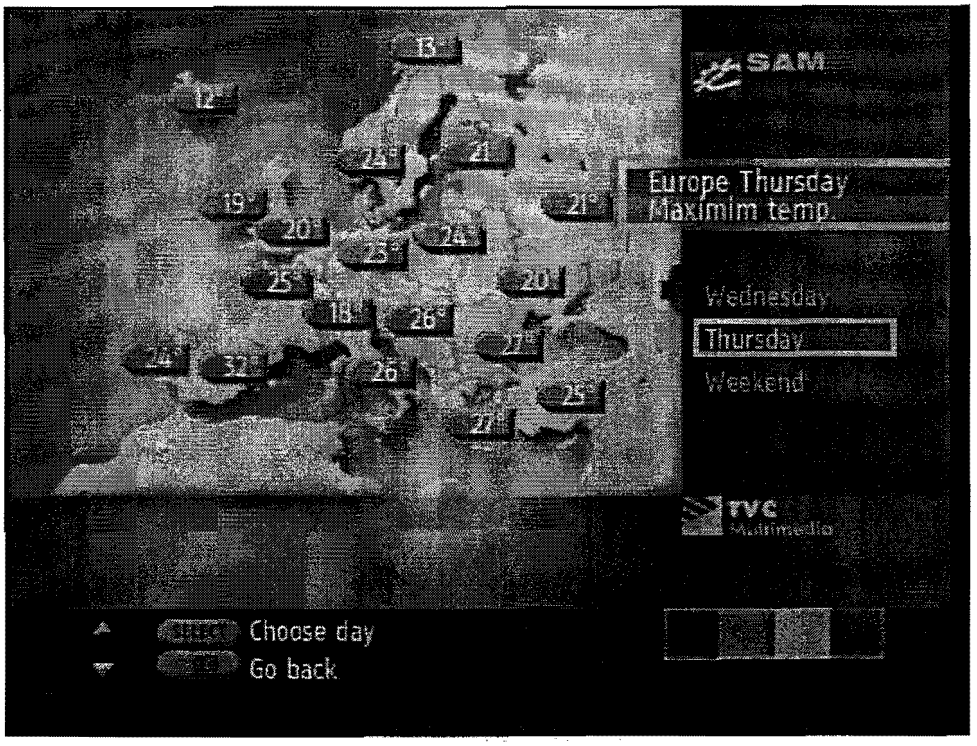
Scope of the DVB MHP



- Independent developers
- Different service providers
- Various application areas

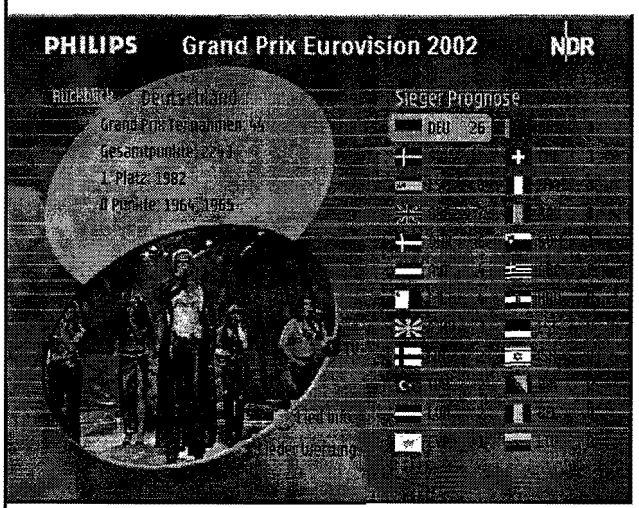


- Independent implementations
- Different hardware
- Different software
- All kind of terminals
  low-end STB and high-end PC



PHILIPS

Sample MHP Application Eurovision Song-Contest 2002



- Application developed for NDR.
- Features:
  - Winner prediction
  - Song rating
  - Background info
- Uses custom designed server-side module on top of generic voting module.

PHILIPS

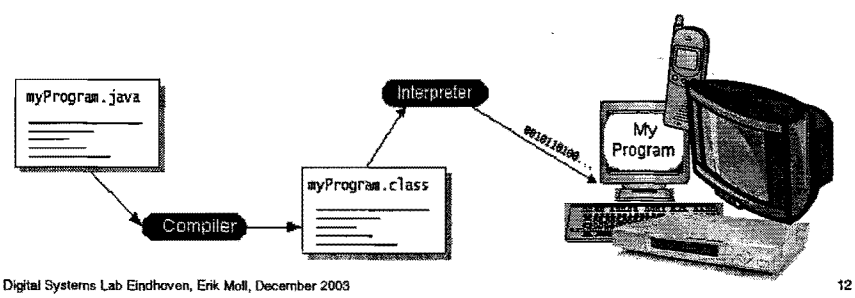
MHP Specification Contents

- MHP is not just Java & Java APIs
  - It defines all the bits in the TV signal for the box to receive
- Non-API contents include:
  - Transport protocols - broadcast & return channel
  - Content formats - video, audio, images, fonts, ...
  - Application model - rules for starting & stopping applications
  - Application signalling - support for application model rules
  - Security model - how to authenticate applications
  - Graphics reference model - how graphics & video go together
  - System integration - glue holding it all together
  - Minimum receiver requirements - remote control, graphics, ..
  - Profiles

PHILIPS

Why Java and Not Another VM?

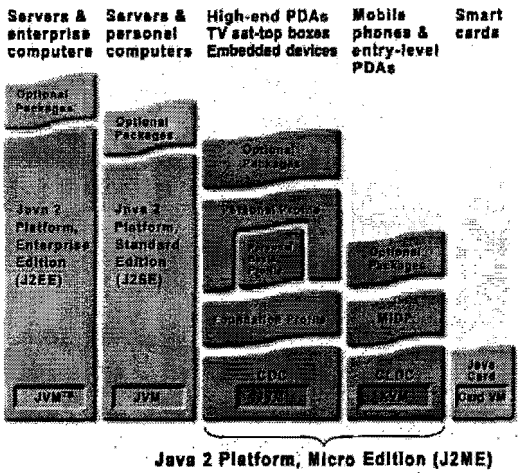
- The only "open" standard for a virtual machine
- Security build in from the start
- Strong conformance & compliance regime



PHILIPS

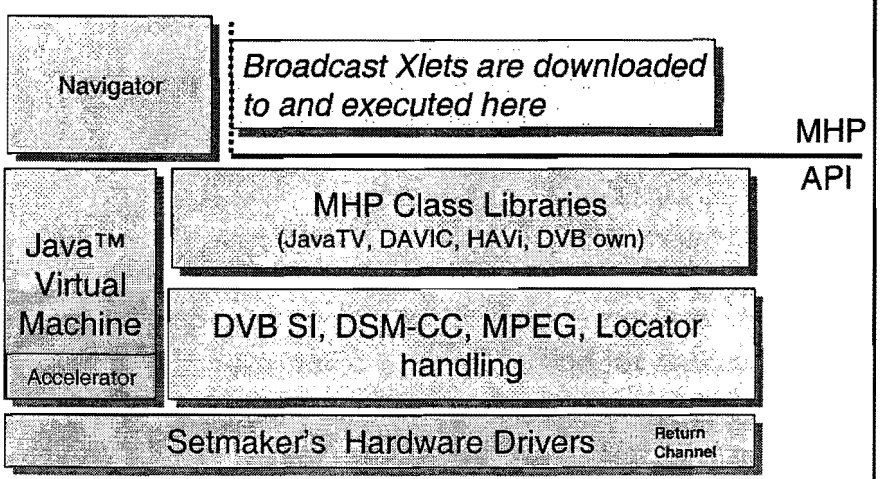
MHP Java profile

- MHP is based on Personal Java
- Personal Java is a predecessor of the Personal Profile
- MHP excludes full AWT from Personal Java (and some more)



PHILIPS

MHP Main System Components



PHILIPS

The main MHP Java APIs

- "Core" APIs
  - PersonalJava 1.1, AWT (but no PC-oriented widget set)
- JavaTV APIs
  - Java Media Framework (JMF), Service selection, Xlet, Protocol independent Service Information (SI)
- DAVIC APIs
  - MPEG-2 section filter, tuning, conditional access, resource notification
- HAVi APIs
  - TV-specific widget set and other TV-specific extensions
- DVB APIs
  - object carousel (broadcasted file-system), DVB specific SI, Application discovery & launching, User preferences

PHILIPS

The MHP Navigator

- Term used for manufacturer resident UI in an MHP
  - Intentionally not controlled by MHP specification
- Includes some form of channel "zapper"
  - Maintain list of available services
  - Present list of available services to end user
  - Expected to handle P+/P- keys which never go to MHP apps
- Provide access to resident and downloaded applications
- Includes set-up UI for the receiver
  - Possibly initiate channel scans
  - Configure (telephone/modem) return channel, ISP details

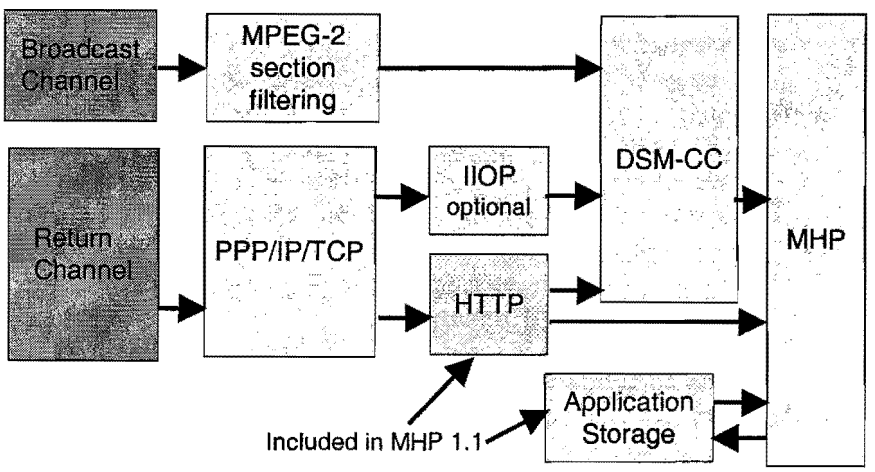
PHILIPS

MHP Applications

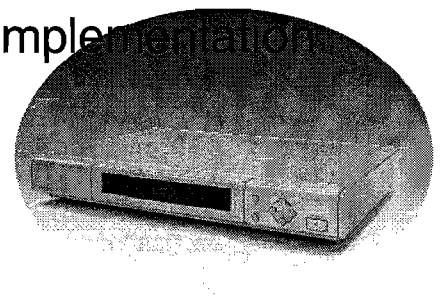
- MHP defines DVB-J Applications, also known as "Xlets"
  - Actually they implement the Xlet interface (javax.tv.xlet)
- Xlets can be loaded in multiple ways
  - From an MPEG-2 transport-stream
  - From an HTTP return channel
  - From storage (Flash memory)
- Xlet lifecycles are controlled in multiple ways
  - By application signalling through the Application Information Table (AIT) in the MPEG-2 stream,
  - By user interaction through the Navigator
  - By another application
- Xlets are all started and stopped by an application manager
  - A part of the runtime system that controls all applications

PHILIPS

Application Downloading "Channels"



The Philips MHP Implementation



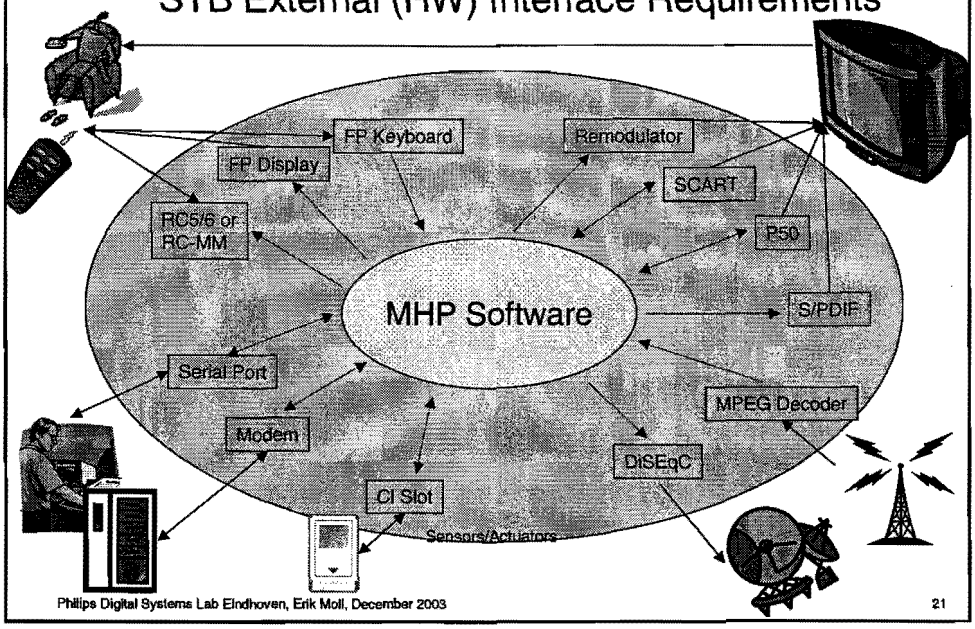
PHILIPS

Top-level Requirements MHP SW stack

- MHP Conformant
  - Implement MHP 1.0.X in a **conformant** way
  - Pass all 10.000+ MHP conformance tests with our code-base
- Portable
  - Support Philips Semiconductors platform, ST Microelectronic DVB reference designs and Linux/Intel PC with DVB PCI-card
  - Based on a well-defined Host Porting Interface (HPI)
- Customizable
  - A custom UI can be build on top of a high-level "Resident Applications API" (RA-API)
  - A lot of diversity had to be dealt with (front-en / modulation type, TV or STB, modem or not, memory sizes, language sets)
- Robust
  - Consumer Electronics products are NOT PCs

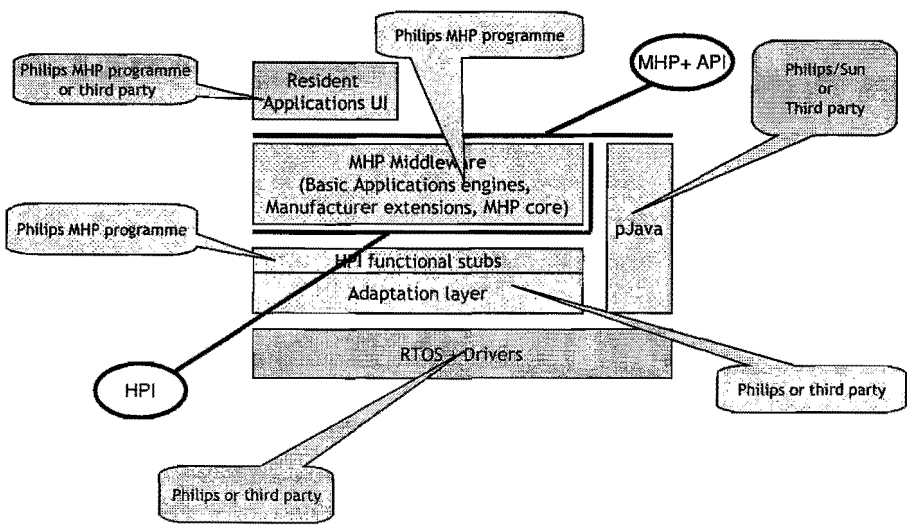
PHILIPS

STB External (HW) Interface Requirements



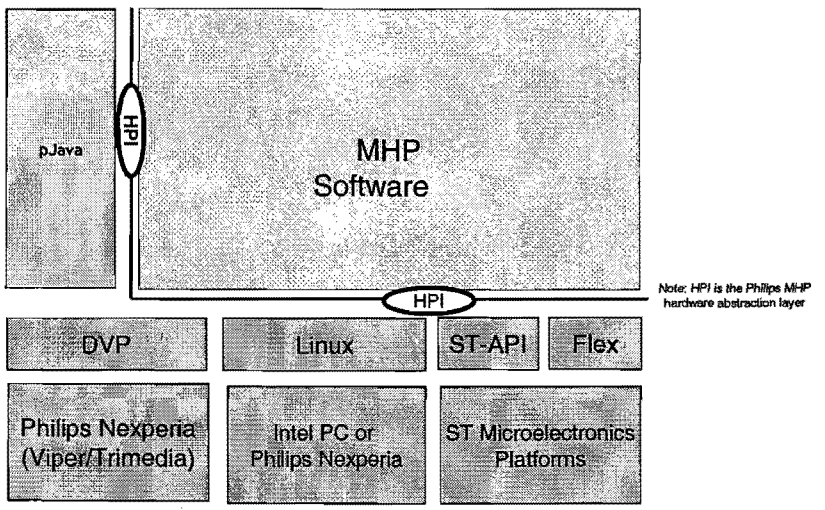
PHILIPS

General structure – MHP SW product



PHILIPS

Portable MHP stack – supported platforms



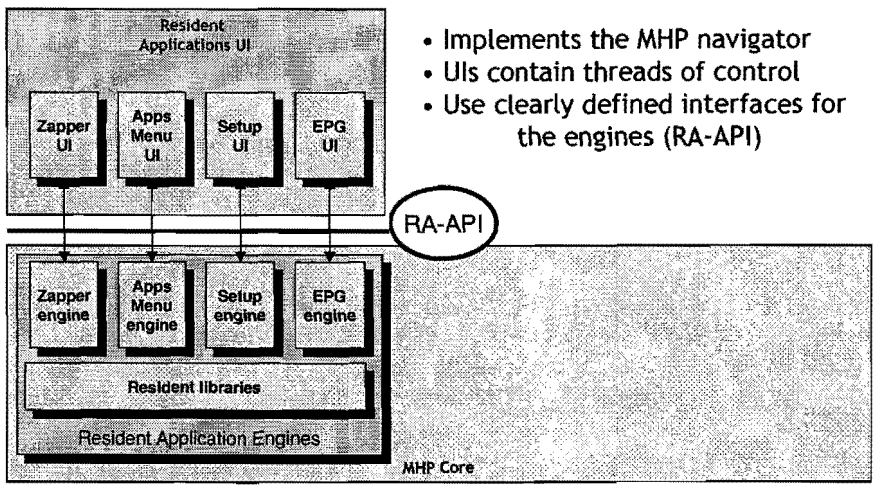
PHILIPS

Linux development system

- Characteristics
  - Intel/Linux PC with subset of HPI implementation (Basic framework HPI, streaming/access HPI without CA)
  - DVB Hauppauge card, CI card
- Advantages
  - Lots of tooling available
  - Faster build/execute cycles
- Disadvantage
  - HPI is still incomplete
- Usage
  - First check of development / improvements / changes
  - Analyze reproducible problems

PHILIPS

Resident MHP Navigator Framework

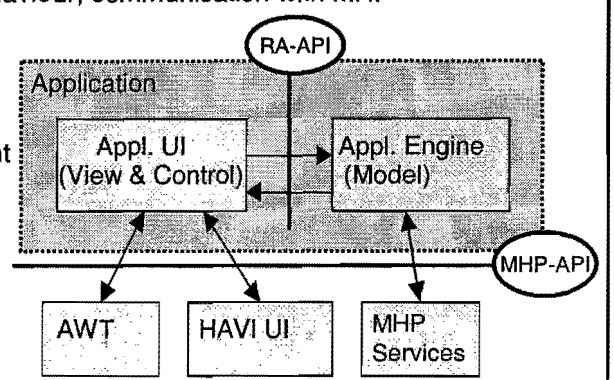


- Implements the MHP navigator
- UIs contain threads of control
- Use clearly defined interfaces for the engines (RA-API)

PHILIPS

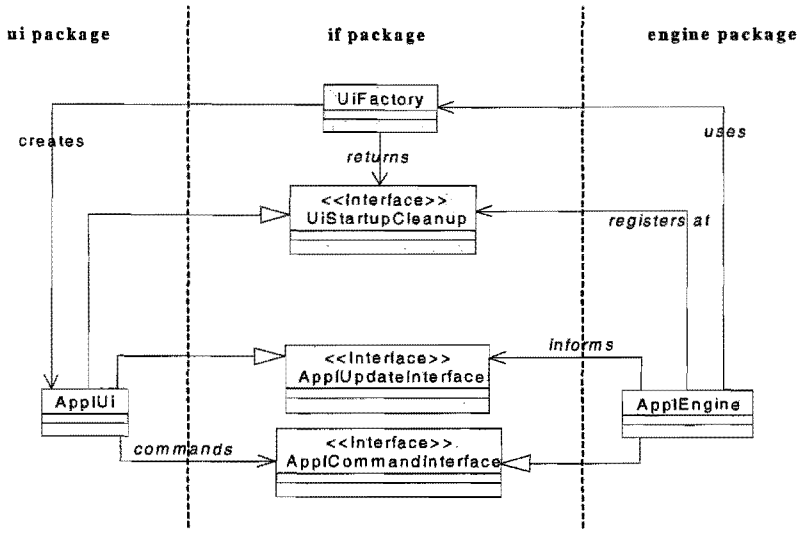
Our MHP Application split-up

- Based on Model-View-Control (MVC) design pattern
  - UI (View and Control) - presentation and control
  - engine (Model) - behaviour, communication with MHP
- Advantages:
  - separate development
  - replaceable UI



PHILIPS

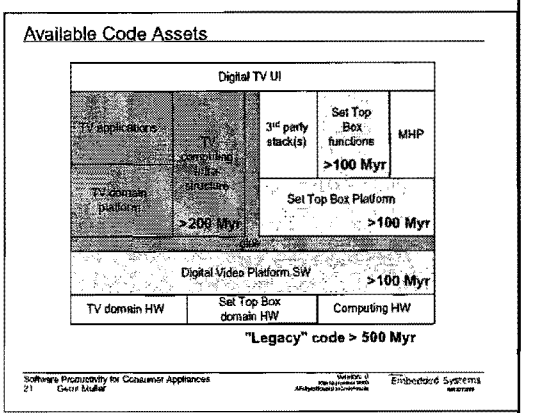
MHP Application – pattern overview



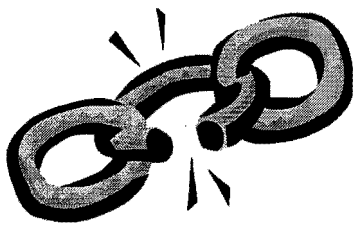
PHILIPS

MHP Product SW Size

- Code size: 16-32MB, ~2MLOCs
- Complexity is considered high for a CE product
- See e.g. Gerrit Muller's presentation to get a feeling for this at http://www.extra.research.philips.com/natlab/sysarch/SoftwareProductivitySlides.pdf



MHP Implementation
Robustness Measures



PHILIPS

Robustness measures

MHP Applications (Xlets) can fail for various reasons.
The MHP product should not fail.

