Outline

Workshop Outline

The consumer media device
  • The big picture
Developing video software
  • Software subsystems
  • Data types
  • Optimization techniques
  • Testing
Trends and conclusions
Big Picture

Consumer media devices are complex systems; several software and hardware subsystems:
- Software: Control, video, I/O, RTOS
- Hardware: GPP/DSP, coprocessor(s), DMA, I/O, memory

Motivation

- Video moving from hardware to software
  - Processors now have the processing power
  - Moving to software brings many benefits, but...
- Very demanding workloads...
  - Complex algorithms, changing rapidly
  - High computational requirements
  - Stringent real-time constraints
    ...stress the processor’s abilities
- Creates challenges
  - Optimization
  - Testing
Workshop Outline

The consumer media device
- The big picture

Developing video software
- Software subsystems
- Data types
- Optimization techniques
- Testing

Trends and conclusions

Software Subsystems

Primary software subsystems include:

- Overall Control: Control, GUI (play, stop, rewind) ...
- Processing: codecs, deinterlacing ...
- Post-processing: color conversion, deblocking ...
- I/O: camera, LCD
- Network protocol: TCP/IP, RTSP
- Real-Time Operating System: Linux, VxWorks ...
Key Software Considerations

Overall control
- Port to OS and hardware platform

Video processing and post-processing
- Starting point?
- Video data representation
- Optimize for speed, memory use, power, etc.

RTOS
- Add/remove features and device drivers

Software integration
- Control + video processing + I/O + RTOS

Testing
- Audio/video quality (test vectors)
- Real-time performance

Software Development

Common division of labor
- Separate teams for groups of related subsystems
- Teams work together to integrate and test

INTEGRATION: Control + Processing + RTOS
- Real-Time Performance Testing

TEAM 1: Control
- Port to RTOS
- Helper Functions
- UI Testing

TEAM 2: Processing
- Algorithm Implementation
- Optimization
- Output Quality Testing

TEAM 3: RTOS
- Port to Platform
- Device Drivers
- I/O Testing

Hot spot
- Video processing can pose significant development challenges
Developing Embedded Video Software

Video Software: What’s Special?

Not like other kinds of software development:

- Extreme computational demands
- Algorithm attributes
- Data access attributes
- Memory bandwidth requirements
- Testing and validation requirements

- Resource constraints
- Standards
- Real-time requirements
- Reliability
- Specialized and complex processor architectures
- Opportunity for lots of parallelism

Optimization is essential!

Workshop Outline

The consumer media device
- The big picture

Developing video software
- Software subsystems
- Data types
- Optimization techniques
- Testing

Trends and conclusions

© 2004 Berkeley Design Technology, Inc.
Data Types

Many important and interesting topics, e.g.,

- **Pixel representation**
- **Image representation**
- Frame stream representation
- **Approximations**
- Error propagation/analysis
- Saturation
- Signal scaling
- Rounding modes

*(Focus topics)*

Pixel Representation

Various ways of encoding a pixel

- RGB (cameras, monitors, and scanners)
- YCbCr, YUV (video codecs, television transmission)
  
  Separating luminance from chrominance eases compression (chrominance can be down-sampled)

Need for color conversion

- Capture and display video equipment: RGB...
- ...while codecs use YUV
- Color conversion can consume significant processing power
  - 30% to 60% of the cycles needed by the video decoder
Image Representation

Various ways of encoding color fields

Planarized 4:2:2

Planarized 4:1:1

Interleaved 4:2:2

Interleaved 4:1:1

Various ways of encoding frames

• Progressive frames (monitors, digital TV)

• Interleaved fields (analog TV, most cameras)
Image Representation

Choose between planarized and interleaved data representation based on:

- Algorithm(s)
  - Are Y, U, and V data processed independently?
- Architecture
  - What is best the best data arrangement for maximizing parallel execution?
- I/O
  - What options are available for acquisition and rendering of video data?
  - What do standards require?

Approximations

Dropping video frames
- Can be done occasionally
- More forgiving than audio

Compromising on precision
- Sometimes, processor load can be dramatically lowered by dropping one bit
  - E.g., staying with 8-bit data instead of 16-bit data
Workshop Outline

The consumer media device
  • The big picture
Developing video software
  • Software subsystems
  • Data types
  • Optimization techniques
  • Testing
Trends and conclusions

Optimization Techniques

Optimization process
  → Profile → Analyze → Optimize

Optimization levels
  • Algorithm level
    • Either processor dependent or processor independent
  • High level language (HLL) level
    • Relies heavily on the compiler
  • Hand-coded assembly language level
    • Yields the best performance
Optimization Techniques

Optimization targets
- Execution speed
  - Using more parallelism
  - Reducing memory accesses
    - Avoid cache, or L1, “thrashing”
- Memory usage
  - May conflict with optimizations for speed
- Energy consumption
  - Minimize off-chip memory accesses

Profiling: Find S-rate Operations

Functions can be classified based on invocation rate:

I-rate (initialization)
< 1 time/sec

K-rate (control)
~10-1,000 times/sec

S-rate (samples)
~10⁴ - 10⁷+ times/sec

~80% of code
~20% of time

Optimization most useful

~20% of code
~80% of time
High-Level Language Optimizations

- Some processors provide instructions specialized for video. But will the compiler use them?
  - Handling saturation efficiently
    - Saturation instruction intrinsics, look-up tables
  - Handling 8-bit arithmetic efficiently
    - Interpolation, parallel operation intrinsic instructions
- Some reference code uses char data types. But what will compiler do?
  - Handling 8-bit data moves efficiently
    - Packing 8-bit data into 16- or 32-bit words

Memory Access Optimization

Video Processing
- Video frame much bigger than typical L1 memory (cache or SRAM)
  - L1: typically 10 to 100 KB
  - Frame: typically 100 KB to 1 MB+ for each frame
Memory Access Optimization

Default processing sequence
- Operation 1 on entire frame
- Operation 2 on entire frame...

