DSPs as flexible Multimedia Accelerators

Robert Baumgartl, Hermann Härtig

Operating Systems Group

Department of Computer Science

Dresden University of Technology

robert.baumgartl@inf.tu-dresden.de

hermann.haertig@inf.tu-dresden.de

Outline

- Introduction and Motivation
- Experimental Environment
- System´s Operation by Example
- The SPROS DSP Kernel
- The Host Server
- Performance Evaluation
- Conclusions
- Outlook

Introduction: DSPs and Multimedia

DSPs are:

- very powerful
- cheap
- designed to handle large-sized streamed data
- easy to operate under real-time conditions
- freely programmable
- integratable into computer systems

increasing complexity of PC-based Multimedia Applications

➡ **Use DSPs as flexible Multimedia Accelerators for PCs ?**

Project Outline

necessary steps:

- identification of ideal mechanism for inter-processor communication
- systematic construction of a coprocessor-optimized DSP kernel
- design and implementation of a management component running at the host
- integration into Dresden Realtime Operating System (DROPS)

goals:

1. substantial performance improvement to solutions based on present techniques
2. knowledge acquisition for the integration of future DSP architectures, generalization

Experimental Environment

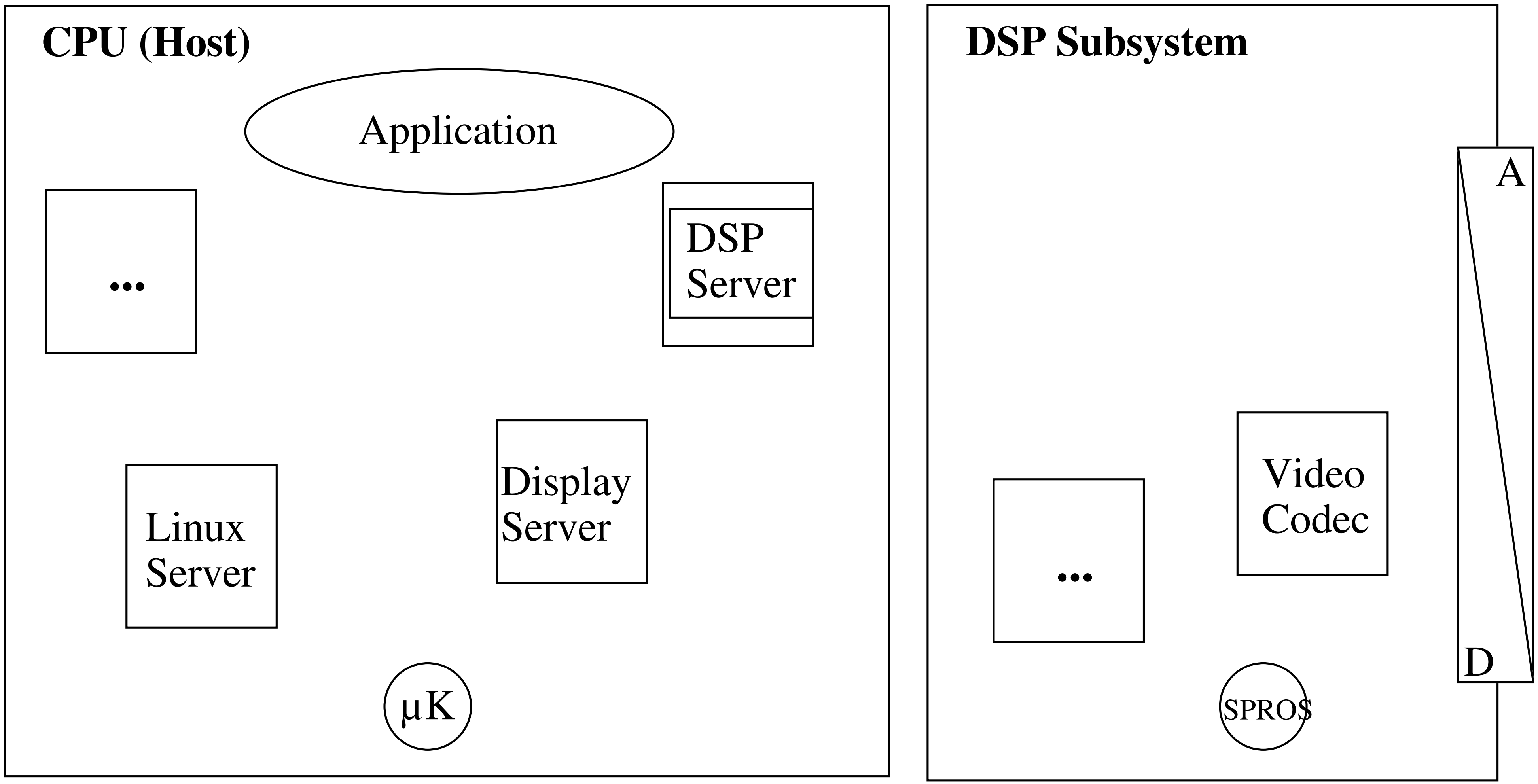
Host:

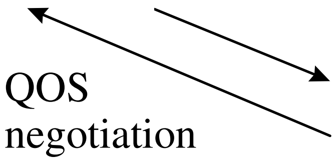
- Pentium MMX, 166 MHz
- 64 MByte RAM
- 512 kBytes 2nd level cache
- Linux, L4

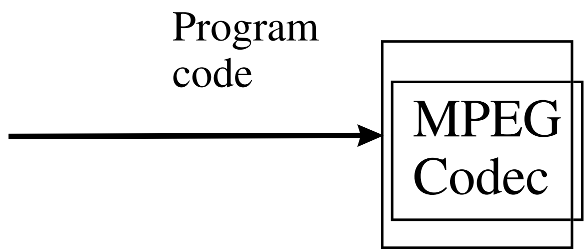
DSP Subsystem:

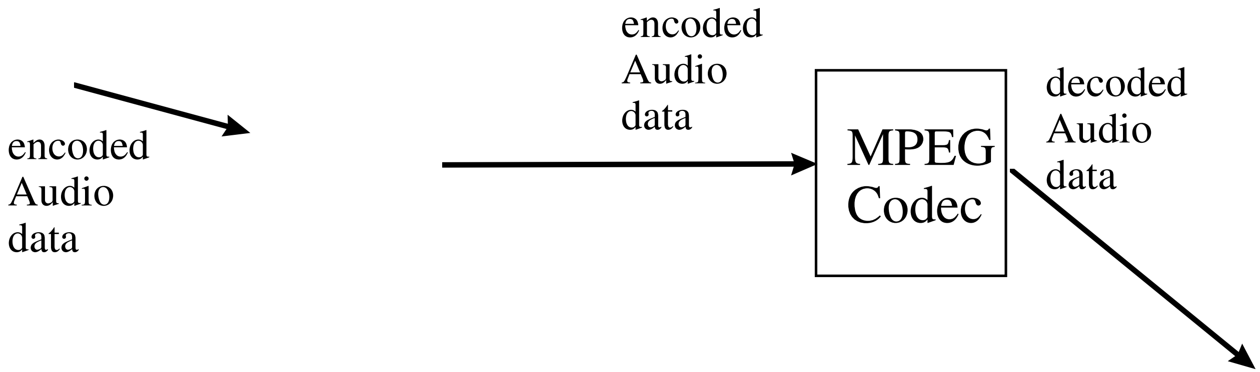
- TMS320C44, 50 MHz
- 128 kBytes local and global SRAM
- AMCC S5933 PCI controller
- modular board, TIM size