- Decoupling Xlets from MHP implementation
  - Threaded launching of Xlets
  - "Clean listener" mechanism <= discussed in detail
- Java Heap monitor
  - Terminate Xlets before memory is exhausted
- Robust Xlet termination strategy
  - Catching unhandled exceptions
  - Releasing used resources
- And more...

PHILIPS

Threaded launching of Xlets

- Decouple threads in the MHP implementation from downloaded Xlets
  - This prevent a simple failure in an Xlet to cause a failure in the MHP implementation
- This implies
  - Threaded launching of Xlets
  - Clean listener mechanism for asynchronous events
  - And more...

PHILIPS

(Clean) listener mechanism

- The MHP specification requires listeners for all asynchronous calls
  - Events report back the result
- The MHP specification allows monitoring external events
- Our own implementation needs other asynchronous calls

PHILIPS

Kinds of listeners in the MHP spec

- One shot listeners
  - Get one event as a result of one call to a time consuming method
  - E.g. receiving a single DVB Service Information element
- Repeated listeners
  - Get certain events as result of a "subscribe" call, until unsubscribed
  - E.g. monitoring Service Information changes
- "Standard" listeners
  - Get all events from a source until unsubscribed
  - E.g. Tuner (Resource) status

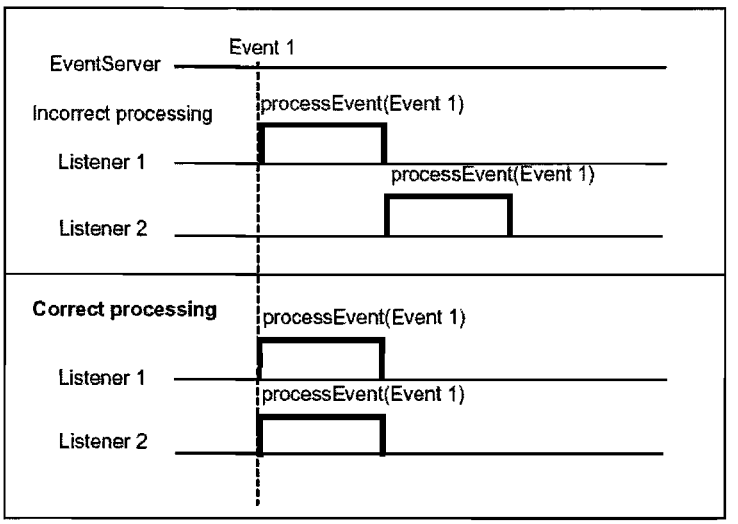
PHILIPS

Requirements on listener framework

- Generic (thread) decoupling scheme
- Safeguard against multiple subscriptions of one listener
- Events are dispatched in parallel to all listeners
- Multiple events are serialized for one listener
- Listeners are disposed when application is destroyed
  - to allow Java garbage collection to work
- Be sober in creating threads

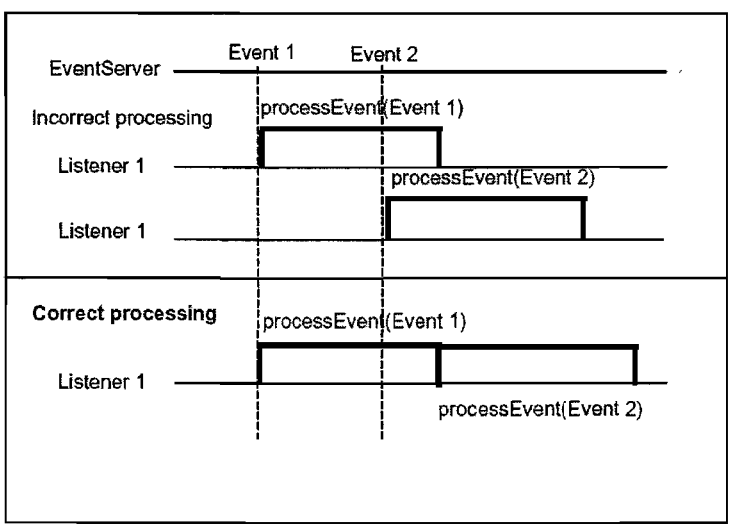
PHILIPS

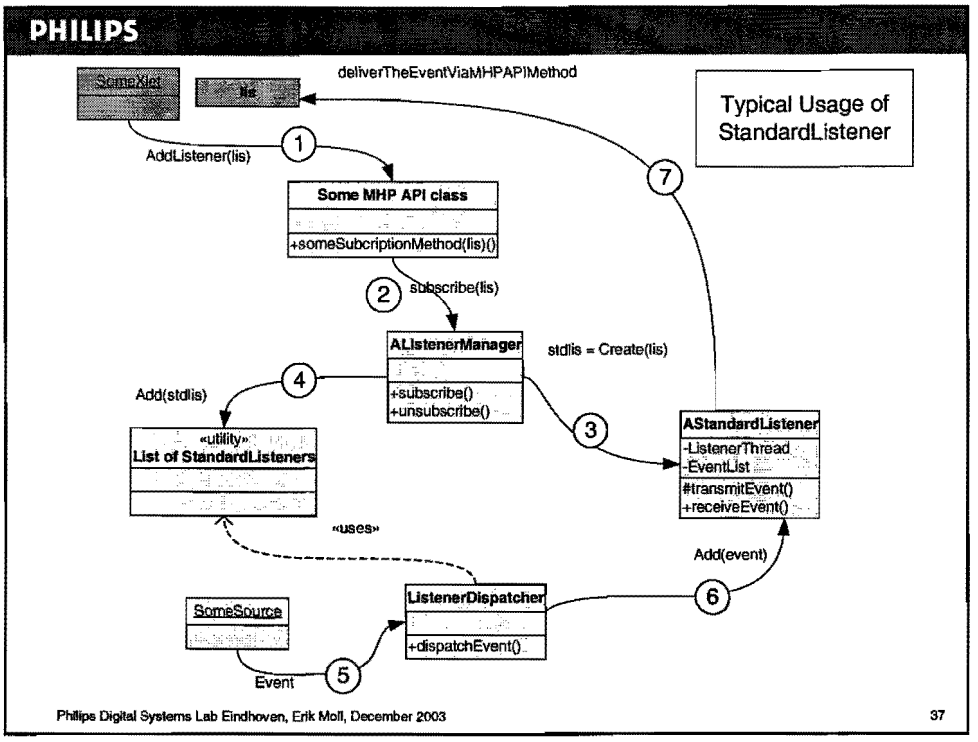
Parallel dispatching

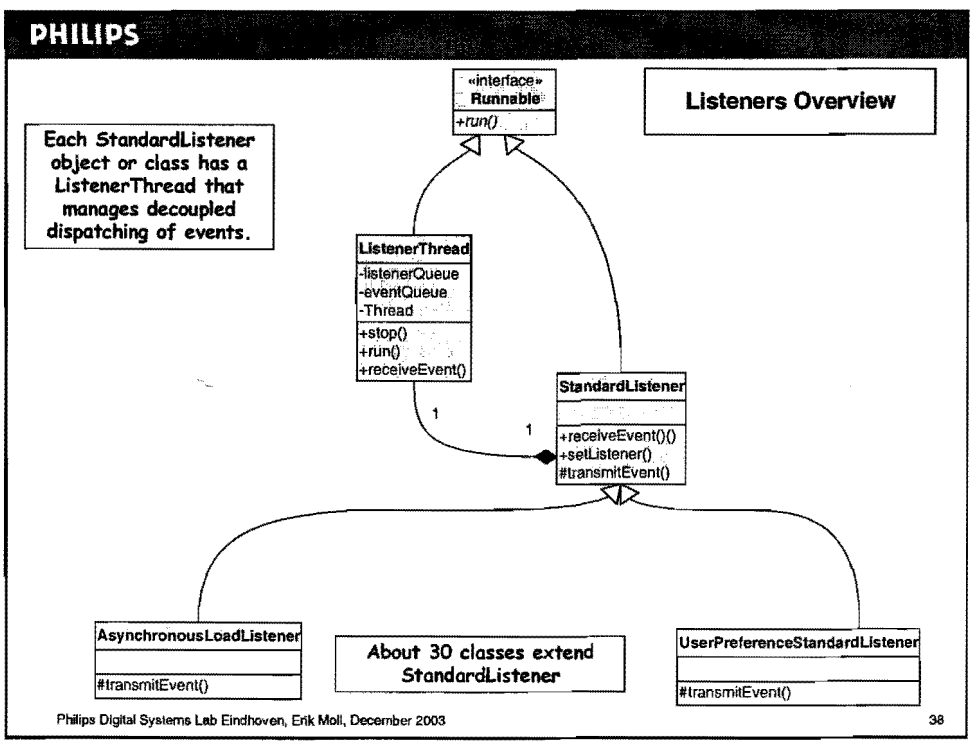


PHILIPS

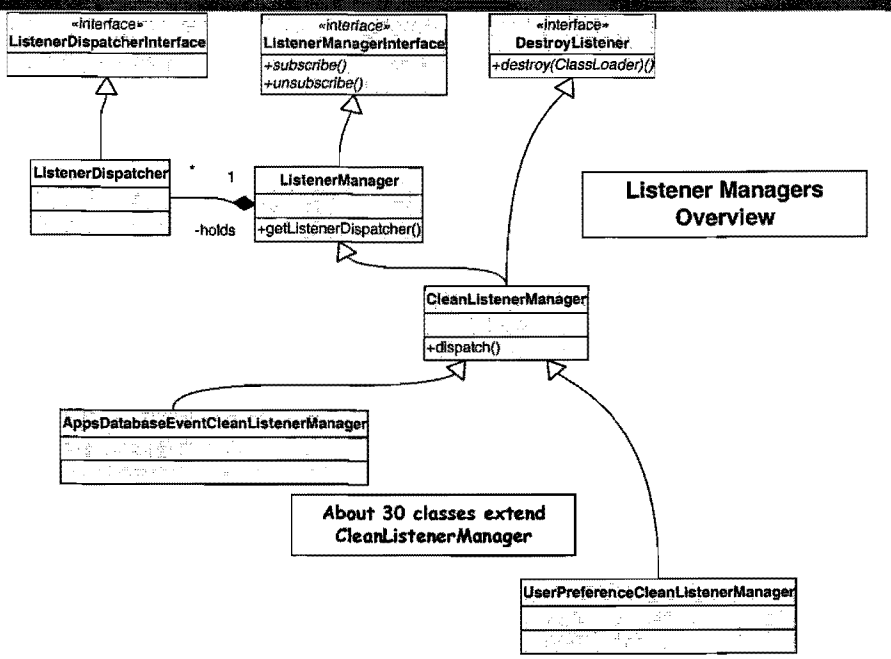
Serialized dispatching
(not defined by MHP spec....)




PHILIPS
Typical Usage of StandardListener
SomeXlet
lis
deliverTheEventViaMHPAPIMethod
AddListener(lis)
1
Some MHP API class
+someSubcriptionMethod(lis)()
2
subscribe(lis)
AListenerManager
+subscribe()
+unsubscribe()
stdlis = Create(lis)
3
4
Add(stdlis)
«utility»
List of StandardListeners
AStandardListener
-ListenerThread
-EventList
#transmitEvent()
+receiveEvent()
7
«uses»
Add(event)
6
ListenerDispatcher
+dispatchEvent()
SomeSource
Event
5
Philips Digital Systems Lab Eindhoven, Erik Moll, December 2003
37


PHILIPS
Listeners Overview
Each StandardListener object or class has a ListenerThread that manages decoupled dispatching of events.
«interface»
Runnable
+run()
ListenerThread
-listenerQueue
-eventQueue
-Thread
+stop()
+run()
+receiveEvent()
StandardListener
+receiveEvent()()
+setListener()
#transmitEvent()
1
1
AsynchronousLoadListener
#transmitEvent()
About 30 classes extend StandardListener
UserPreferenceStandardListener
#transmitEvent()
Philips Digital Systems Lab Eindhoven, Erik Moll, December 2003
38

PHILIPS



PHILIPS

Check Listener Requirements

- Generic (thread) decoupling scheme
  - OK – ListenerThread handles dispatching asynchronously
- Safeguard against multiple subscriptions of one listener
  - OK – ListenerManager.subscribe() checks.
- Events are dispatched in parallel to all listeners
  - OK – for single ListenerThread per listener object
  - NOT OK – for single ListenerThread & listenerQueue per class
- Multiple events are serialized for one listener
  - OK – one synchronized queue in ListnerThread
- Listeners are disposed when application is destroyed
  - OK – destroyListener takes care of this
- Be sober in creating threads
  - NOT OK – one thread per listener object
  - OK – one thread per listener class

Trade-off

PHILIPS

Summary of what I presented

- An overview of DVB-MHP
  - DVB-MHP defines a Java based open system for interactive digital TV
- How we implemented DVB-MHP
  - A big but portable and customizable SW system for a CE device
  - Using a Linux implementation as a development tool
- Robustness challenges in this CE SW system
  - Zooming into "clean listeners"
  - Our Philips MHP stack achieves "Consumer Electronic" quality
- Questions?

"System on Chip Architectures for MPEG-4 Video"

Dipl.-ing. Mladen Berekovic

University of Hannover, Germany

SOC Architectures for MPEG-4

Mladen Berekovic, Peter Pirsch,
Hans-Joachim Stolberg, Andreas Dehnhard,
and Sören Moch

Institute of Microelectronic Systems

Universität Hannover

Outline

- Introduction
- MPEG-4 Overview
- SOC Architectures for MPEG-4
  - Dedicated implementations
  - Hybrid implementations
  - Programmable approaches
- Multimedia Processor Design Examples
  - Dedicated: Fujitsu
  - Hybrid: Toshiba, Trimedia
  - Fully programmable: TI's OMAP architecture
  - HiBRID-SOC from University of Hannover
- Conclusion

ISO-MPEG Video Compression standards

- Industry standard: ISO-MPEG
  - Digital TV: DVD, DVB, set-top boxes
  - Video Conferencing / 3GPP
  - Streaming Video
- ITU-T H.261-H.264
  - ITU standardization efforts on video coding
  - Joint standards with ISO-MPEG: H.262/MPEG-2 and H.264/MPEG-4/10
- MPEG-1/2 (1991, 1995)
  - Hybrid video coding scheme based on block matching
- MPEG-4 (since 1996)
  - Ongoing standardization effort for universal coding of multimedia contents
  - New profiles add object-based functionalities

MPEG-4 Overview

- 10 different parts (1-14, to be continued)
  - Video coding: Part 2 (Visual), Part 10 (AVC)
- Subdivided in profiles and levels
  - Targeting specific application (> 12 profiles)
- Profiles specify subsets of video coding "tools"
- Only few profiles adopted by market yet
  - Simple Profile (SP): Video communication, 3GPP
  - Advanced Simple Profile (ASP): DVD, Internet
  - Core Profile: Few implementations exist
- MPEG-4 Part 10: Advanced Video Coding (AVC)
  - Non backward-compatible
  - Completely new tools and profiles
  - Still based on hybrid-coding scheme, no object-based functionality

MPEG-4 Part 2 Profiles & Levels (excerpt)

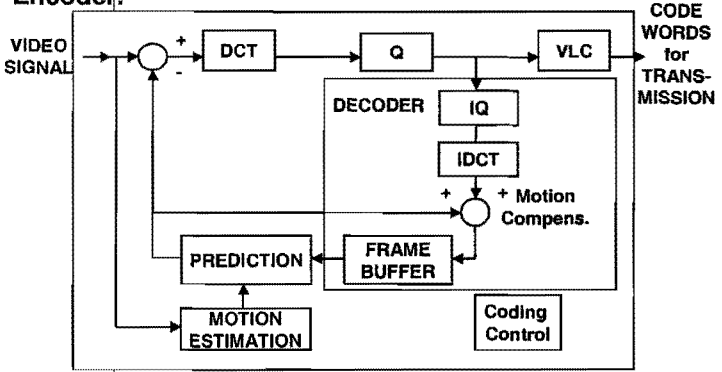
Visual Profile	Version/ Amend.	Levels(Size)	Object based	Max bitrate kbps
Simple	IS	L0 (QCIF)-L3 (CIF)	N	64-384
Core	IS	L1 (QCIF)-L2 (CIF)	Y	384-2000
Main	IS	L2 (CIF)-L4 (HDTV)	Y	2000-38400
Advanced Coding Efficiency (ACE)	v.2	L1 (CIF)-L4 (HDTV)	Y	384 38400
Advanced Simple	Amd 2	L0 (QCIF)-L5 (D1)	N	128-8000
Fine Granularity Scalability	Amd 2	L0 (QCIF)-L5 (D1)	N	128-8000

MPEG-4 Visual Coding Tools

- MPEG-4 video coding still based on hybrid-coding scheme
  - New tools added to improve coding efficiency
- MPEG-4 introduces profiles with object-based coding
  - Interactive manipulation of video content possible
  - New tools necessary to handle
    - coding of shaped objects
    - scene compositing from multiple objects
- Large, growing number of profiles & levels
  - Targets great variety of applications