Observation: Most video algorithms operate on independent blocks of data (8x8, 4x4, lines ...)

Optimization: process one block at a time through multiple algorithm steps
- Subset stays resident in L1, cutting external memory accesses

External memory accesses
- Cache misses,
- DMA overhead

Input frame \( \rightarrow \) Temp block \( \rightarrow \) Output frame

© 2004 Berkeley Design Technology, Inc.
Memory Access Optimization

Process second block

- Input frame
- Temp block
- Output frame

Memory Access Optimization

Process last block
External memory accesses have been minimized

- Input frame
- Temp block
- Output frame
Workshop Outline

The consumer media device
  • The big picture
Developing video software
  • Software subsystems
  • Data types
  • Optimization techniques
  • Testing
Trends and conclusions

Developing Video Software: Testing

Testing

Hardware/development platform
  • Challenges with data set sizes
Video processing software
  • Operating modes
  • Data dependencies
  • Output quality
System level (hardware + software)
  • Real-time requirements
  • Worst-case conditions

© 2004 Berkeley Design Technology, Inc.
Hardware/Development Platform

Video codecs consume and produce vast amounts of data:
- Compressed bit streams up to ~10+ Mbps
- Uncompressed streams up to 1+ Gbps

Processor simulation model usually far too slow
- Real hardware is needed

Development board must have means to
- Supply large test vectors
- Capture potentially even larger output
  - In digital form for verification

Codec Software: Operating Modes

Video codecs typically have several operating modes
- MPEG-4 video:
  - 21 profiles (18 in part 2 and 3 in part 10)
  - 5 kbps – 1 Gbps bit rates
  - Sub-QCIF to studio resolution
  - Various mixes of I, P, and B frames
  - Progressive and interlaced video

All valid combinations must be thoroughly tested
- Standard reference test vectors probably not sufficient to identify all potential bugs
Codec Software: Output Quality

Difficult to measure quality in context of “lossy” compression algorithms
• Sum of absolute differences (SAD) still most common approximation of codec quality
• Intentionally not bit-exact
• Post-processing algorithms may improve visual quality but deteriorate SNR
  • Deblocking
  • Deringing
  • Sharpening

⇒ Visual inspection of output quality is key

Quality measured:
• Using reference codec and test vectors
• Tests must stress algorithm features...
  • E.g., different motion estimation modes
• ... but also implementation features
  • Fixed-point features (saturation, accumulation, etc.)
  • Various implementation flavors for different sets of parameters (filter size, frame size, etc.)
**System Level: Real-Time**

Real-time performance is not optional
Processor is often underpowered
- Careful codec optimizations can pull underachievers up to real-time performance

![Diagram showing processing in real-time](image)

**System Level: Worst-Case Loading**

Most demanding operating mode
- Highest frame rate and frame size
- Interrupts enabled and active (UI and I/O)

Most demanding data
- Video codecs have data dependent execution paths
  - E.g., pixel, ½-pixel, or ¼-pixel motion compensation
  - E.g., ratios of I, B, P frames
- Algorithms have data-dependent memory accesses
  - E.g., motion-compensated deinterlacing using either data from current frame (cache) or previous frame (ext. memory)

Worst-case performance difficult to identify in video

© 2004 Berkeley Design Technology, Inc.
Workshop Outline

The consumer media device
  • The big picture

Developing video software
  • Software subsystems
  • Data types
  • Optimization techniques
  • Testing

Trends and conclusions

Conclusions

Successful video software development:

  • Demands knowledge of the application, algorithms, and processor and mastery of a wide range of skills and tools

  • Typically requires aggressive optimization in order to meet tough real-time deadlines

  • Requires a well-thought-out testing strategy!
Trends

Processors are getting faster and compilers are getting better but
- Newer video algorithms (e.g., AVC) more demanding
- Video processing remains the most demanding task
- Pre- and post-processing increases processor load

Optimized software libraries and hardware accelerators are more common
- Signal processing function level, e.g., IDCT, ME, etc.
- Application level, e.g., MPEG-4 Video Simple Profile

Trends & Conclusions

Heterogeneous processors
- Architectures
  - Processor core + programmable logic
  - Multi-processor SoCs
  - Coprocessors
  - Accelerators
- Workload
  - Proposal: off-load compute-intensive S-rate operations to custom logic or specialized processor
  - Reality: inter-processor communications and synchronization overhead can be deadly
Trends & Conclusions

Trends

Highly parallel architectures

• Architectures
  • MiMagic 6 APA (NeoMagic)
  • Adaptive Computing Machine (Quicksilver)
  • FPGAs (Altera, Xilinx, etc.)

• Workload
  • Proposal: Use many identical processing units to process independent blocks of data in parallel
  • Reality: Some video algorithms do not lend themselves well to parallel implementations (spatial and temporal dependencies)

For More Information...

www.BDTI.com

Free Information

• White papers/presentation slides on
  • DSP software optimization
  • Streaming media implementation
  • Processor architectures and performance
  • Digital audio compression
• Article reprints on DSP-oriented processors and applications
  • EE Times
  • IEEE Spectrum
  • IEEE Computer and others
• comp.dsp FAQ