Example: An MPEG 1 Audio Decoder

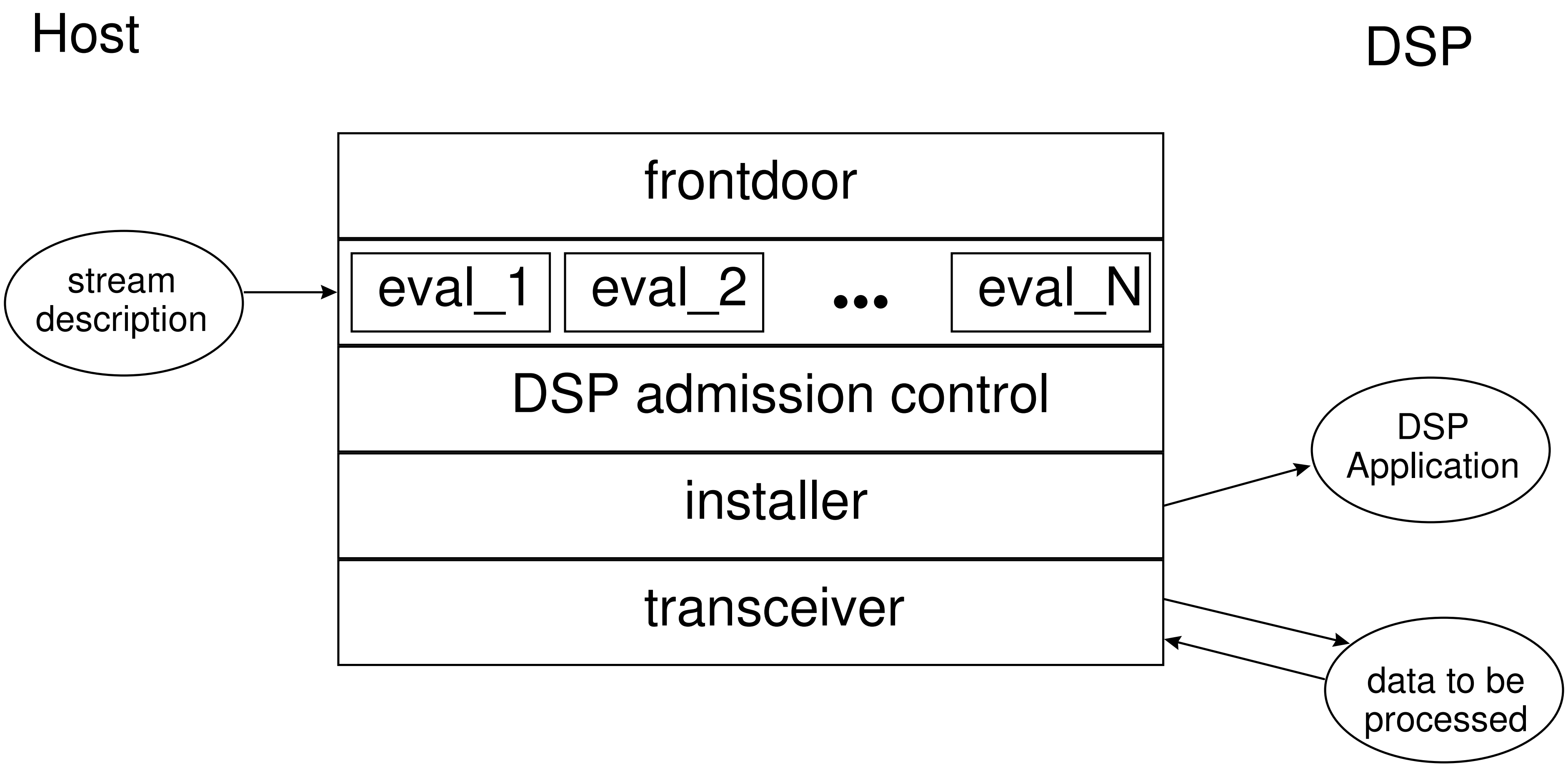



QOS
negotiation


Program
code
MPEG
Codec


encoded
Audio
data
encoded
Audio
data
MPEG
Codec
decoded
Audio
data

Host Server Architecture



The SPROS DSP Kernel

Limitation to:

- periodic real-time tasks
- tasks with constant execution time (constant deadline)