MPEG Hybrid Video Coding Scheme

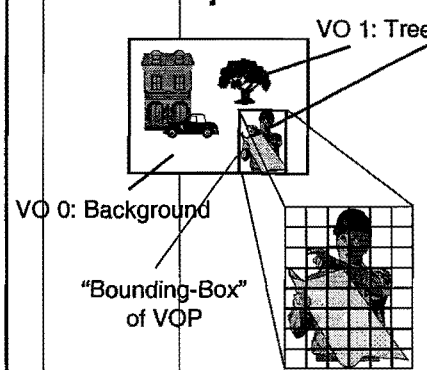
- Encoder:



- Decoder: Inverse functions, included in encoder
- Macroblock Processing: (I)DCT, (I)Q, Motion-Comp., -Estimation

MPEG-4: Object-based Video Coding

- VOP: Projection of VO into image plane



- ☐ Blocks outside VOP
  - ❶ not coded
- ☐ Blocks inside VOP
  - ❶ block-based coding like in MPEG-1/2
- ☐ Boundary blocks (BB) of VOP
  - ❶ new prediction and coding algorithms

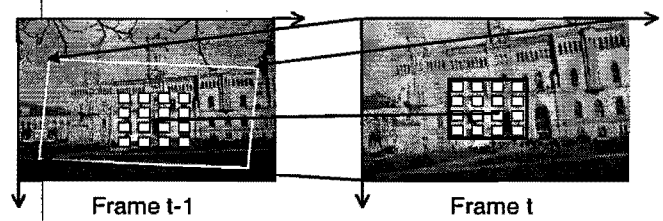
→ MPEG-4: independent coding & transmission of each VO
→ Compositing, manipulation of VOs to scenes in decoder

MPEG-4: New Coding Tools

- Coding of non boundary blocks similar to MPEG-1/2
  - MC, DCT, Q, VLD
- New tools for improved coding efficiency (MPEG-4 SP/ASP)
  - AC/DC prediction (SP+ASP)
  - Quarter-Pel MC (ASP)
  - Global Motion Compensation (GMC: ASP)
- Deblocking & deringing filters (optional post-processing)
- New algorithms for coding of boundary blocks
  - Context based arithmetic encoding (CAE) for binary shape mask
  - Padding: expands BB texture to full 16x16 texture block
  - Shape Adaptive DCT
- Additional processing step: scene compositing
  - Alpha Blending, Perspective transform of VOs
  - Similar to 3D- texture-mapping algorithms

New Tools Example: Image Warping for GMC

- Global motion parameters for camera movement
  - Translation, pan, zoom, rotation
  - Single set of parameters per frame
- Affine Transformation with 3 Warping Points
- 6 Warping Paramters A, B, C, D, E, F
- Bilinear Interpolation of reconstructed pel



SOC Implementations for MPEG-4

- Application driven design for specific profile(s)
- Function-Specific Implementations
  - Simple Profile, QCIF (mobile phones)
- Hybrid implementations
  - Processor core + hardware coprocessors
  - Adds more flexibility
  - SP, MPEG-2, QCIF-D1
- Programmable solutions
  - Multiple, individually optimized processor cores
  - Multiple profiles, standards, applications

Dedicated VLSI Implementations for MPEG-4

- Design Target
  - Low power, low cost
  - Single application solution

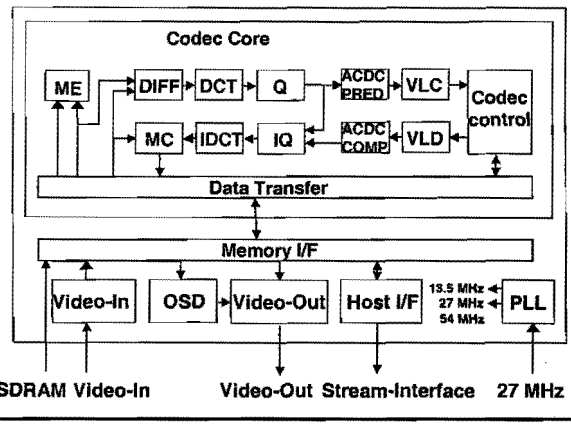
Fixed Design: Direct Mapping of tasks on HW blocks

- Approach
  - Mapping of individual tools onto different processing blocks
  - Optimize each block in terms of
    - Parallel processing capabilities
    - Size
    - Power
  - Determine processing schedule

Function-Specific Architectures @ Minimum Cost

Dedicated MPEG-4 VLSI Example: Fujitsu 2002

- Hybrid video coding scheme directly mapped on hardware
- MPEG-4 SP@L3: QCIF and CIF CODEC, 13.5 MHz



Hybrid SOC Implementations for MPEG-4

- 16/32 bit RISC processor for control operations
- Function-specific co-processor modules
- Standardized on-chip system bus for IP-cores, interfaces

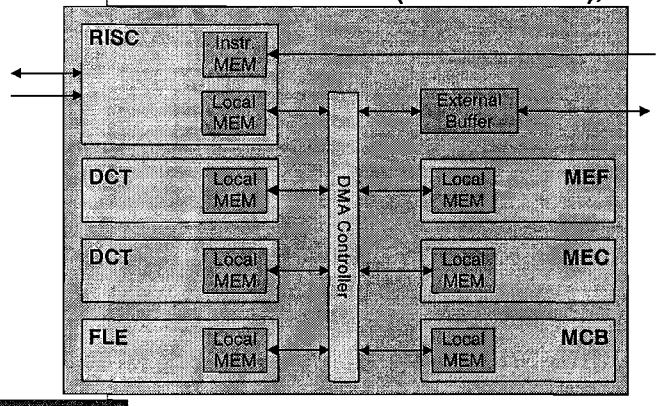
System-on-Chip (SOC) Design

Flexible co-processors offer "Programmable Solutions"

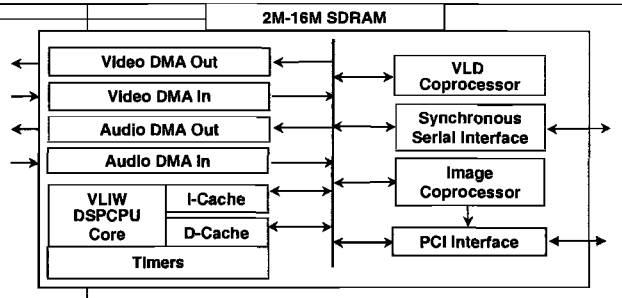
- Typical implementations
  - Toshiba
  - InTime
  - Sigma Designs
  - .....

MPEG-4 Hybrid SOC Architecture: Toshiba

- Toshiba Hybrid SOC MPEG-4 Video Codec TC35274
- QCIF codec for MPEG-4 SP@L1 (G3PP 3G-324M), 70 MHz



Hybrid-Architecture: Philips Trimedia TM-1000



- 140 MHz 32-bit VLIW core: 5 operation slots / 27 function units
- Specialized instructions with subword parallelism
- Hardwired coprocessors for VLD, image processing
- 32-bit on-chip PI-bus connects CPU with all cores and interfaces

MPEG-2 Decoding

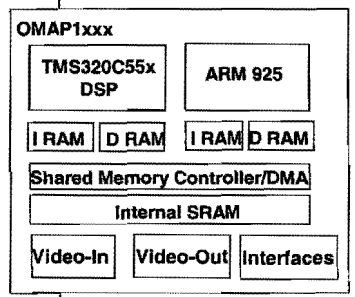
Programmable Implementations for MPEG-4

- Rapid evolution of algorithms
- Approach
  - Isolation of similar data and processing types in application
  - Mapping onto independent, programmable processor cores
  - Optimize each core in terms of
    - Parallel processing capabilities
    - Instruction set
    - Load-/store architecture

SoC Design: Heterogeneous architecture with multiple programmable, task-specific cores

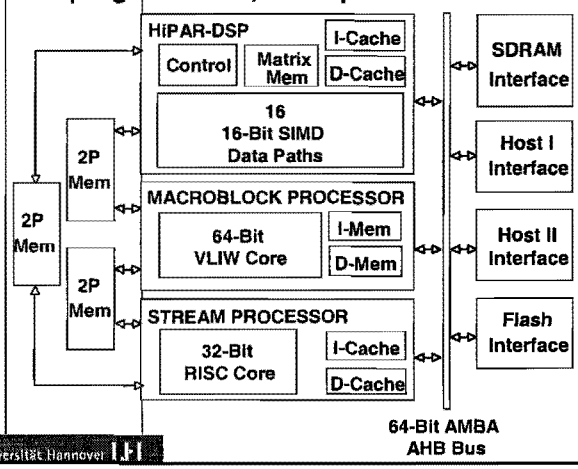
Programmable SOC Example: TI/ST OMAP

- TI/ST-Micro OMAP-Platform: MPEG-4 SP CIF (3GPP), 200 MHz
- ARM+DSP, AMBA bus architecture
- Fully programmable solution



HiBRID-SoC Multi-Core Architecture for MPEG-4

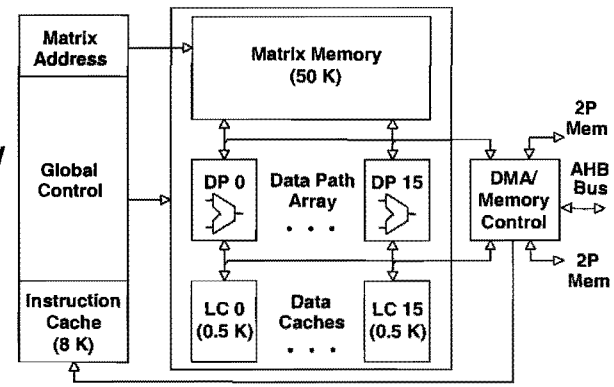
- 3 programmable, task-specific cores:



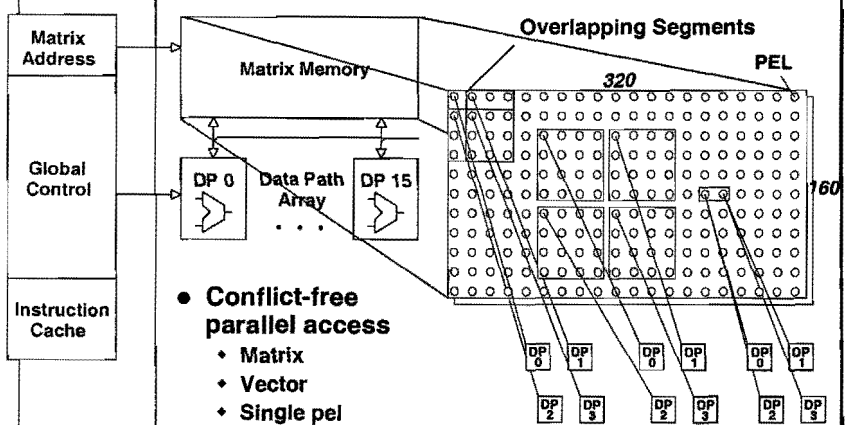
- HiPAR-DSP
- Macroblock Processor (MP)
- Stream Processor (SP)
- Dual-port memories
- 64-Bit AMBA AHB system bus
- Various interfaces

HiPAR-DSP Core

- Image processing, FFT, segmentation
  - Large filter kernels
- 16 SIMD-controlled 16-Bit data paths (DP)
- DP: 4-way VLIW
  - 1x Load/Store
  - 3x Arithm/Ctrl
- I-, D-caches
- Matrix Memory supporting 4x4 shared access



HiPAR-DSP Matrix Memory

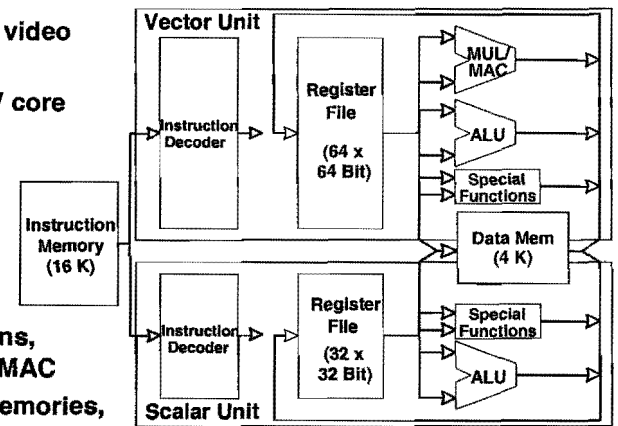


- Conflict-free parallel access
  - Matrix
  - Vector
  - Single pel

Efficient support of 2D signal processing

HiBRID-SOC: Macroblock Processor (MP) Core

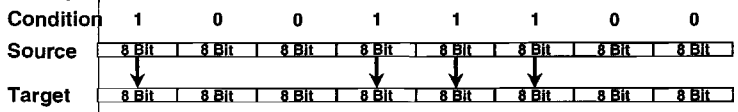
- Block-based video processing
- 2-issue VLIW core
- Vector unit (64 Bit, splittable)
- Scalar unit (32 Bit)
- Specialized ISA extensions, 128-Bit split MAC
- Local I-, D-memories, autonomous DMA



Macroblock Processor: Flexible Parallelism

- Subword parallelism (SIMD) on vector unit
  - Splittable 64-Bit registers: 8x8-, 4x16-, 2x32-Bit
  - Conditional execution on individual subwords

Example: "Conditional Move"



- Flexible combination of scalar and vector instructions in a single 64-bit VLIW

VLIW		Execution
vector	scalar	parallel
vector	vector	parallel (on distinct units)
scalar	scalar	sequential

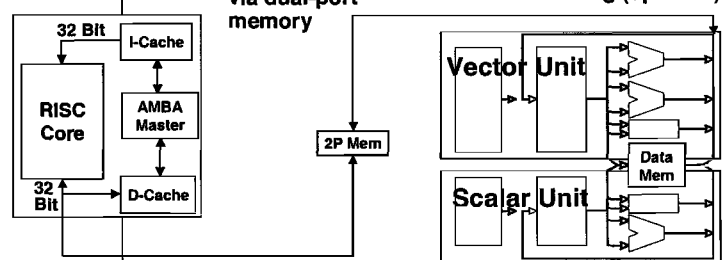
High performance (vector), high code density (scalar)

MPEG-4 ASP Mapping on HiBRID-SoC's SP, MP

SP core
- Bitstream parsing
- VLD
- Motion vector decoding
- IQ

MP core
- IDCT
- Reconstruction
- Motion Compensation
- Global MC
- Quarter-Pel MC
- Deblocking (optional)

Synchronization, data exchange via dual-port memory

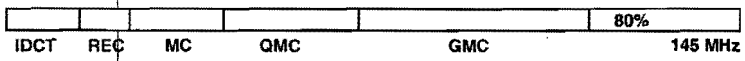


MPEG-4 Performance on HiBRID-SoC

- Implementation on UMC 6LM 0.18μ technology: 145 MHz
- Decoder

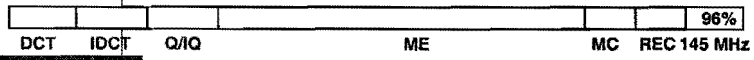
MPEG-4 Simple Profile, 352x288@15Hz (CIF), 384 kBit/s

MPEG-4 Advanced Simple Profile ("DivX-5")
720x576@25Hz (D1), 1.5–3 MBit/s



- Encoder

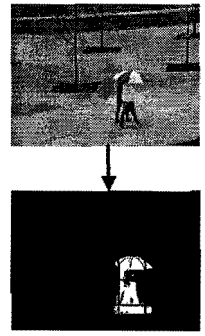
MPEG-4 Simple Profile, 720x576@25Hz (D1), 3 MBit/s



Universität Hannover IJI 25

Object Tracking on HiBRID-SoC's HiPAR-DSP

- Core algorithms for object tracking
  - Detection of object pixels
  - Connected component labeling
  - Identification of regions of interest (ROIs)
- Object pixel detection
  - Mapping of data parallelism of filter-based algorithms onto 4x4 datapath array
  - Adaptability to different scenarios and algorithms, e.g., moving vs. static camera
- Connected component labeling
  - Divide-and-Merge algorithm to exploit data parallelism
- ROI identification
  - Connected components analysis, e.g., bounding box calculation during labeling



Universität Hannover IJI 26

Custom Security Application Scenario

- Multi-channel surveillance system with ROI focus

...0010101...

Transmission

...1111001...

...000111...

...1010101...

Background: Low resolution

ROI: High resolution

HiBRID-SoC *(Sender)*
1) Object Tracking
2) ROI Detection
3) Custom MPEG-4 Encoding
4) Transmission

HiBRID-SoC *(Receiver)*
Multi-channel MPEG-4 decoding
Reduced complexity outside ROI

MPEG-4 alternatives: H.264/AVC, JPEG2000

Conclusion

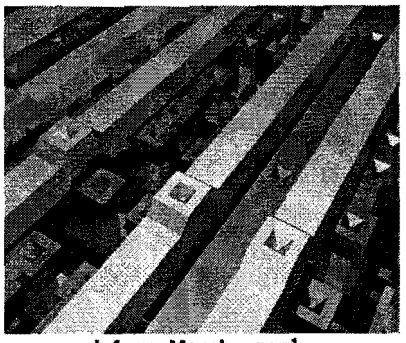
- MPEG-4 targets many different applications with different parts, profiles and level definitions
- New tools
  - Improved coding efficiency
  - New functionality (object based coding)
  - Increasing computational complexity
- Different MPEG-4 SOC solutions
  - Limited to specific profiles, levels (SP/ASP)
- Programmable multi-processor SOC architectures
  - Multi-profile, multi-standard solution
  - Early adoption of new standards, i.e. AVC, possible

"On the Design of Multimedia Software and Future System Architectures"

Dr.ir. Egbert Jaspers, Prof.dr.ir. Jef van Meerbergen

Philips Research Labs, Univ. of Technol. Eindhoven, NL

On the Design of Multimedia Software and Future System Architectures



Jef van Meerbergen[1]
Egbert G.T. Jaspers[2]
[1]Philips Research Labs, Eindhoven, The Netherlands
[2]LogicaCMG / Philips Research Labs, Eindhoven, The Netherlands

Outline

- Introduction of future system
- Scalable architectures for reuse
- SW Design for scalable systems
- Design for predictable performance
- Conclusions

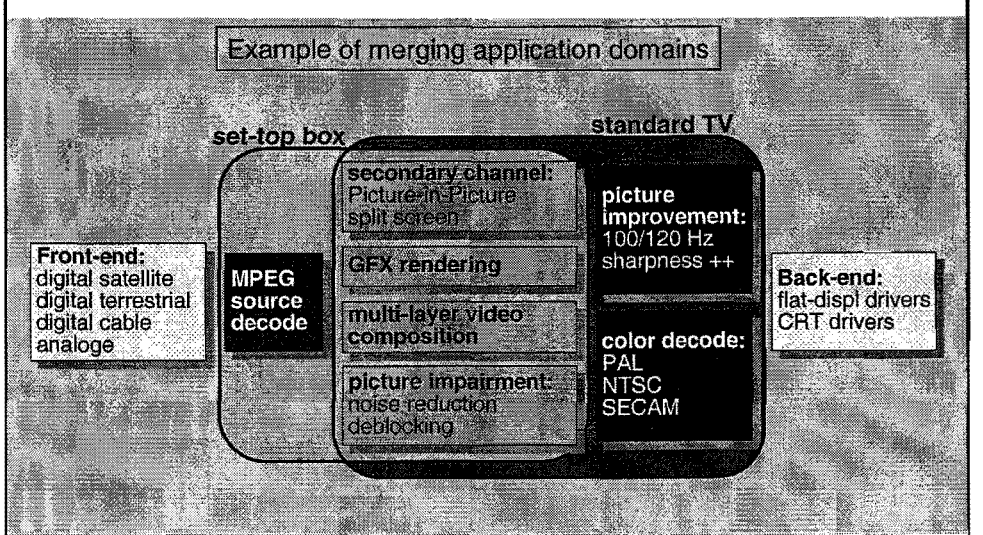
Introduction
Observation

- Increasing complexity of SoCs in CE
  - Design costs exceed BOM
- Non-scalable solutions and resource sharing
  - Introduces bottlenecks for up scaling
  - Unpredictable system behavior increases risks
  - Execution architecture assessment required
- Generic solutions that are broadly applicable
  - Distributed development costs over large volumes
  - Reuse of design effort
  - E.g. a SoC for TVs, Set-top boxes, and DVD players

Philips
Research

PHILIPS

Introduction
Observation



Philips
Research

PHILIPS

Introduction

Consequences for future CE

- Increasing flexible / programmable HW in complex SoCs
- Computational intensive media processing
- Due to increasing design costs, silicon area becomes less relevant
- Identified major design challenges
  - **Scalable architectures for reuse**
  - **SW design for scalable systems**
  - **Resource-scalable media processing for QoS**
  - **Design for predictable performance**

Philips
Research

PHILIPS

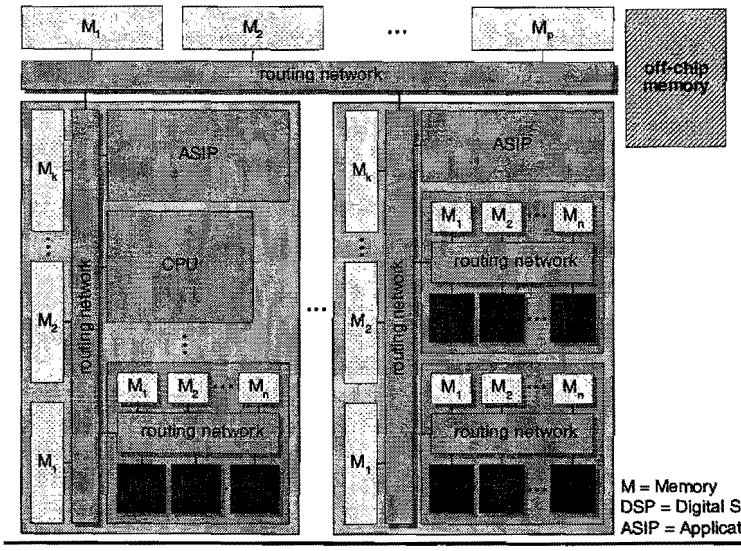
Outline

- Introduction of future system
- Scalable architectures for reuse
- SW Design for scalable systems
- Design for predictable performance
- Conclusions

Philips
Research

PHILIPS

Scalable architectures
Proposed system architecture concept



Philips Research

PHILIPS

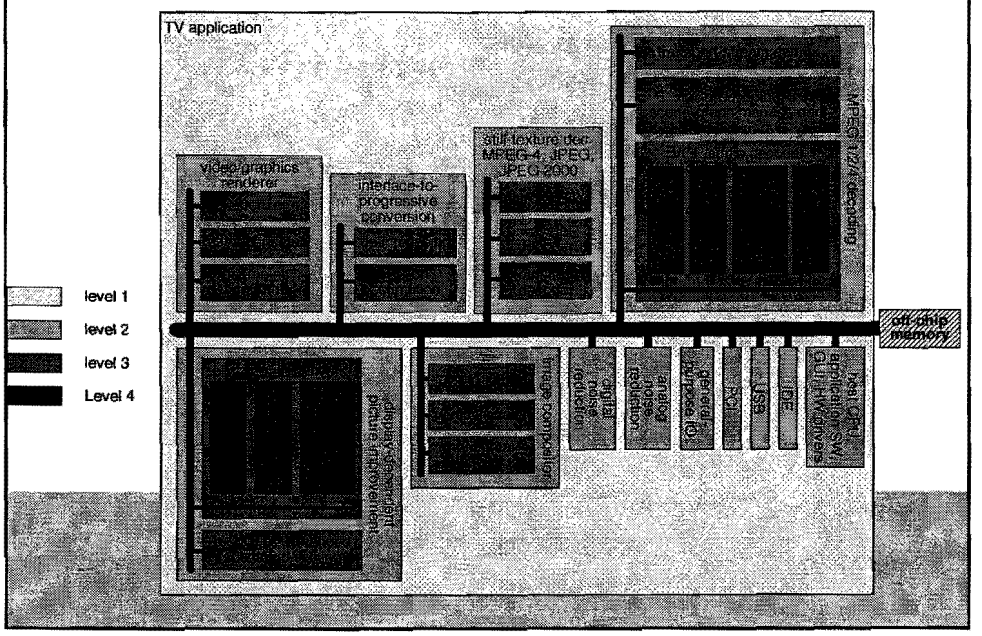
Scalable architectures
Characteristics

- Hierarchical communication network providing:
  - Routers connecting a redundant topology
  - Bandwidth and latency guarantees
- Distributed and hierarchical memories
  - Optimised for locality of data
  - Off-chip SDRAM at highest level
- Multiple programmable processors
- Support of special-purpose HW for acceleration

Philips Research

PHILIPS

Scalable architectures
Example TV system



Scalable architectures
Example TV system d'cont

- Shared primitive tasks in separate subsystem
  - E.g. ME for motion-blur filtering in LCD, 50-to-100Hz in CRT, or subfield generation in PDP
- Maximize locality of data by clustering
  - E.g. DCT and MC local in the video object (VOP) decoder
- Tasks that occur in different position of task graph in separate subsystems
  - E.g. Deinterlacing for display (backend) or vertical video processing (front end)
- Sharing of ASIP is limited
- Partitioning into a hierarchical tree is application specific

Outline

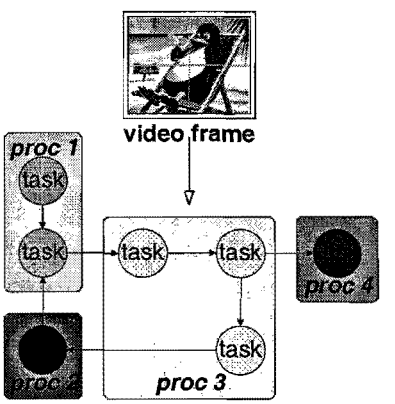
- Introduction of future system
- Scalable architectures for reuse
- SW Design for scalable systems
- Design for predictable performance
- Conclusions

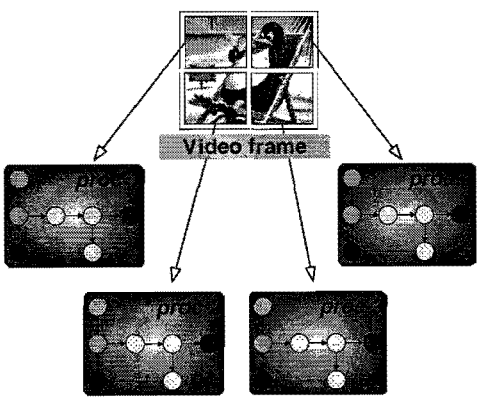
Philips
Research

PHILIPS

SW for scalable processor systems

Partitioning of a H.264 decoder (1/2)





Functional partitioning:
Individual tasks mapped on CPUs

Data partitioning:
Application instances mapped on CPUs

Philips
Research

PHILIPS

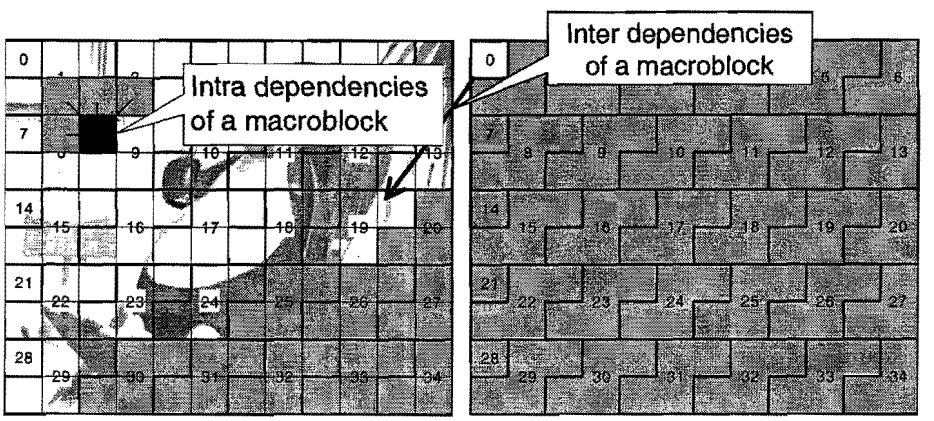
SW for scalable processor systems
Partitioning of a H.264 decoder (2/2)

- Advantages functional partitioning
  - Natural architecture
  - Reusability
  - I-cache performance
- Advantages data partitioning
  - More potential parallelism
  - Better scalability
  - Efficient communication
  - D-cache efficiency
  - Natural load balancing

Philips Research

PHILIPS

SW for scalable processor systems
Data partitioning example



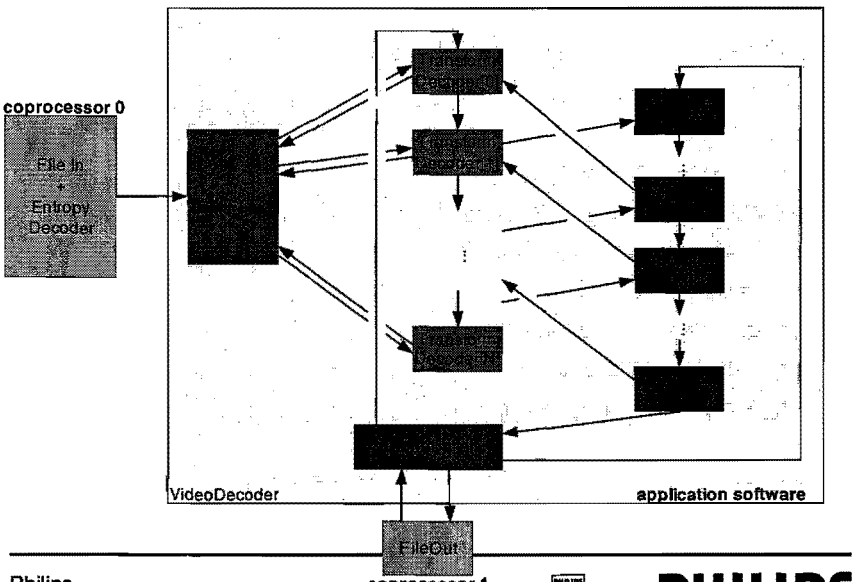
Reference picture

Next B- or P picture

Philips Research

PHILIPS

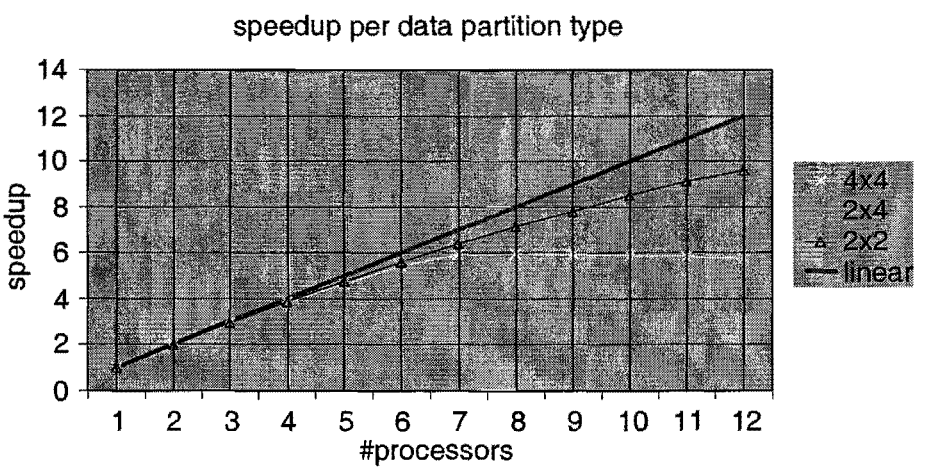
SW for scalable processor systems
Multi-threaded design



Philips
Research

PHILIPS

SW for scalable processor systems
Results on scalability



Philips
Research

PHILIPS

Outline

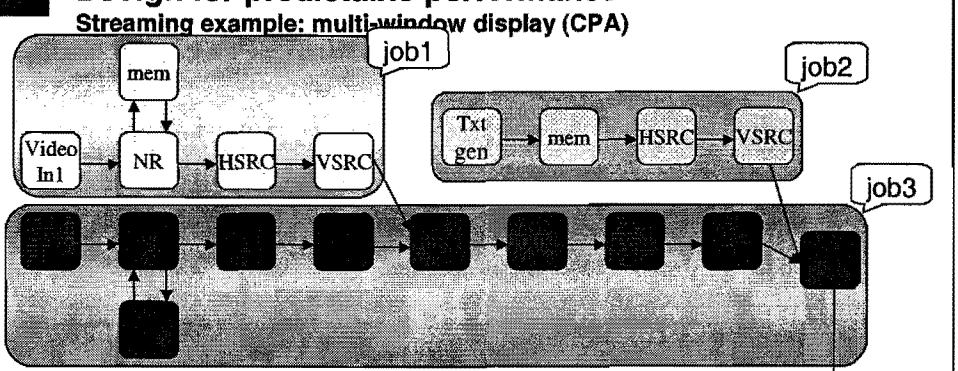
- Introduction of future system
- Scalable architectures for reuse
- SW Design for scalable systems
- Design for predictable performance
- Conclusions

Philips
Research

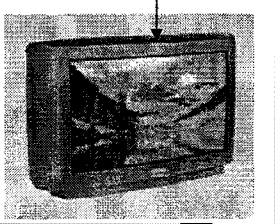
PHILIPS

Design for predictable performance

Streaming example: multi-window display (CPA)



- Tasks = well defined compute kernels
- job = activities controlled/started/stopped by the user
- different modes: 16 - 32 - 64 MHz
  - 1 stream: 64 MHz
  - PIP: 2 x 32 MHz
- different graphs with different orderings

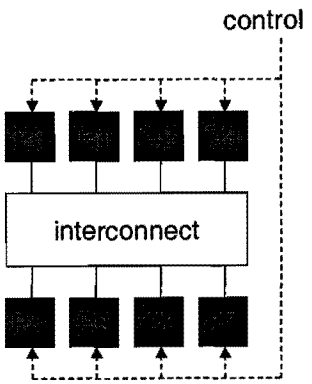


Philips
Research

PHILIPS

Design for predictable performance
Streaming architecture

- processors as building blocks
- spanning a wide range of programmability
  = RISC, VLIW, DSP, ASIP, hardwired
  = high computational efficiency
- embedded memories
- IP reuse, including compiler
- control = microcode ... memory mapped registers
- Interconnect = NoC
- coarse level reconfigurability
- computation in space: top level (TLP)
- computation in time: within processors (ILP)



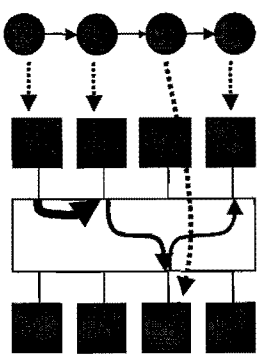
Philips
Research

PHILIPS

Design for predictable performance
Mapping of a job

Computation: Task level parallelism
tasks ==> processors
e.g. T0 ==> P0 for 50% of time

Communication resources
channels ==> connections (often with guaranteed throughput e.g. 16 - 32 - 64 MHz)

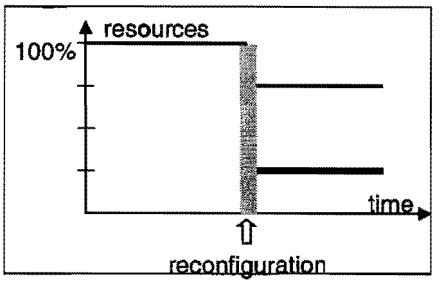


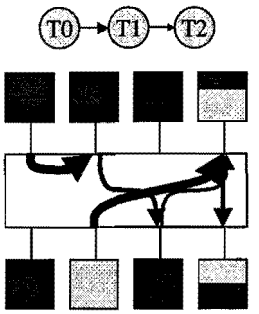
Philips
Research

PHILIPS

Design for predictable performance
Mapping multiple jobs

Multiple jobs can be active simultaneously.
When can a second job start ?
Are the requested resources available ?
If not, can the quality level be lowered ?
If not, can other jobs go for a lower quality ?
If yes, independent from other jobs ?

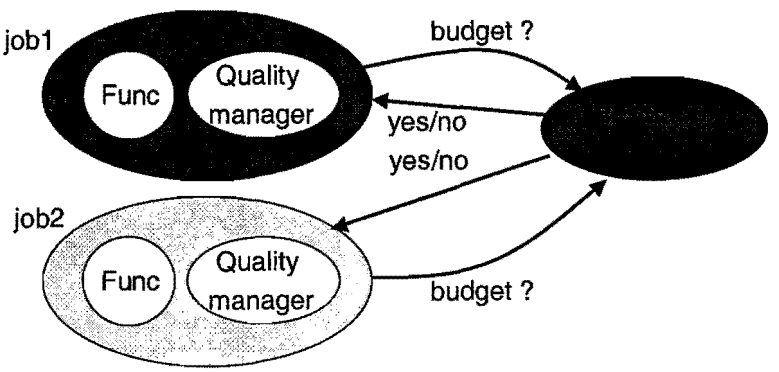




Philips Research

PHILIPS

Design for predictable performance
Reconfiguration: negotiation of budgets



[Steffens/IPA]

Philips Research

PHILIPS

Design for predictable performance

Predictability: What is needed ?