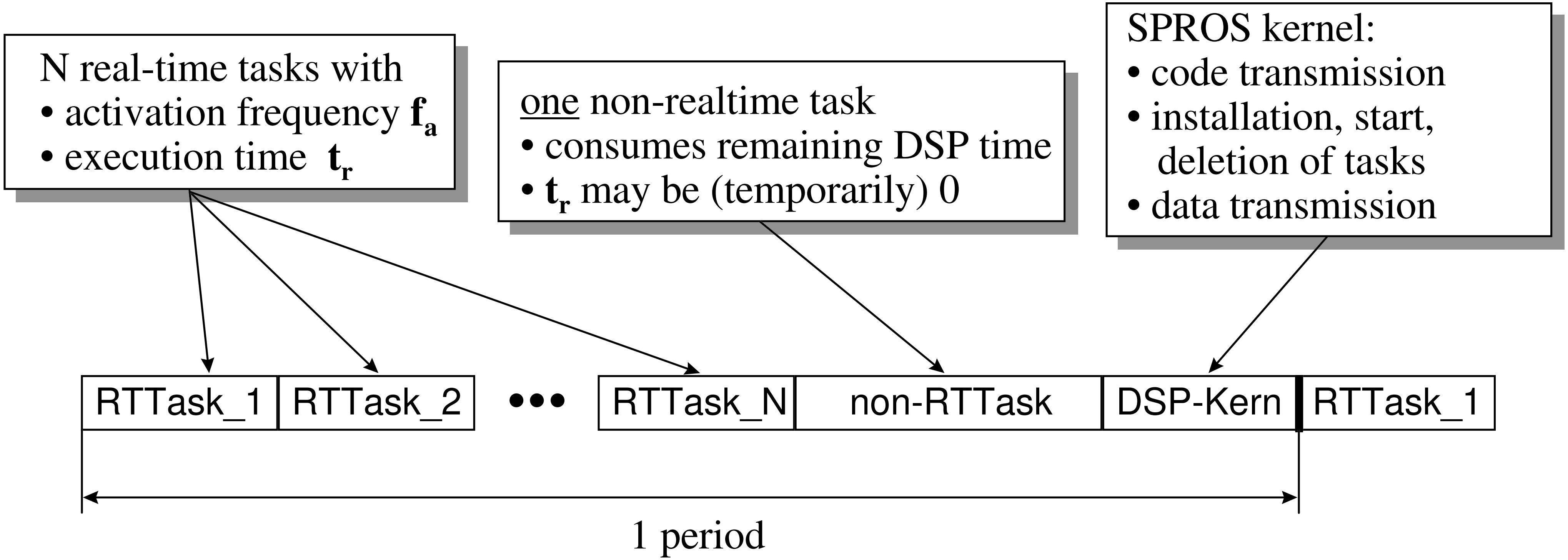
⇨ tasks described by 2 parameters:

1. activation frequency f_a
2. execution time per activation t_r

minimum necessary functions:

- installation, start and deletion of tasks,
- context switch,
- (static) scheduling,
- communication to host incl. synchronization mechanisms,
- management of communication buffers,
- management of peripheral devices (e.g., audio codecs).