Predictability = a reduction of uncertainty

- reduce uncertainty to negotiation phase [Steffens/IPA][Goossens]
  - stating which resources you need (comm. + comp.)
  - having the RM to commit or reject your request.
  - renegotiate when the requirements change
- within a configuration:
  - for one job
    - reasoning about performance: design for a given budget + timing constraints (design time)
  - for multiple jobs
    - compositional = no interaction between jobs = concept of virtual platform (run time)

==> impact on the architecture

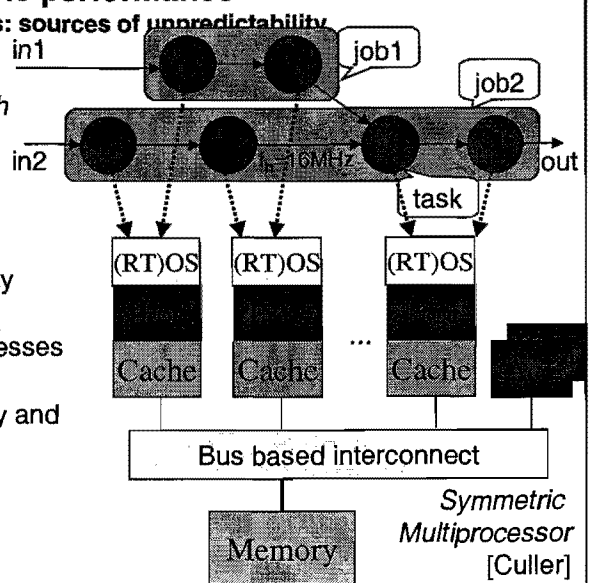
Philips Research

PHILIPS

Design for predictable performance

Conventional architectures: sources of unpredictability

Kahn/Yapi task graph



Symmetric Multiprocessor [Culler]

- rationale: driven by flexibility
- dynamic load balancing via flexible binding of Kahn processes to processors
- key issue: cache coherency and memory consistency
- performance analysis via simulation

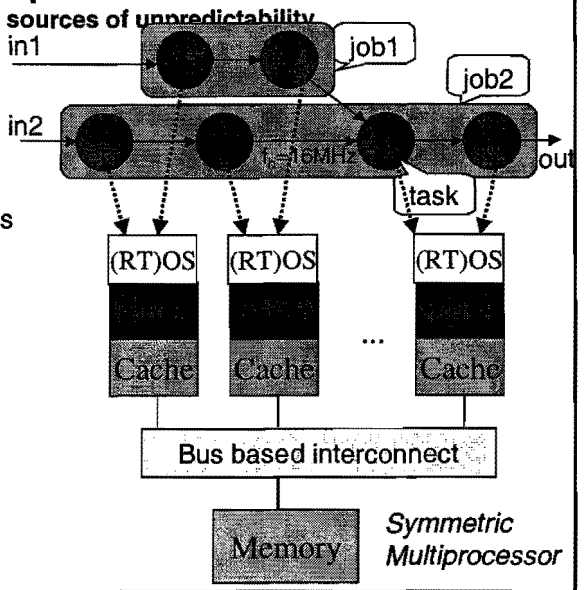
Philips Research

PHILIPS

Design for predictable performance

Conventional architectures: sources of unpredictability

Kahn/Yapi task graph



- architecture: resource conflicts
  - task switches
  - cache misses
  - busses, bridges
  - memories
- cost
  - coarse level sync
  - 2 x power dissipation
  - expensive caches

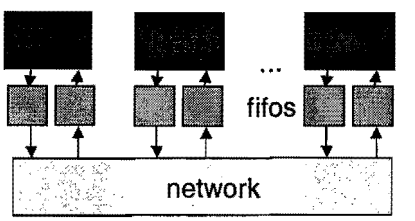
Philips
Research

PHILIPS

Design for predictable performance

New reconfigurable architecture

- driven by predictability and low cost
- no multiple copies of data (single producer, single consumer)
- reflect task graph structure via programming connections over the network
- decoupling computation from communication with small fifos
- compute in time: ILP within 1 proc
- compute in space: TLP (coarse level reconfigurability)



[Hijdra/ESAS+IT]

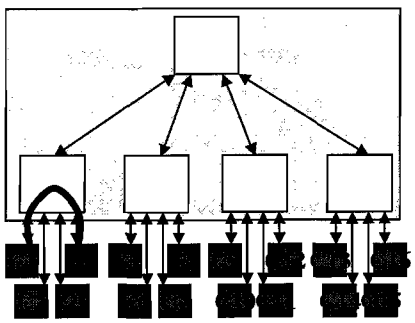
Philips
Research

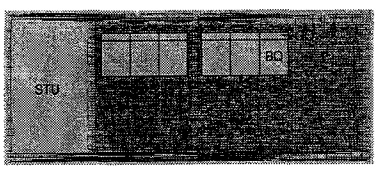
PHILIPS

Design for predictable performance
Networks-on-Silicon

- a graph based structure with routers and links
- point-to-point segmented interconnect ==> scalable
- DTL/CoReUse compliant design entity, e.g. embedded cores, IP blocks, subsystems, pipes, tiles ... + memories
- Services = connections + properties (guaranteed throughput, best effort...)

[Aethereal/Goossens ESAS/VLSI D&T]



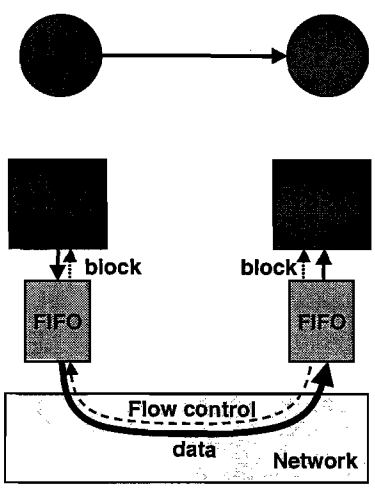


•0.25 mm2 CMOS12

Philips Research

PHILIPS

Design for predictable performance
New reconfigurable architecture



- synchronization: fine grain and local
- smaller latency = smaller memories
- processor blocks if FIFO is full/empty
- network uses a guaranteed throughput (GT) connection
- flow control blocks end-to-end network connections

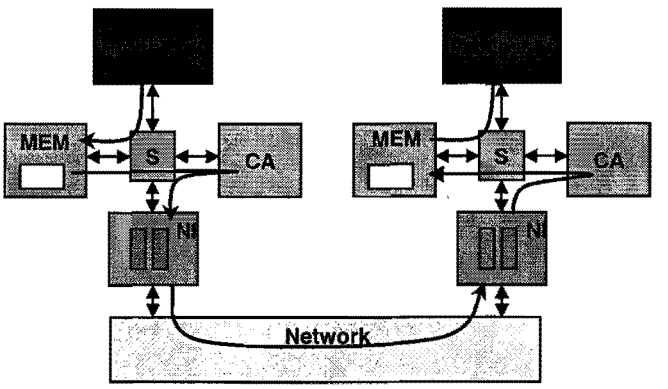
[Hijdra/ESAS+IT]

Philips Research

PHILIPS

Design for predictable performance

New reconfigurable architecture: programmable cores



Different buffer sizes = different synchronization rates

Philips
Research

PHILIPS

Conclusions

- Due to increasing SoC complexity design costs exceed the BOM
- Focus in system design shifts from minimal silicon area towards:

1. Generic and scalable architecture for reuse
- *Hierarchy in the communication network and the memory*
- *Flexible / programmable with special-purpose HW for acceleration*
- *Application-specific partitioning into a hierarchical structure*

1. SW design for scalable architectures

2. Resource-scalable media processing for QoS

3. Design for predictable performance
- *Multiprocessors with different levels of programmability and an on-chip network (NoC) and resource management (budgets)*
- *Mapping/programming tools based on SDF (single job)*
- *Compositional approach for multiple jobs*

Philips
Research

PHILIPS

Thanks for your attention !!!

Questions ?

Philips
Research

PHILIPS

Improving Flexibility and Robustness in Consumer Terminals: QoS Control Framework

Laurenţiu M. Păpălău, Clara M. Otero Pérez and Elisabeth F.M. Steffens
Philips Research Laboratories Eindhoven (PRLE)
Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
E-mail: {laurentiu.papalau, clara.otero.perez, liesbeth.steffens}@philips.com

Abstract—**We are developing a QoS control framework for software media processing in consumer terminals. We present the initial implementation of the concepts and ideas of our QoS control framework in our QoS demonstrator. This QoS control framework contributes to flexibility and robustness. It contributes to flexibility by enabling a wide variety of applications to run concurrently by adjusting the delivered QoS. It contributes to robustness by adjusting resource usage to the available resources.**

Keywords— **Multimedia Systems; Architectures; QoS; Reservations; Control Framework**

I. INTRODUCTION

Media processing in software enables consumer terminals to become flexible, i.e. support a wide range of applications executing concurrently. However, in spite of Moore's law, they will remain heavily resource constrained, imposing high pressure on silicon cost and power consumption. Although required to be flexible in a resource-constrained environment, consumer terminals are still required to be robust.

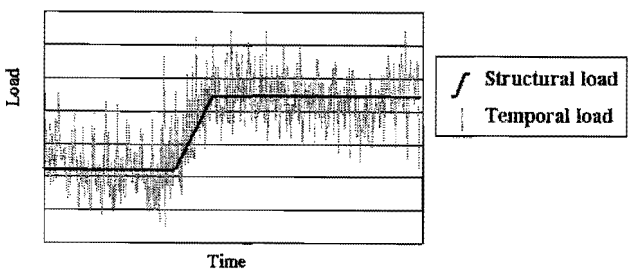


Figure 1: Load fluctuations

Furthermore, the load of media applications is highly dynamic due to the dynamic nature of the audio/video media content. This dynamic load jeopardizes the predictability and robustness of consumer terminals. We limit ourselves to the load generated by an incoming encoded digital video stream (MPEG-2). Two types of load fluctuations can be observed in a running terminal: structural and temporal load fluctuations (see Figure 1). Structural load fluctuations occur when the complexity of the incoming stream changes considerably, for instance at scene changes. Temporal load fluctuations are continuously happening (e.g. more difficult scenes require more resources).

Quality of Service (QoS)-based approaches are often used when dealing with limited and shared resources. QoS is defined as the "collective effect of service performance that determines the degree of satisfaction of the user of that service" (ITU-T Recommendation E.800-Geneva 1994). In consumer terminals, the delivered service is rendering audio/video content. Audio uses relatively few resources and is very sensitive to resource changes. Hence we do not scale down audio quality. Typical QoS parameters for video are picture quality, number of deadline misses, constant frame quality and latency. A typical example of latency aspect in the delivered QoS is the maximum delay between video and audio in order to still have lip synchronization.

Our QoS control framework adapts the QoS of media applications to maximize the delivered QoS of the system given the available resources.

To control the QoS provided by the applications, the media applications need to be scalable. Scalable media applications provide a set of discrete QoS levels with associated QoS values and resource requirements (e.g. processor requirements). For work on scalable applications see [1] and [2].

The allocation of resources to applications is done by our reservation-based resource manager (see [3] and [4]), which provides guaranteed resource budgets to media applications. The resource budgets contribute to robustness by preventing temporal interference between applications.

Reservation and adaptation are two means to provide robustness and flexibility. The contribution of reservation to robustness was already shown in [5]. The focus of this paper is on the contribution of adaptation to flexibility and robustness in the context of resource-constrained systems.

The remainder of the paper is structured as follows: Section II presents our QoS control framework, Section III presents the common building block for the QoS control framework, Section IV presents the implementation of the QoS control framework and Section V presents the conclusions and future work.

II. QoS CONTROL FRAMEWORK

The main responsibilities of our QoS control framework are: to provide system level QoS and to deal with the dynamics of the applications. The dynamics come from two sources. On one hand the user induces changes to launch new applications and also to adapt the QoS levels of individual applications in different situations. On the other hand there are structural and temporal load fluctuations.

System level mode changes (e.g. new application started, changes in the input source) require reshuffling of resources at the system level. Structural load fluctuations require cooperation between application and the QoS control framework. Temporal load fluctuations need to be addressed locally by the applications in order to stay within the provided resource budget.

Figure 2 depicts QoS control framework, which consists of a *Quality Manager* (QM) and a number of *Resource Consuming Entities* (RCEs), each with a local RCE Controller. Typically, all processing components contributing to the same application are grouped into the same RCE. The Resource Manager (RM) provides guaranteed budgets to the RCEs on request from the QM.

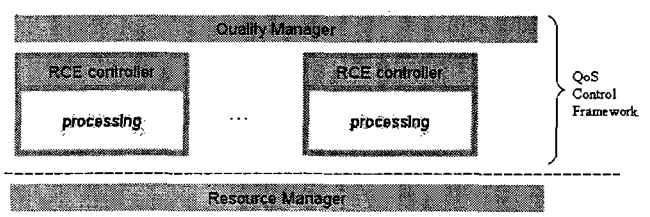


Figure 2 QoS control framework

The *QM* is responsible for choosing the appropriate QoS levels in such a way that the resource requirements for the set of applications fit the underlying platform. It is also responsible for adapting the RCEs' QoS levels based on monitoring information from the RCE Controller and RM. The QM deals with the structural load fluctuations by providing a short transition time between QoS levels. Whenever a structural load change is detected for an application, the QM adapts the QoS level of the specific application and, if required, also of the other applications in the system. By adapting the QoS levels of the applications, different configurations of applications are possible, so the QM contributes to flexibility of the system.

To choose appropriate QoS levels for the applications, the QM has a set of possibilities for each application. Based on given criteria a certain combination is chosen. If the combination is feasible (the required requirements fit on the underlying platform) then this combination is effectuated. Whenever the combination is not appropriate (because the QM monitors the system and detects so or receives notifications from an RCE Controller), a decision is taken to choose another combination of QoS levels for the applications.

The *RCE Controller* makes sure that the RCE performs acceptably within the limitations of its resource budget. For adaptation algorithms provided by an RCE controller see [6]. Each RCE has its own RCE Controller that deals with the temporal load fluctuations (e.g. graceful degradation) by adjusting the QoS value of the application. However, if it does not succeed, it informs the QM, which adapts the QoS level of the application and, if required, also the QoS levels of the other applications in the system. By staying within the resource budget, the RCE controller contributes to the robustness of the system, and more specifically to the intra-RCE robustness provided that the application does not crash, but it has mechanisms to still provide an output (skip frames, graceful degradation).

An RCE controller chooses parameters for the application in such a way that the required QoS level is provided. Based on monitoring information or notifications from the processing part, the RCE Controller detects when the parameters are no longer appropriate and tries to change them while still providing the same QoS level with its associated resource budget. If it does not manage to keep the same QoS level, it informs the QM.

III. CONTROL MODULE

The RCE Controller and QM share the same control module template[4]. The control module is designed to work in a hierarchy of control modules (see [7]). In general such a control module performs the following steps:

1. Select a feasible control setting
2. Effectuate the selected control setting
3. Monitor the controlled system to verify the adequacy of the control setting
4. Accept notifications from the controlled system(s)
5. Whenever needed as a result of step 3 or 4, adjust the control setting by going back to step 1

When step 1 cannot find an adequate control setting, notify the controller.

Steps 1-2 can be triggered by select and set calls from the controller, or by step 5. Step 3 determines the autonomous behavior of the controller with a characteristic frequency.

Based on these steps we can deduce the interfaces that such controller provides and requires (see Figure 3).

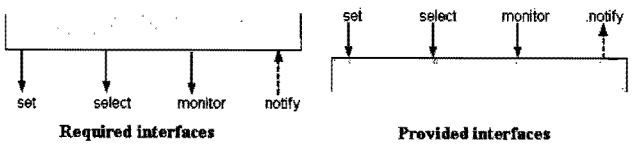


Figure 3 Control module interfaces

A control module serves as controlling module for the layer below and as controlled module for the layer above (see Figure 3). In our QoS control framework the QM is only a controlling module, whereas an RCE Controller is a controlled (by QM) and controlling (for RM) module. The RM and RCE processing part should provide the controlled module interfaces. The control framework can be extended since QM can be also a controlled module for a higher layer that takes into account not only the terminal, but also the network (see [4]).

IV. QoS DEMONSTRATOR

Our QoS demonstrator is an experimental platform that shows how the QoS control framework contributes to robustness and flexibility.

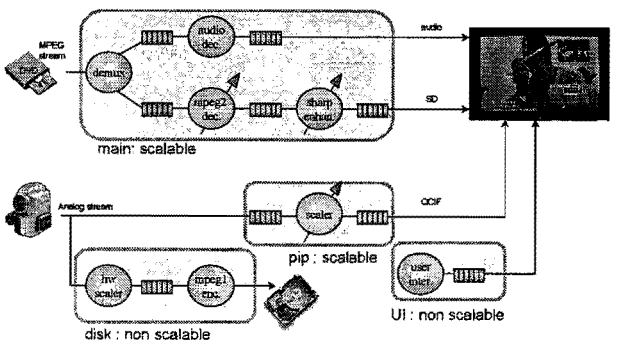


Figure 4: Demonstrator applications.

In the demonstrator, four applications execute concurrently (see Figure 4), namely: play DVD movie (Main), record analog video signal to disk (Disk), show the analog video input on a Picture in Picture (PiP), and show the User Interface (UI).

The main application is "play a DVD movie" (Figure 4). The MPEG2 stream coming from a DVD is demultiplexed and decoded (audio and video). A sharpness enhancement component further improves the video quality. The arrows indicate the scalable components (i.e. provide different QoS levels). For information on scalable components see [1] and [2].

Table 1 shows the minimum and maximum QoS levels of the demonstrator applications together with their resource requirements.

	Minimum		Maximum	
Application	**QoS level**	**Resource requirements**	**QoS level**	**Resource requirements**
Main	0	60%	7	80%
PiP	0	5%	3	15%
Disk	0	30%	0	30%
UI	0	5%	0	5%

Table 1 Applications QoS levels and resource requirements

Currently, we are instantiating the control module template for the QM and integrating it with the RM and a simple RCE controller. In our QoS demonstrator we show that the QoS control framework contributes to flexibility by the following scenario: by monitoring the RCE Controllers, the QM detects deadline misses and decides to switch to a lower QoS level and corresponding resource budget.

V. CONCLUSIONS AND FUTURE WORK

We showed how our QoS control framework contributes to flexibility by adapting the delivered QoS of the media applications to the available resources. Furthermore, the QoS control framework contributes to robustness (RCE Controller responsibility) by limiting the impact of quality derived from the temporal load fluctuations.

There are several research directions for future work:

- More complex algorithms for the RCE Controller will be integrated in the QoS control framework.
- Also more complex decision strategies for the QM will be implemented.
- The QoS control framework will be extended to treat the system level mode changes in a structured way.

The demonstrator is an experimental platform in which new concepts and ideas will be tested.

VI. REFERENCES

[1] C. Hentschel, R. Braspenning, and M. Gabrani, "Scalable algorithms for Media Processing", in *ICIP, IEEE International Conference on Image Processing, Proceedings*, Oct. 2001.

[2] T. Lan, Y. Chen, and Z. Zhong, "MPEG2 decoding complexity prediction and dynamic complexity control in a TriMedia processor", in *MMSP, IEEE Workshop on Multimedia Signal Processing, Cannes, France, Proceedings*, Oct. 2001.

[3] R. Rajkumar, K. Juwa, A. Molano, and S. Oikawa, "Resource kernels: A resource-centric approach to real-time and multimedia system", in *SPIE/ACM Conference on Multimedia Computing and Networking, Proceedings*, Jan. 1998.

[4] C.M. Otero Pérez, E.F.M. Steffens, P.D.V. Stok van der, A. Alonso, J.F. Ruiz, and R.J. Bril, "QoS-based Resource Management for Ambient Intelligence," in *Ambient Intelligence: Impact on embedded-system design, T. Basten, M. Geilen, H. de Groot (eds.)*. Kluwer Academic Publishers, 2003.

[5] C.M. Otero Pérez and I. Nitescu, "Quality of Service Resource Management for Consumer Terminals: Demonstrating the Concepts", in *14th Euromicro Conference on Real-Time Systems (ECRTS), Vienna, Austria, Proceedings Work in Progress Session*, M. González Harbour, Ed. June 2002, vol. 36/2002 of pp. 29-32.

[6] C.C. Wüst, E.F.M. Steffens, and W.H.J. Verhaegh, "Adaptive QoS Control for Real-Time Video Processing", in *Work in Progress Session of the 15th Euromicro Conference on Real-Time Systems (ECRTS), Proceedings, Porto, Portugal, pp. 49-52*, 2003.

[7] R.J. Bril, C. Hentschel, E.F.M. Steffens, M. Gabrani, G.C. van Loo, and J.H.A. Gelissen, "Multimedia QoS in consumer terminals", in *IEEE Workshop on Signal Procesing Systems (SIPS), Antwerp, Belgium, Proceedings*, Sept. 2001.

AN SIMD-VLIW SMART CAMERA ARCHITECTURE FOR REAL-TIME FACE RECOGNITION

R. Kleihorst[1], H. Broers[2], A. Abbo[1], H. Ebrahimmalek[1], H. Fatemi[3], H. Corporaal[3] and P. Jonker[4]

[1]Philips Research Laboratories, Eindhoven,NL
[2]Philips CFT, Eindhoven, NL
[3]Eindhoven University of Technology, NL
[4]Delft University of Technology, NL
Email: richard.kleihorst@philips.com

ABSTRACT

There is a rapidly growing demand for using smart cameras for various applications in surveillance and identification. Although having a small form-factor, most of these applications demand huge processing performance for real-time processing. Face recognition is one of those applications. In this paper we show that we can run face recognition in real-time by implementing the algorithm on an architecture which combines a parallel processor with a high performance digital signal processor. Everything fits within a digital camera, the size of a normal surveillance camera.

I. INTRODUCTION

Recently, face detection and recognition is becoming an important application for smart cameras. Face detection and recognition requires lots of processing performance if real-time constraints are taken into account[1].

Face *detection* is the detection of faces in the scene from video data, it is usually done using color and/or feature segmentation. Face *recognition* is de actual recognition of the person based on the pixels that span the face region found in the detection process. Face recognition is usually performed by either neural network matching or by feature measurement and matching through a database. For robust recognition, the face needs to be at a proper angle and completely in front of the camera.

What we want to show in this publication is that it is possible far smart camera architectures to achieve good, real-time face recognition results. A "smart camera" is hereby defined as a stand-alone device which is prefferably programmable with a size not bigger than a typical video surveillance camera. In our situation it is programmed in such a way that video goes in and the names of recognized people come out. A speech synthesizer output takes care of this and it will also advice the persons in the scene to look straight in the camera and/or to come closer if the person is detected but the recognition reliability is not high enough for positive identification.

The platform we suggest for face recognition is the Intelligent Camera (INCA+) produced by Philips CFT [2] as shown in Figure 1. This camera houses a CMOS sensor, a parallel processor for pixel crunching and a DSP for the high level programs. We will show in this paper that this platform is ideal for face recognition.

The contents of the paper is as follows: In Section II we explain about the architecture of camera, In Sections III and IV respectively, we explain about the algorithm that we used for face detection and recognition. The results are given in Section V and conclusions are drawn in Section VI.

II. MOTIVATION OF THE ARCHITECTURE

Face recognition consists of a face detection and face recognition part. In the detection part face blobs (groups of pixels spanning a face) are detected in the scene and they are forwarded to the face recognition process where the found face blobs are matched to a database with a set of stored faces in order to recognize and identify them.

These two parts of the algorithms work on different data structures. While the detection parts works on all pixels of the captured video and is pixel oriented (low level image processing), the recognition

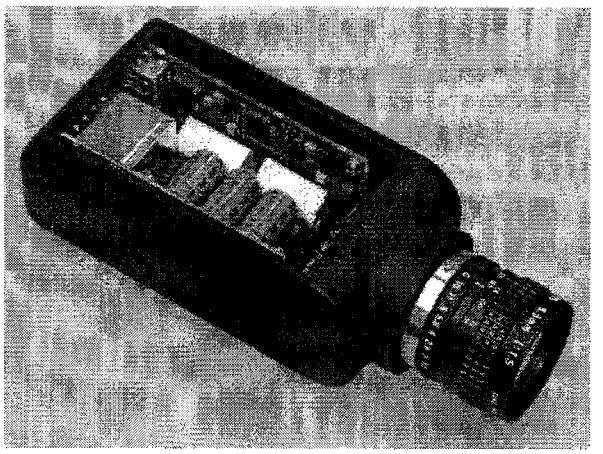


Fig. 1. INCA camera

part is working on face objects and is face oriented (high level image processing). The detection part has to do similar operations for all pixels in the scene to determine if or not the pixel belongs to a face-blob. While we have a high amount of pixels in a life video stream, the operations are simple and similar for each pixel, allowing data-level parallelism.

The data rate in the recognition part is not that high, it only works on a few hundred faces per second but it has a high amount of operations in an iterative way while a database is "scanned". Because of the higher complexity of the instructions and the combination with an operating system, this part of the algorithm is best mapped on a task-parallel architecture.

The different aspects of the two algorithmic tasks have made us to choose for a dual processor approach where the low-level image processing approach of the face detection part is mapped on a massively parallel processor "Xetal" [3] working in SIMD (Single Instruction Multiple Data) mode. The high level image processing part of recognition is mapped on a high-performance fully programmable DSP core "TriMedia" [4]. This DSP has a VLIW (Very Long Instruction Word) architecture where instruction fetch, data fetch and processing are performed in a pipelined fashion.

For the defined task the two processors can be simply connected in series as shown in Figure 2. The Xetal does face detection, the TriMedia does face recognition and the operating system also runs on Trimedia.

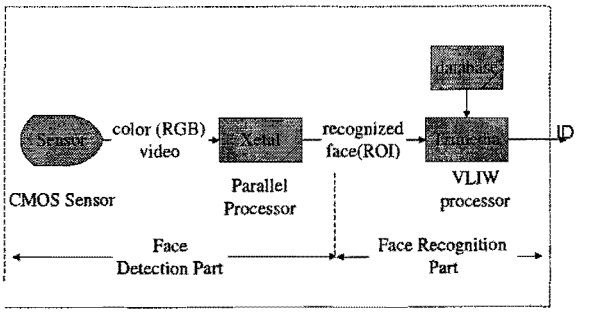


Fig. 2. Architecture of the INCA Camera

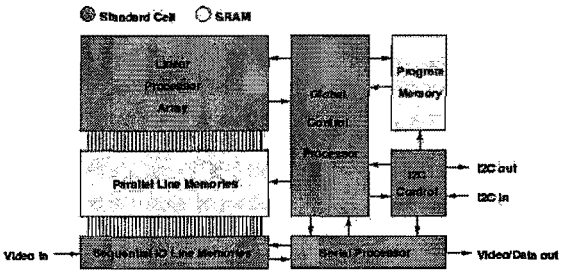


Fig. 3. Xetal Architecture

First part of the architecture is CMOS sensor, it can take up to 30 frames per second with a resolution of 640 × 480 pixels. The SIMD Xetal processor exploits massive parallelism. It contains 320 pixel level processors and each pixel processor is responsible for 2 columns of the image. It can handle up to 1041 instructions for each pixel. It has 16 line memories to save information [3]. Figure 3 shows the architecture of the Xetal processor in more detail. This processor directly reads the pixels from the CMOS image sensor and performs the face detection part. Coordinates and subregions of the image where prospective faces are found are forwarded to the TriMedia. The TriMedia exploits limited instruction level parallelism; it can handle 5 operations in parallel. This processor scales and normalizes the subregions and matches them to the faces in his database. In the fashion of a real "smart" camera, only IDs are reported. These IDs are send to a speech synthesizer that greets the person recognized or asks the person to identify himself when not recognized.

III. FACE DETECTION

In the face detection part we take an image from the sensor and detect and localize an unknown num-

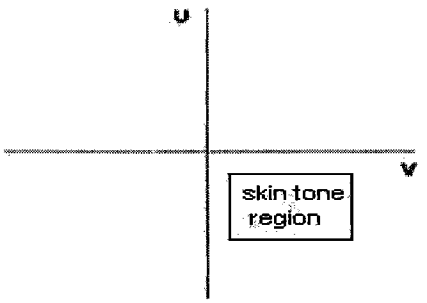


Fig. 4. Skin region in UV Spectrum

ber (if any) of faces. Faces are found by colour specific selection. By removing too small regions and enforcing a certain aspect ratio of the selected region of interest (ROI) the detection becomes more reliable.

We detect skin parts in the image by searching for the presence of skin-tone coloured pixels or groups of pixels. The representation of pixels as they are delivered by the colour interpolation routines from the CMOS sensor image are in RGB form . This is not very suitable for characterizing skin colour. The components in RGB space not only represent colour but also luminance, which varies from situation to situation. By going to a normalized colour domain such as YUV, this effect is minimized [5], [6]. The YUV colour domain is more suitable for the detection because it separates the luminance (Y) with the colours (UV). Y value can vary from 0 to 255 whereas the U and the V can have values from -128 to 128. A continuous auto white-balance and exposure system ensures that the color spectra are well defined, even under coloured lighting conditions.

By using the YUV colour domain not only the detection has become more reliable but the skin tone indication has become easier, because skin tone can now be indicated in a 2 dimensional space. We defined the skin tone region as a square in the UV spectrum. Everything in this region passes as skin-pixel and a result is shown in Figure 5. Some checks on Y are also performed to filter out the very high and low brightness regions where U and V are ill-defined.

To increase the reliability, the field of detected skin-tone and non skin-tone pixels is filtered using a 7×7 erosion and dilation filter. This filter removes small

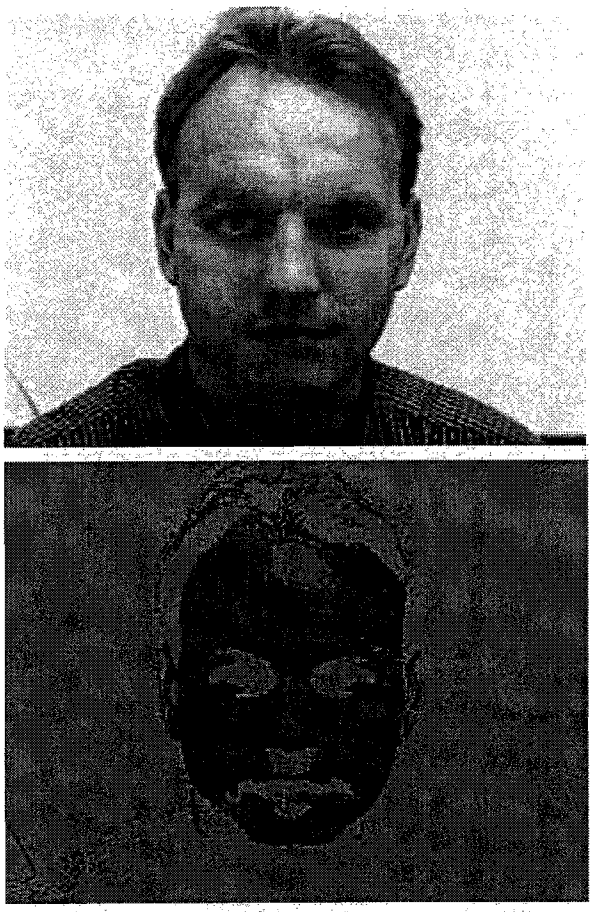


Fig. 5. The input face and classification of skin pixels, note the number of small detected regions that have to be removed.

pixel regions that are too small to be faces in the scene. The result for 3 faces in the scene is shown in Figure 6. Eventually all three faces will make up three detected regions.

By imposing a (width:height) ratio on the detected blobs of around (1:1.6), the face regions are separated from other skin-coloured blobs like hands. Final result is a region of interest spanning only the face such as shown in Figure 7.

Horizontally and vertically through the face, a gray-level projection is performed whose minima enable the detection of the position of the eyes in order to normalize the face blob around the eye positions before feeding it to the recognition phase [7]. See Figure 9 for the results and the principle. Xetal does the horizontal projection and Trimedia does the vertical projection and finds the minima.

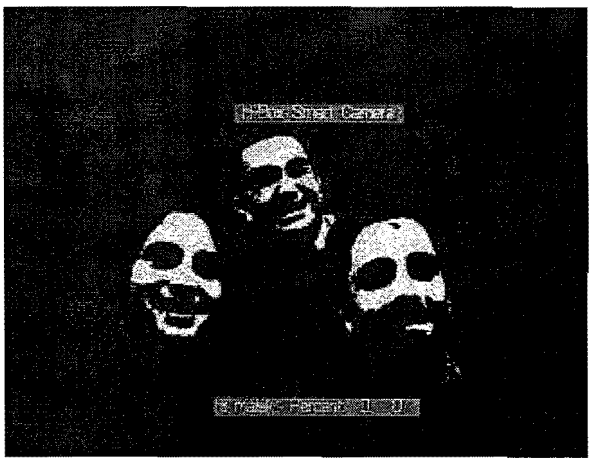


Fig. 6. Skin Tone Detection Result

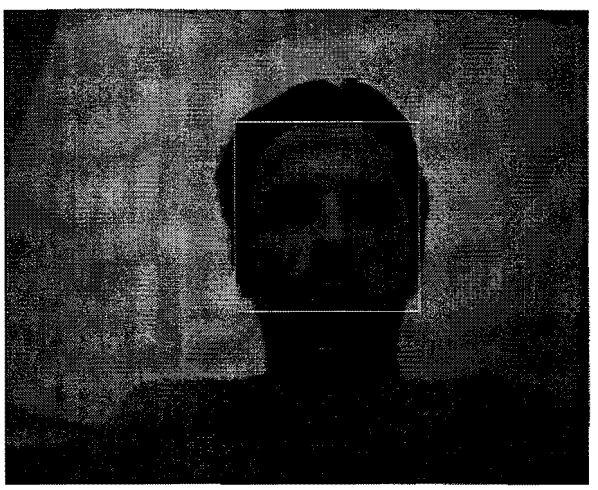


Fig. 7. Region of Interest result

Next to this projection data, the face detection part only sends luminance and coordinates of the face to the recognition part as defined in Figure 8. The Trimedia will only address the relevant regions. This reduces the data content significantly.

IV. FACE RECOGNITION

This section introduces the neural net face recognition process. As input for the recognition process, the face blob detected in the previous section is normalized around the eye positions and than identified with respect to a face database.

For this purpose a Radial Basis Function (RBF) neural network is used [8]. The reason behind using an RBF neural network is its ability for clustering similar images before classifying them. RBF based

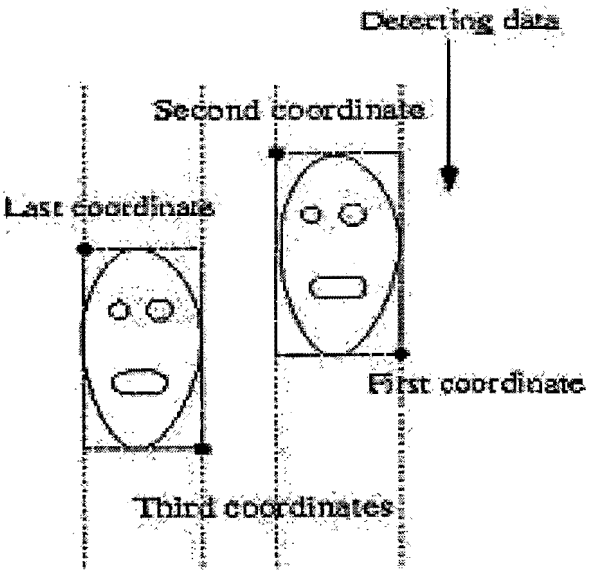


Fig. 8. Coordinates of faces found in the input image

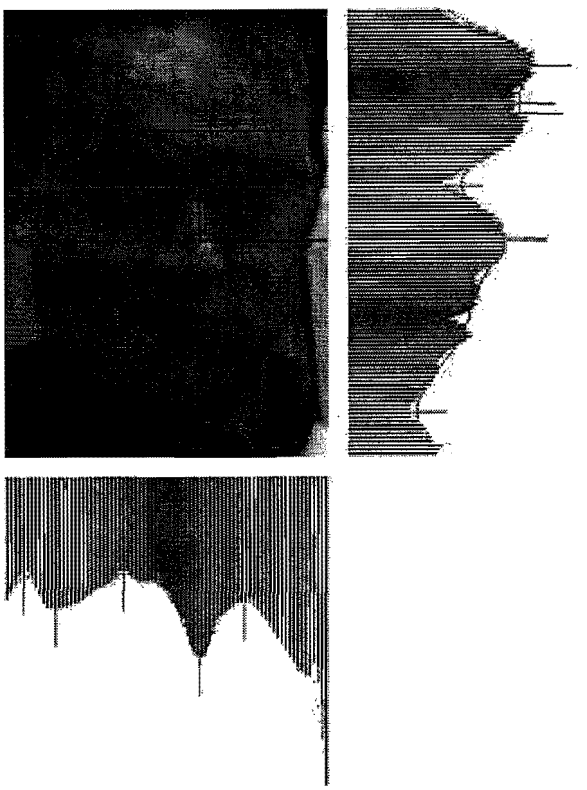


Fig. 9. This image shows the projection data and the face blob region that is send to the Trimedia, the latter processor detects the eyes and nose for normalization as indicated by the crossing lines.

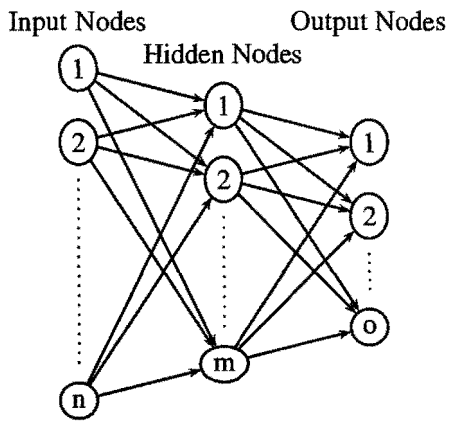


Fig. 10. *Architecture of an RBF Neural Network*

clustering received wide attention in the neural networks community. Apart from good clustering capabilities RBF networks have a fast learning speed, and a very compact topology.

IV-A. Architecture of RBF Neural Network

An RBF neural network structure is demonstrated in Figure 10. Its architecture is similar to that of a traditional three-layer feed forward neural network. The input layer of this network is a set of n units, which accepts the elements of an n dimensional input feature vector (here, the RBF neural network input is the face which is gained from the face detection part. Since it is normalized with a $64 * 72$ pixel face, it follows that $n = 4608$).

The input units are completely connected to the hidden layer with m hidden nodes. Connections between the input and the hidden layers have fixed unit weights and, consequently it is not necessary to train them. The purpose of the hidden layer is to cluster the data and decrease its dimensionality. The RBF hidden nodes are also completely connected to the output layer.

The number of outputs depends on the number of people to be recognized (for example, for 100 persons $o = 100$). The output layer provides the response to the activation pattern applied to the input layer. The change from the input space to the RBF unit space is nonlinear, whereas the change from the RBF hidden unit space to the output space is linear.

The RBF neural network is a class of neural networks, where the activation function (basis function) of the hidden units is known by the distance between the input vector and a prototype vector. The activation function of the RBF hidden node is stated as follows [9]:

$$F_i(x) = G_i(\| x - c_i \|^2 / \sigma_i), \quad i = 1, 2, \ldots, m \tag{1}$$

where x is an n-dimensional input feature vector (normalized face $64*72$), c_i is an n-dimensional vector called the center of the RBF hidden node, σ_i is also an n-dimensional vector called the width(also called radius) of RBF hidden node and m is the number of the hidden nodes. Normally, the activation function G of the hidden nodes is selected as a Gaussian function with mean vector c_i and variance vector σ_i as follows:

$$F_i(x) = e^{\left(-\frac{\|x-c_i\|^2}{\sigma_i^2}\right)}, \; i = 1, .., m \tag{2}$$

Because the output units are linear, the response of the k'th output unit (among the o number of outputs) for input x is given as:

$$Out_k(x) = B_k + \sum_{i=1}^{c} F_i(x) * W(i,k), \; k = 1, .., o \tag{3}$$

where $W(i,k)$ is the connection weight of the i'th RBF hidden node to the k'th output node and B_k is the bias of the k'th output.

IV-B. Using RBF neural network

The first step in face recognition is normalizing the region of interest(as shown in Figure 7) to the size of the faces stored in the identification database($64 * 72$ pixels) and after that feed them to the neural network input. Subsequently, we calculate the output for each person, and we consider the maximum value between the outputs and report that as the recognized person. Figure 11 shows the main kernel for using the RBF neural network.

V. MEASUREMENTS AND PERFORMANCE

In this section we evaluate the performance of our algorithm. Since the face recognition, and not the detection part, turned out to be the major bottleneck we

```
// compute output of hidden node
L1:
 for 0 < i < Number_Hidden_Node{
  sum =0
  for 0<j< Number_Input_Node(64*72=4608){
     temp = data[j]-center_value[i][j]
     temp = temp * temp
     temp = temp / sigma_value[i][j]
     sum = sum+temp
  }
  out_hiddennode[i] = exp(-sum)


// compute output
L2:
 for 0 < i < Number_Output(5 person){
  sum =0
  for 0 < j < Number_Hidden_Node(20){
     sum = sum +
      (out_hiddennode[i][j]*weight[i][j])
  }
  sum = sum + bias_value[i]
  output[i] = temp
 }
```

Fig. 11. *Kernel for RBF Neural Network*

concentrate on that part first. At the end of this section we evaluate the overall performance and recognition rate.

V-A. Face Recognition

The algorithms described for using the RBF neural network have certain demands on the processing power, bandwidth and flexibility of the architectural template. To measure the performance, we first need to extract the kernel loops and loop-nests, which are done in the RBF neural network. If we take the example of face recognition, the input is a normalized face ($64 * 72$ pixels, hence 4608 inputs), there are m hidden nodes (resulting in $4608 * m$ weights between the input and hidden layers), and o output nodes, depending on the number of people to be recognized (hence $m * o$ weights between the hidden and output layers).