SPROS Scheduling

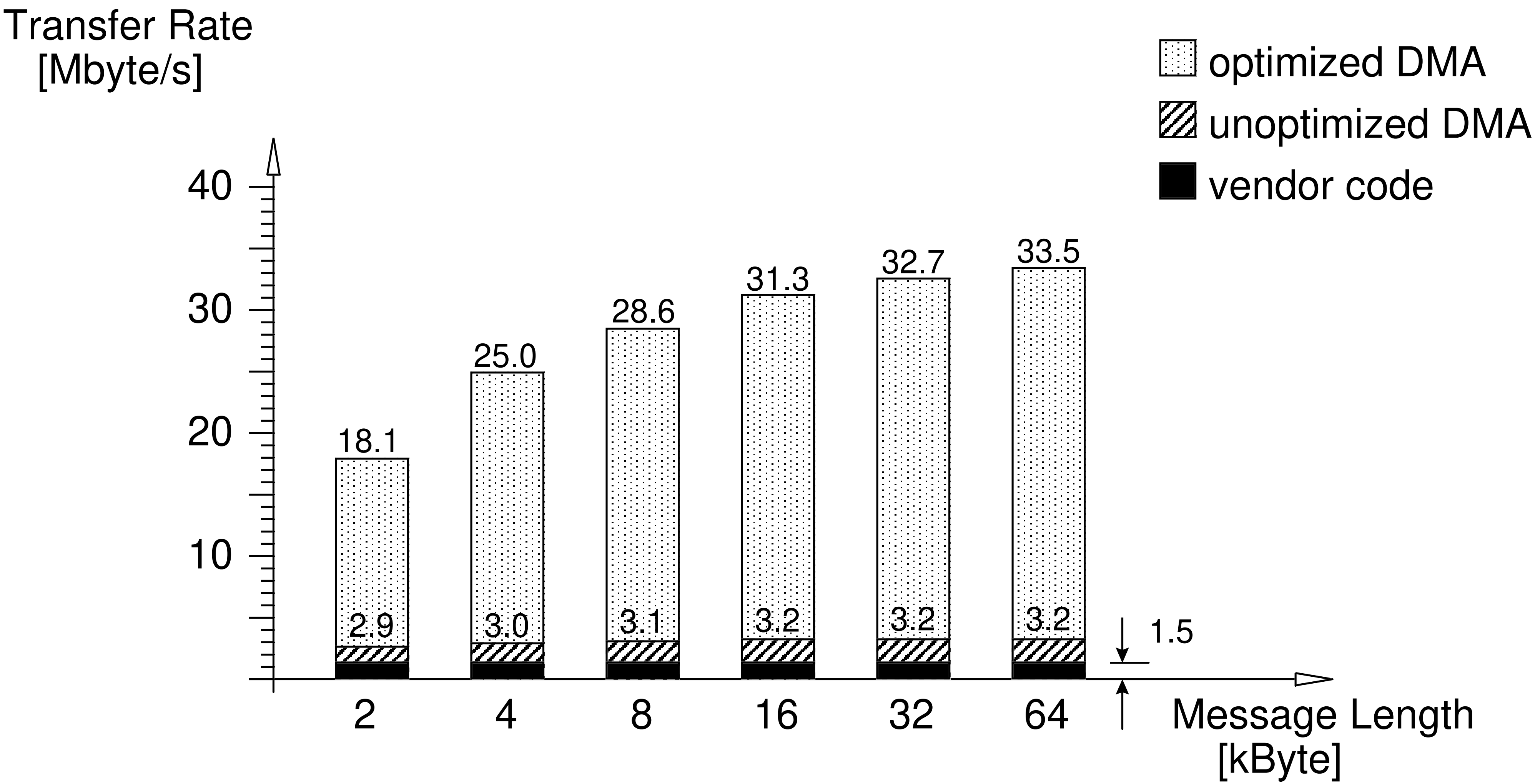


- schedule computation by host
- timer-activated context switch

Summary: SPROS Features

- preemptive, static scheduling
- supports dynamic task installation
- very exactly predictable timing
- can be used by any DSP application
- not portable
- minimum set of abstractions
- fully implemented in assembler language
- PCI busmaster DMA for data transmission

Communication Performance



Preliminary SPROS Results

context switch time:	4.4 µs
	80 + 30 = 110 cycles
memory consumption:	approx. 4000 words
interrupt latency:	3 cycles

Conclusions

Preconditions:

- fast hardware communication mechanism
- host OS must preserve communication performance
- powerful DSP chip with peripheral devices
- very efficient kernel (assembler implementation) with minimum set of abstractions
- wide DSP software base

➡ **DSPs can be exploited as flexible Multimedia accelerators!**

Outlook

next project stages:

- completion of the host server
- SPROS kernel fine tuning

outlook:

- (partly) automatic generation of DSP-specific kernel and host server
- integration of flexible hardware (FPGA)
- transition to new DSP architecture

further information:

`http://os.inf.tu-dresden.de/~baumgrtl/work.html`

MMX vs. DSPs

Image Processing Filter:

$$y = \sum_{x=1}^{N} n(x) \cdot m(x) + \sum_{x=1}^{N} n(x) + \sum_{x=1}^{N} m(x)$$

Processor	f_{clock}	Processing Time (1 vector element)
Pentium MMX	166 MHz	21 ns
DSP TI C44	50 MHz	120 ns
Pentium II	300 MHz	11.6 ns
DSP C6x	200 MHz	10 ns

*D*resden *R*ealtime *Op*erating *S*ystem (DROPS)