V-B. Adapting for Real-time performance

We observed that most of the running time of the algorithm was spent on calculating the output of hidden nodes and calculating the output (see Figure 11). For example, if the number of hidden nodes is equal to 20 and we want to recognize faces of five persons, the number of executed instructions on a Trimedia (166 Mhz.) is about $12 * 10^6$, and the number of cycles(taking memory delays into account) is about $15 * 10^6$, which corresponds to $90ms$. This is far from real-time, therefore we employed several optimizations like:

- Replace all division operations in the program.
- Use single precision floating point instead of double precision.
- Use local variables instead of global variables.
- Perform loop-unrolling.

Then the number of executed instructions is reduced to $4 * 10^5$ and the number of cycles is reduced to $7 * 10^5$(thus, resulting in 4.2 ms execution time).

V-C. Complexity

The execution time in the RBF loopnests is related to m, n, and o (see Figure 11). The time complexities for the first ($L1$) and second ($L2$) loopnests are:

$$T_{L1} = O(m.n) \quad , \quad T_{L2} = O(m.o) \qquad (4)$$

Therefore, the total time complexity is given by:

$$T_{(m,n,o)} = O(m.n + m.o) = O(m.(n + o)) \qquad (5)$$

It is easily seen that the memory size required for allocating all variables has the same complexity:

$$M_{(m,n,o)} = O(m.(n + o)) \qquad (6)$$

Because m is independent of n and o (m is more or less constant, and equal to the number of characteristic in a face), execution time and memory size are linear in n and o [10].

V-D. Overall practical performance

Our algorithms have been mapped to a handheld camera device as shown in Figure 1. After programming using a host computer and a firewire or ethernet link the camera starts running the face recognition application. Because faces are recognized at video rate and with more possible faces per frame (a maximum recognition rate of 230 faces per second is possible), an operation system running in the camera has to control the reporting process. The operating system obtains the IDs of the recognized person and monitors the reliability of recognition as reported by

the face recognition part. If this is high enough, a person is positively identified and will also not be reported in subsequent frames until he/she leaves the scene or another person shows up.

A connected speech synthesiser reports the name of the identified person, asks an unknown person to identify himself or instructs the persons in the scene to look at or approach more towards the camera.

The overall performance we reach ranges from a recognition rate of 97% with a false detection rate of 1 out of 20 to a recognition rate of 90% with a false detection rate of 1 out of 50 dependent on the settings. These numbers are for a real-time (up to 230 faces per second) stand-alone system with 5 stored "identifiable" faces.

VI. CONCLUSIONS AND FUTURE WORK

Face recognition is becoming an important application for smart cameras. However, up till now, the processing required for real-time detection, prohibits integration of the whole application into a small sized, consumer type of camera. This paper showed that by:

1. Proper selection of algorithms, both for face detection and recognition,
2. Adequate choice of processing architecture, supporting both SIMD and ILP types of parallelism,
3. Tuning the mapping of algorithms to the selected architecture,

this integration can be achieved. We implemented the algorithms on a small smart camera. As a result we can recognize one face per 4.2 ms, when we are searching for 5 persons, with 90% recognition rate and only 1% failure rate.

Future research will focus on further tuning the mapping of the algorithms, e.g. by replacing floating point operations with fixed point, trying other (cheaper) activation functions (see eq. 2), and further parallelization of the RBF neural network. This should allow for further speedups needed when searching in much larger databases that can contain large numbers of identifiable faces.

A major part of future work will also be to use the audio feedback in a better way, and in increasing the reliability of recognition which is too low now for professional systems [11]. Although the processing time will probably increase, we believe that the performance will be highly sufficient.

VII. REFERENCES

[1] B. L. E. Hjelmas, "Face detection: a survey," *Computer Vision and Image Understanding*, vol. 83, pp. 236–274, 2001.

[2] Centre For Industrial Technology. http://www.cft.philips.com/, 2003.

[3] A. Abbo and R. Kleihorst, "Smart cameras: Architectural challenges," in *Proceedings of ACIVS 2002 (Advanced Concepts for Intelligent Vision Systems)*, (Gent, Belgium), 2002.

[4] TriMedia Technologies. http://www.trimedia.com, 2003.

[5] T. Majoor, "Face detection using color based region of interest selection," tech. rep., University of Amsterdam, Amsterdam, NL, 2000.

[6] R.L.Hsu, M.Abdel-Mottaleb and A.K.Jain, "Face detection in color images." http://www.cse.msu.edu/~hsureinl/facloc/index_facloc.html, 2003.

[7] F. Zuo and P. H. de With, "Fast human face detection using successive face detectors with incremental detection capability," *Proc. SPIE*, no. 5022, 2003.

[8] J. Haddadnia, K. Faez, and P. Moallem, "Human face recognition with moment invariants based on shape information," in *Proceedings of the International Conference on Information Systems, Analysis and Synthesis*, vol. 20, (Orlando, Florida USA), International Institute of Informatics and Systemics (ISAS'2001), 2001.

[9] Y.-H. Hu and J.-N. Hwang, eds., *Handbook of neural network signal processing*. CRC Press, 2002.

[10] H. H.Fatemi, R.P.Kleihorst and P.Jonker, "Real time face recognition on a smart camera," in *Proceedings of ACIVS 2003 (Advanced Concepts for Intelligent Vision Systems)*, (Gent, Belgium), 2003.

[11] Electronic Privacy Information Center. http://www.epic.org/privacy/facerecognition, 2003.

Architecture for Multi-Client Multi-Channel Compressed Video Streaming

R.G.J. Wijnhoven, M.C. Jacobs, P.H.N. de With, E. Jaspers
Eindhoven University of Technology / Bosch Security Systems B.V.,
Eindhoven, The Netherlands
E-mail : R.G.J.Wijnhoven@student.tue.nl

Abstract — **In this paper we describe an architecture for a multi-client, multi-channel video streaming server targeted at the security market. For each connected client, the best selection of available compressed video streams per channel is made, optimizing the perceived quality of the video channels requested by the client. We propose techniques to scale the bitrate per video stream and introduce a scheduling scheme to select the best video streams for a given set of channels and bandwidth constraints.**

Keywords — **Architecture; streaming video; multi-client; multi-channel; scheduling; bitrate scaling; restricted shortest path problem**

I. Introduction

In the video security market, digital video recorders (DVR) have been available for some time. DVRs store video from multiple channels (cameras) and offer remote display of recorded and live video over a network connection. Multiple clients can connect to the system and view video simultaneously (see Figure I).

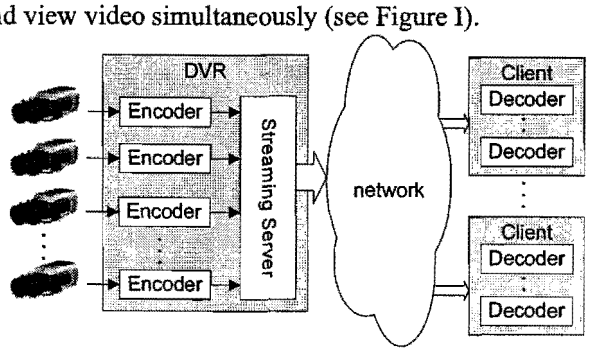


Figure I: Use case.

DVRs often use still-image-based compression techniques like wavelet- or JPEG-compression. To increase video quality and storage times, there is a shift to compression techniques that exploit the temporal correlation of video. These techniques are called inter-coded compression and are often based on one of the MPEG standards [4]. MPEG defines a group of pictures (GOP) as one independently decodable frame (I-frame) followed by a number of inter-predicted frames (P-frames). Such a predicted frame is encoded using the difference between the current frame and the motion-compensated previous frame. Consequently, higher compression factors can be achieved. However, due to the dependencies between encoded frames (P-frames), all preceding data in a GOP needs to be transmitted and decoded in order to decode a certain frame. Skipping frames without considering the coding scheme will result in errors in the decoded video. Thus, it is hard to adapt to changing network conditions compared to simply dropping frames when using still-image-based compression. Thus, scalability of inter-coded video is less straightforward.

MPEG defines scalable profiles like the Fine Granularity Scalability profile (FGS), that include spatial, SNR and/or temporal scalability. We do not consider these profiles, because they are too computationally intensive.

Most systems currently on the market that transmit inter-coded video are single-client based, or broadcast a single video channel to multiple-clients. Furthermore, they don't offer scalability in bitrate: only a limited number of (pre-)compressed streams is available. Examples are streaming video servers on the Internet: watching movie trailers (on-demand) or broadcasts of concerts (live).

A proposal for multi-channel video encoding for broadcast over a network with fixed bandwidth can be found in [1]. A complexity measure per video channel is proposed, to partition the total bandwidth over the total number of channel encoders. The video encoders are dynamically controlled to obtain the target bitrates. This solution cannot be used, since the bandwidth is not fixed and the encoders cannot be dynamically controlled (see Section III).

II. Document Layout

Chapter III describes the problem. The proposed architecture and its components are discussed in Section IV. The scheduling algorithm, used to select the best streams for a given set of channels, is explained in Section V. This paper ends with conclusions and future work.

III. Problem Definition

In the considered system, the following constraints are identified:

- Bandwidth limitation by the server (e.g. set by the system administrator) or by the network constraints.
- Multiple clients can be connected, each viewing multiple channels.
- Because one source coder can have multiple consumers (multiple connected clients and storage on disc) the parameters of the source coders cannot be changed dynamically. This makes the system independent of the used inter-coded based source coder. Also, the target bitrate (CBR/VBR) per video stream is specified.
- There is limited computing power available.
- Video quality requirements for surveillance purposes are different from consumer entertainment; lower resolutions and frame rates are accepted, also to increase storage times.

The objective of the streaming server is to obtain the best perceived quality for connected clients, given the mentioned constraints.

IV. Architecture

Figure II shows the overall architecture of the video server. It is divided in four main components: the source reader, the bitrate scaler, the dispatcher and the controller. Each will be described separately.

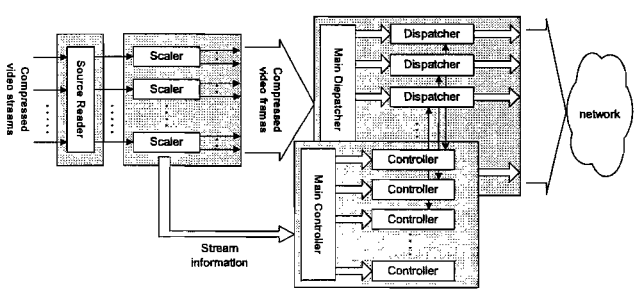


Figure II: Architecture of the multi-client, multi-channel video streaming server.

The actual transmission of data over the network and the measurement of network bandwidth is beyond the scope of this paper. Information on this subject can be found in [2] and [3].

A. Source Reader Component

The source reader receives compressed video frames (for all channels) from the source and transmits them at the display rate. Each video channel can have multiple source streams. Inputs of this component can be real-time video encoders, but also disc readers (in case of stored video).

The video streams are sent to the bitrate scalers. These scalers create streams with lower bitrates (see Section B).

B. Bitrate Scaler Component

This component contains a scaler for each source stream. Each scaler is able to scale a source stream to a limited number of output streams, each having a different bitrate. The bitrate of the output streams is calculated and together with other information about each stream sent to the controller component.

Let us consider methods for decreasing the bitrate of the video streams:

- Transcoding: video streams are partially decoded and consecutively encoded at different rates. Although this solution offers graceful degradation of video quality for a fine granular scalability in bitrate, the required system resources could exceed the available budget.
- Using multiple compressed source streams per video channel, each with a different frame rate. These streams are created before entering the source reader. They require extra processing costs in the encoders, although streams at low frame rates are relatively computationally inexpensive.
- Frame dropping: decode only some of the P-frames of each GOP, and display only the ones at regular intervals. See the example in Figure III, where the frame rate is reduced with a factor of three, while dropping only two out of six frames. Fractions of the original frame rate can be obtained, yielding a relatively small bitrate reduction. The method is computationally expensive for the decoding at the client, in comparison to decoding a regular stream at the same frame rate. The simplest case of frame dropping is the case where all P-frames are dropped, and only I-frames are transmitted. If even more reduction in bitrate is needed, also I-frames can be dropped.

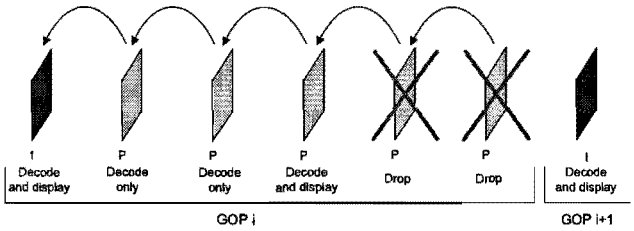


Figure III: Frame dropping

For our implementation, we choose not to apply transcoding because of processing constraints. Per channel we use a limited number of predefined streams, having different bitrates, and implement frame dropping. The number of encoders and the choice of encoder parameter settings for the streams is very important. However, this is out of the scope of this paper.

C. Dispatcher Component

The dispatcher component contains one main dispatcher, and a client dispatcher for each connected client. The main dispatcher receives compressed frames from the scaler component. Each frame is part of a video stream and is sent to each client dispatcher that requires the stream. The selection of the streams is calculated and updated by the corresponding client controller. Subsequently, each client dispatcher concatenates all received frames and forwards them to the network buffer for transmission.

D. Controller Component

As in the dispatcher component, one main controller is available, and a client controller for each connected client. The main controller receives information about the streams from the bitrate scaler component. When the main controller receives information regarding a change in available bandwidth, rescheduling is started. The main controller divides the available total bandwidth over the client controllers, which independently execute the scheduling algorithm (see Section V).

V. SCHEDULING ALGORITHM

Inputs for the scheduling algorithm are the available network bandwidth for the client, the channels that are requested by the client and the available streams per channel. Properties of each stream are the bitrate, frame rate, resolution, GOP-length, time to next I-frame, I-/P-frame size ratio and the currently selected stream for this channel. For each channel, a stream needs to be selected, giving the overall optimal quality over all channels, while the total bitrate does not exceed the bandwidth budget.

To solve this problem, we construct a tree, where each level in the tree represents a channel. Each edge represents a stream and has two values: a feasibility value (bitrate) and an optimality value (quality value). Each child node of an edge will be the parent node for the next channel. Each leaf represents a stream combination for the used channels. An example of a tree with two channels, each having three different streams, is shown in Figure IV.

The algorithm needs to select the optimal solution from the set of leaves. Each leaf represents a combination of streams (a unique path from the root to the leaf). The sum of the bitrates over the path needs to be less than the bandwidth budget. From this subset we select the path with the highest quality value.

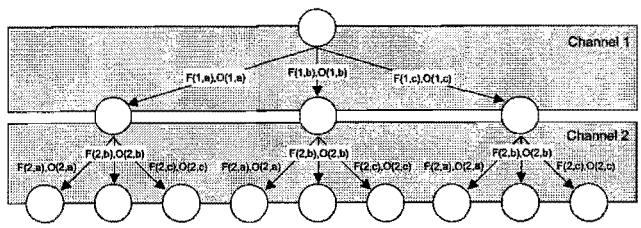


Figure IV: Tree for scheduling, with F(c,s) the feasibility value of channel c and stream s and O(c,s) the optimality value.

This problem is know as the Restricted Shortest Path (RSP) problem and is NP-complete. Extensive research in this field has been done in the area of QoS network traffic routing. An overview of (heuristic) selection algorithms is given in [5].

A. Implementation

The streams are stored in the tree, sorted to bitrate. We use a depth-first search to calculate all the feasible paths, starting with the lowest bitrate streams. When the sum of bitrates of streams on a path fits the available bandwidth, the path is stored. If the bandwidth budget is exceeded, we stop the algorithm because the streams are sorted to bitrate.

From the remaining set of feasible paths, we select the path with the highest quality value.

B. Stream Quality Value

We propose to use a combination of stream properties to calculate the quality value. The following properties are used:

- Frame rate: higher frame rates correspond to a higher perceived quality.
- Bitrate: when two streams have the same frame rate, the stream with lowest bitrate will have a higher quality value (assuming an equal perceived quality of the decoded video).

- I-/P-frame ratio: in case of motion, P-frames have a relatively larger size. For video with motion, higher frame rates are preferred, compared to streams with little motion.
- GOP-length: closely related to time to next I-frame. If GOP-length is high, time to next I-frame will also be high on average.
- Time to next I-frame: if the bandwidth budget decreases, we want to send less bits to the network, so a higher time value is desired. On the other hand, for very low frame rates, a lower time value is preferred.
- Currently selected stream: it is preferred not to switch streams for a channel too often.
- Resolution: higher resolution leads to higher perceived quality.

VI. CONCLUSIONS

We proposed an architecture for an embedded streaming server that is independent of the used source coders and type of network connected. The number of input sources and bitrate scaling methods is variable, as is the number of connected clients. A scheduling algorithm selects the optimal combination of streams with graceful degradation, depending on the used bitrate scaling methods.

First results of a PC-based software implementation look promising and provide a good base for further research.

VII. FUTURE WORK

The performance of the scheduling algorithm is heavily dependent on the choice of the quality value calculation. Therefore, more work needs to be done, to find a good choice for combining the various properties of a stream into one quality value.

Although each client scheduling algorithm will return the optimal stream selection for the given bandwidth budget, the overall solution may not be optimal, since the sum of the client bitrates may be much less than the overall bandwidth budget in the main controller. Therefore, the algorithm has to be extended to find a globally optimal solution.

REFERENCES

[1] G.J. Keesman, *Multi-Program Video Data Compression*, Ph.D. Thesis, TU-Delft, October 10, 1995

[2] N.G. Feamster, *Adaptive Delivery of Real-Time streaming Video*, M. Eng. Thesis, Massachusetts Institute of Technology, May 2001

[3] W. ten Kate, *Internet Streaming: Transporting Continuous Media*, in Proc. Embedded Video Streaming Technology (MPEG-4) and the Internet, IEEE, EESI, Eindhoven, The Netherlands, Dec. 20, 2003, Page(s) 119-144

[4] Moving Picture Experts Group (MPEG) Homepage, http://mpeg.telecomitalialab.com/

[5] F. Kuipers, P. van Mieghem, *An overview of constrained-based path selection algorithsm for QoS routing*, Communications Magazine, IEEE , Volume: 40, Issue: 12 , Dec. 2002, Page(s): 50 –55

A Scenario-Based Approach for Predicting Timing Properties of Real-Time Applications

Egor Bondarev
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, Netherlands
Phone: +31 (0)40 247 2480
E-mail : E.Bondarev@tue.nl

Peter H.N. de With
LogicaCMG/TU Eindhoven
P.O. Box 513, 5600 MB Eindhoven, Netherlands
Phone: +31 (0)40 247 8210, +31 (0)40 29 57 777
E-mail : mailto:P.H.N.de.With@tue.nl

I. Introduction

Embedded systems are often characterized by two closely coupled properties: limited resources and real-time constraints for executing running applications. The limitation of resources, such as memory size, memory bus and processing power, makes it more difficult to guarantee the real-time execution of applications. However, having that guarantee is crucial for e.g. multimedia devices.

During the design phase, in order to ensure that an application will fit on a target device, it is important to determine or predict the resource usage of an application. The resource-usage *prediction* is a technique to estimate the amount of consumed resources by analyzing the design and/or implementation of an application.

In the Space4U[1] project [2], which is an extension of the ROBOCOP project [1], a component-based architectural framework was introduced for the middleware development of high-volume embedded devices. Component-based development complicates the resource-usage prediction per application, because actual resource consumption is distributed over individual components. In this paper, we propose a technique for predicting the timing property of a component composition, also called a component assembly.

The key problem of such a *predictable assembly* is to first find and express a component's timing property, and, second to combine them in order to make predictions over a composition of those components. It should be noticed that there is a clear difference between the component and application timing properties. In case of component development, the designer deals with metrics such as worst-case, mean-case and best-case execution times per function. In case of component composition or application development, the designer focuses on finding the following properties: end-to-end response time and processor utilization bound of an application.

II. Component Dependencies

In the Space4U project and in this paper, we propose a predictable assembly technique that allows the translation of an already known property of components into a property of an application assembling these components. This technique is completing and refining the coarse scenario-based predictable assembly model [3] that was proposed in the ROBOCOP project.

A primary benefit of the ROBOCOP framework is that a component designer specifies not only provided interfaces, but also required interfaces of a component. While provided interfaces help an application developer to find a component that would do the work, the required interfaces specify what other components a particular component may depend upon. Finally, the provided and required interfaces allow the application developer to describe explicitly static component dependencies via interfaces within an application (see the simple decoder example in Figure 1).

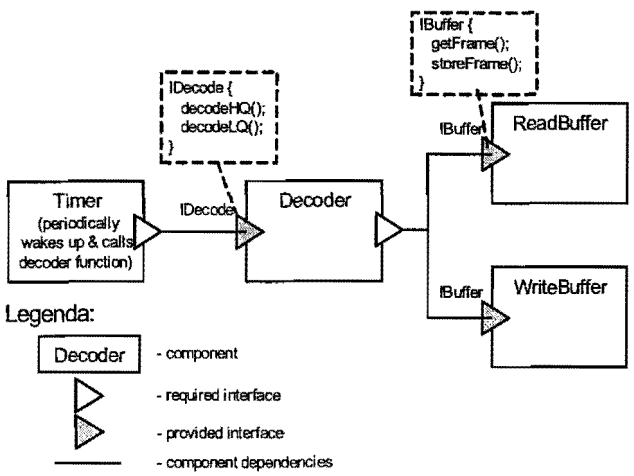


Figure 1. Static component dependencies for decoder

[1] Space4U is part of the ITEA research programme funded by the European Union.

Normally, each interface encapsulates a set of functions. Taking into account interface dependencies and, analyzing the most commonly used scenarios of an application, a developer can identify function call sequences per task. Following this, if the timing properties (WCET) of those functions involved in a task execution are given by a component developer, we can find the timing property of the complete task, while considering the commonly used scenario. The scenario can be represented by a sequence chart diagram (see the "high-quality decoding" scenario in Figure 2).

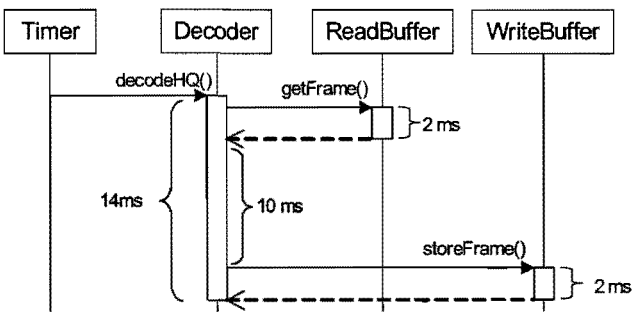


Figure 2. Sequence chart diagram for a decoding task

Thus, with the predictable assembly technique, we can derive the *execution (response) time* of a complete task in an application.

Unfortunately, the ROBOCOP predictable assembly model does not consider two important properties: using a multitude of tasks in an application and means for synchronization between tasks. Especially the last property is extremely important for real-time performance analysis.

III. An Improved Model For Prediction

In order to find a processor utilization rate of a task, the execution time can be divided by a period or minimum inter-arrival time of a task. The cumulative processor utilization rate of a set of application tasks represents the application *processor utilization bound.*

The described scenario-based approach in Section II is a cornerstone for the predictable assembly technique. To apply the technique in practice, we introduce an *assembly description language.* The language allows an application developer to specify all behavioral and aspects of an application that may influence resource usage. We have found that a significant part of those aspects deal with synchronization of tasks. The final result of the specification is an application task model.

The designer, while writing the specification, takes as an input (see Figure 3):

a) Available component description (incl. WCET per function and provides/requires interfaces),
b) Commonly used scenarios.

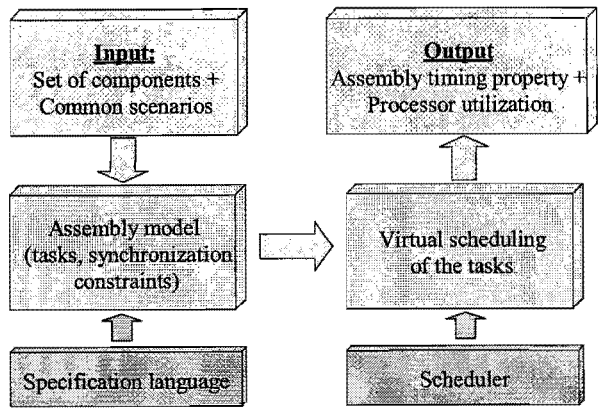


Figure 3. Work flow of the predictable assembly technique

In the assembly specification model, the designer specifies component dependencies, describes tasks and corresponding function sequence calls, and finds synchronization aspects between the tasks. When the specification is available, it actually represents application tasks properties. Having the tasks properties, it is possible to schedule these tasks. Therefore, the final step of the predictable assembly technique is to apply virtual scheduling on the application model. As a result, the following application timing properties are found:

- Response time of critical tasks,
- Processor utilization bound,
- Schedulability of the application on a target.

IV. Conclusions

The improved predictable assembly technique enables the prediction of application timing properties already at the design stage. The technique can be extended for predicting memory and bus usage.

For future work, we propose to conduct several case studies with multimedia applications and investigate a prediction-error rate of the technique for different application domains.

References

[1] ROBOCOP public home page [http://www.extra.research.philips.com/euprojects/robocop/]
[2] Space4U public home page [http://www.extra.research.philips.com/euprojects/space4u/]
[3] Merijn de Jonge, Johan Muskens and Michel Chaudron. *Scenario-Based Prediction of Run-time Resource Consumption in Component-Based Software Systems.* In Proceedings of 6th CBSE conference. May 3-4, 2003

Modeling Predictable Multiprocessor Performance for Video Decoding

Peter Poplavko[1,3] and Milan Pastrnak[1,2] *
[1] Eindhoven University of Technology
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
[2] LogicaCMG Eindhoven / [3] Philips Research Labs Eindhoven
{P.Poplavko, M.Pastrnak}@tue.nl

Abstract — **This work addresses implementation of decoding an MPEG4 bitstream with multiple arbitrarily shaped video objects (AS-VOs). Such multimedia applications pose challenging requirements on embedded systems design with respect to compositionality, scalability, and predictability in order to meet real-time constraints. *Multiprocessors* based on networks-on-chip (MP-NoC) are appropriate platforms that satisfy these requirements [3]. The *workload* of the AS-VO-decoding changes dynamically at run-time. To control the system resources, it is favorable to have *workload estimation* of one video frame (or VOP), before the frame (or VOP) is decoded. To make this task easier, the encoder puts a few *complexity parameters* in the VOP header. For single-processor implementation, linear *complexity functions* on can be used to obtain the workload [6, 2]. Preliminary results show that with the full a-priori knowledge of the input bitstream, the model is accurate within 6% in average [6]. For multiprocessor implementation, we extend these models to *parametrical IPC graphs* [7]. In this case, the same accuracy is possible, but hardly feasible at run-time due to coding efficiency reasons. In [7], a feasible approach has been proposed, which yields a safe upper bound on the workload, but on the price of high error. We discuss the current status in our modeling approach.**

Keywords — **network-on-chip; system-on-chip; timing model; performance evaluation; resource estimation; real-time; data-flow graph**

I. INTRODUCTION

In order to run real-time multimedia applications on an on-chip multiprocessor system, techniques have to be developed to control the resource usage by those applications on the multiprocessor platform.

To tackle the interactivity and dynamism of applications, a real-time environment is required to coordinate multiple *jobs* on the set of processors. Informally, a job is an activity started and stopped by some unpredictable run-time events, which can come from user actions (e.g., pushing a button) or from changes in the video scene. We currently assume that that each job is assigned to decode one video object. We also assume that each running job is invoked regularly. At each invocation, it has to produce a certain number of output data samples (macroblocks or MBs) that constitute together one frame (video object plane, or VOP). A *real-time control hierarchy* must ensure that the complete application (the set of active jobs) provides output with the best quality that can be achieved while meeting the timing constraints [9].

Complex multimedia applications pose challenging requirements on embedded systems design with respect to *compositionality* and *scalability*. Moreover, the authors believe that, in order to meet the real-time constraints at low cost, *predictable hardware behavior* is necessary. We study a design of multiprocessor network-on-chip (MP-NoC), which intrinsically supports these requirements [1, 3].

The *multiprocessor workload* of each job and its communication bandwidth requirements may change over time, e.g., when the dimensions, shape and texture of the correspondent video object changes. The real-time control hierarchy should track these changes in a timely manner and adapt to them at run-time, e.g., by allocating more processors to the job or by reducing the amount of computations at the expense of visual quality [9].

In this paper, we present an overview on the use of *parametrical inter-process communication (IPC) graphs* [7, 8] as models allowing estimation of the workload of the multiprocessor jobs. We develop IPC graph models for the decoding of arbitrarily shaped objects, as defined in the MPEG4 standard.

This paper is organized as follows. Section II gives background information about the MP-NoCs. In Section III, we briefly present current IPC graph model together with the results on accuracy. Section IV concludes this paper and mentions future work.

*Supported by the European Union via the Marie Curie Fellowship program under the project number HPMI-CT-2001-00150.

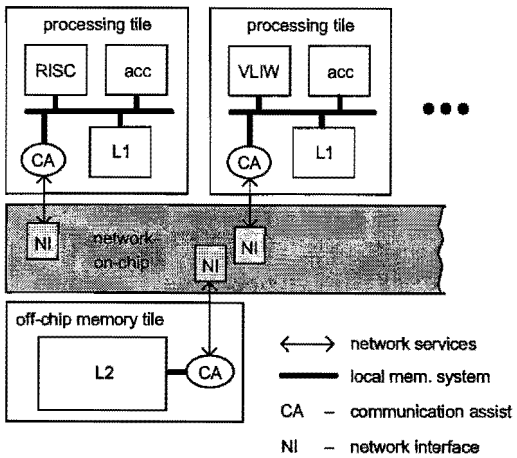


Figure 1: Architecture with tile-based MP-NoC.

II. MULTIPROCESSOR NETWORK-ON-CHIP

On chip multi-processor architectures are important for the implementation of MPEG-4 (multimedia) applications [2].

The growing design complexity results in the need in the platforms featuring *compositionality, design reuse* and *design scalability*. The MPEG-4 standard supports these requirements from the application side. The network-on-chip design paradigm supports them from the hardware side [3].

Meeting timing constraints requires predictable timing in communication. On-chip networks can support predictable timing by offering a *guaranteed throughput* service [3].

To manage complexity and support predictability, we use a hierarchical approach in the system architecture design. Currently, we assume the simplest variant of this approach, in the form of a so-called *tile-based* architecture as depicted in Figure 1.

This type of architecture assumes two levels of hierarchy. The lower level shows the details of processing tiles, and the higher level consists of a set of tiles connected by the on-chip interconnection network and a level-2 (off-chip) memory. A *processing tile* represents a small self-contained embedded computer, consisting of one or two embedded CPU cores (e.g., RISC), local level-1 memory (denoted as L1 in Figure 1) and application-specific accelerators. A *communication assist* serves in each tile as a gateway from the local, tile-specific memory system to the standard network services.

The tile-based hierarchical approach can be seen as a natural extension of the existing video decoders that handle with 1 or 2 processors only one principal MPEG-4 video object containing the complete picture of the video stream [2]. To implement, e.g., the MPEG-4 Core profile, multiple smaller video objects can be handled by running multiple instances of a single-object-decoding job on the replicated processing tiles.

III. WORKLOAD ESTIMATION

In the design of a real-time control hierarchy it would be favorable to have a method to estimate in advance how many processor cycles each job consumes from its processors (*job workload*).

We propose to construct at design time a parametrical timing model and then use it at run-time for the workload estimation. We require the encoder to put the complexity parameters into the image headers.

A. Design-Time Model Construction

To express the parallelism and the communication of multiple processes executing the main loop of the job, we use an *inter-process communication graph,* which is an instance of a *homogeneous dataflow graph* model of computation [5]. As an example, we show in Figure 2 an IPC graph for intra-frame shape-texture decoding. The graph is constructed in two major steps, presented below.

Step 1. Resource Estimation [2, 4, 6]

For each main *computation actor* (software routine like CAD, CBP, VLD...) the *complexity function* is defined for the given data granularity (e.g., single *macroblock* (MB)). We currently assume that all processors are ARM7TDMI cores with flat local memory with single-cycle access.

For example, the *complexity function* of inverse quantization actor is estimated by:

$$t_{IQ} = 1.39\text{K}\cdot\varphi + 44\cdot N_{AC} \qquad (1)$$

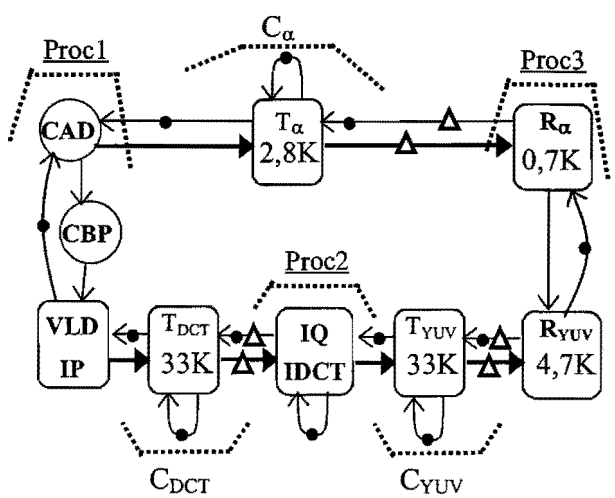


Figure 2: IPC graph of MPEG-4 shape-texture decoder of an intra-frame of a video object (or I-VOP) [7]

where φ and N_{AC} are *complexity parameters*. In particular, φ is the total number of non-transparent sub-blocks and N_{AC} is the total number of non-zero AC coefficients. We have observed that for the ARM7 processor, the complexity functions are accurate within 6% in average, given that all parameters are known exactly [6].

Note, that if all actors are mapped to a single processor, the total workload is computed by simply adding the complexity functions together. We obtain again the accuracy in the order of 6%. For multiprocessor job, it is not so straightforward to compute the workload.

Step 2. IPC Graph Construction [7, 8]

This step is performed after the actors have been assigned to processors, and inter-processor data communication has been assigned to network connections. For each processor Proc_*i*, we introduce a process cycle into the IPC graph. For each connection C_j, we introduce a subgraph that models the connection. In Figure 2, we see an IPC graph containing three process cycles (Proc1, 2 and 3) and three connection subgraphs (C_α, C_{DCT} and C_{YUV}). See [7] for more details.

B. Run-Time Workload Estimation

In [7] we propose a way to use IPC graph to obtain workload estimation for the decoding of one image of a video object, called *video object plane* (VOP). One macroblock (MB) passes all stages of processing in every iteration of IPC graph. If complexity parameters of all MBs in VOP are known, the delays of all actors at each iteration are also known. In this case, a longest-path computation in IPC graph unfolded multiple times would yield the job workload with the same accuracy as the complexity functions (around 6%). However, it is not feasible to enforce the encoder to put all complexity values for all MBs into the header.

To tackle this problem, we propose in [7] a *scenario approach*. All MBs of the VOP are divided into several scenarios. For the shape-texture decoding we have identified three scenarios: "transparent MB", "boundary MB" and "opaque MB"). For each scenario, we require the encoder to put in the header one set of complexity parameters which *characterizes* all MBs that belong to that scenario (*characterization set*). In addition, we need two extra parameters, namely, J_s, or the total number of MBs in scenario s and L_s, or the number of transitions to scenario s from other scenarios.

We estimate the job workload on all processors at run-time as follows [7]:

$$workload = \sum_s \lambda_s \cdot J_s + \sigma_s \cdot L_s, \qquad (2)$$

λ_s and σ_s (throughput and lateness) are certain properties of the IPC graph in scenario s, which can be computed by applying fast graph analysis algorithms.

The desirable properties of workload estimation are *safety* and *tightness*. The *safety* means that there is enough confidence that the real workload shall not exceed the estimated value. The *tightness* requirement means that the estimated value is not too pessimistic.

For the safety reason, we proposed in [7] to characterize each scenario with the maximum values of each parameter over all MBs belonging to the scenario. However, for the safety, we had to pay with the tightness. In [7], we have evaluated (2) for an I-VOP of a test bitstream and have observed 55% overestimation of the real workload.

In future work, to improve this result, we will try to exploit the 'smoothing' effect that large FIFO buffers and larger data granularities have on the workload variations and to use a characterization set which is in between the maximum and average set of parameters.

IV. Conclusions and Future Work

This paper is an overview of our work on modeling the performance of dynamic video-decoding applications. We assumed multiprocessor networks on chip as a target platform, because they meet several important requirements of multimedia application design. The proposed models capture both computation and communication within a dynamic video-decoding job. The models can be used for the run-time workload tracking in the real-time control hierarchy of the platform. Currently we can obtain a safe, but not tight (55% or more) overestimation of the workload.

In the future work, we will develop more elaborate models and estimation methods for the chosen application driver (shape-texture decoder). We will also investigate the possibilities for a real-time control hierarchy, like the one proposed in [9].

Acknowledgement

The authors are grateful to Marco Bekooij, Philips Research Labs Eindhoven, for some valuable feedback concerning the modeling approach.

References

[1] M. Bekooij, O. Moreira, P. Poplavko, B. Mesman, J. van Meerbergen, M. Duranton, and L. Steffens, Predictable Embedded Multiprocessor System Design, *Proceedings Philips Conference on DSP*, 2003.

[2] M. Berekovic, H.-J. Stolberg, and P. Pirsch, Multicore System-On-Chip Architecture for MPEG-4 Streaming Video, *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 688-699, Vol.12, No. 8, August 2002.

[3] K. Goossens *et al.*, Guaranteeing the quality of services in networks on chip, *Networks on Chip,* edited by A. Jantsch and H. Tenhunen, Kluwer Academic Publishers, pp. 61-82, 2003.

[4] R. Lauwereins, M. Engels, M. Ade, J.A. Peperstraete, Grape-II: A System-Level Prototyping Environment for DSP Applications, *IEEE Transaction on Computer*, pp. 35-43, Vol. 28, No. 2, February 1995.

[5] E.A. Lee and D.G. Messerschmitt, Static Scheduling of Synchronous Data Flow Programs for Digital Signal Processing, *IEEE Transactions on Computers*, pp. 24-35, Vol. 36, No.1, 1987.

[6] M. Pastrnak, P. Poplavko, P. H. N. de With, and J. van Meerbergen, On Resource Estimation of MPEG-4 Video Decoding for A Multiprocessor Architecture, *Proceedings of PROGRESS 2003*

[7] P. Poplavko, M. Pastrnak, J. van Meerbergen, and P. H. N. de With, Mapping of an MPEG-4 Shape-Texture Decoder onto an On-chip Multiprocessor, *Proceedings of PRORISC 2003*, http://www.ics.ele.tue.nl/~epicurus

[8] P. Poplavko, T. Basten, M. Bekooij, J. van Meerbergen, and B. Mesman, Task-level Timing Models for Guaranteed Performance in Multiprocessor Networks-on-Chip, *ACM Proceedings CASES'03*, pp.63-72, Oct. 30-Nov.1, 2003.

[9] C. M. Otero Pérez, L. Steffens, P. van der Stok, S. van Loo, A. Alonso, J. F. Ruiz, R. J. Bril, and M. G. Valls, QoS-Based Resource Management for Ambient Intelligence, in *Ambient Intelligence: Impact on Embedded-System Design*, ed. By T. Basten, M. Geilen, H. de Groot, Kluwer Academic Publishers, 2003.

THE DESIG
MUL
ARCH