TI | Developer Conference
March 7-9, 2007 • Dallas, TX

Minds in Motion

Measuring System Performance on the TMS320DM644x Platform

Catalog/EEE DSP Applications, Texas Instruments

Zhengting He, DSP SW App. Engr. z-he2@ti.com

Mukul Bhatnagar, DSP HW App. Engr. mbhatnagar@ti.com

Jackie Brenner, SMTS, j-brenner@ti.com

Benchmarks

- Provide a means to select a platform for application development and measure the effectiveness of the solution for that application.
- The three major things we want to measure are time, space and energy.
- As systems grow more complex with multiple processors and O/S requirements, the benchmarking process gets that much more complex.

Presentation Agenda

- Introduction
- DVEVM software description
- Necessary equipment and software
- Measuring the processor loading on the demo programs
- Demo memory usage
- Power measurements on the demos
- H.264 decode demo
- Conclusion

Introduction (1 of 2)

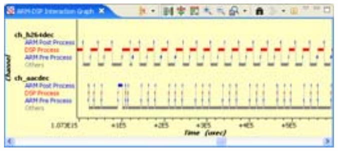
- The DaVinci platform offers a complete solution for many multimedia applications requiring advanced video codecs.
- The solution consists of a DM644x dual-core architecture with a rich mix of I/O peripherals and a complete software environment including inter-processor communication software and a codec framework that enables developers to use a variety of codec options.
- The DVEVM (Digital Video Evaluation Module) showcases the DM644x architecture and the associated digital video and audio system software solutions.

Introduction (2 of 2)

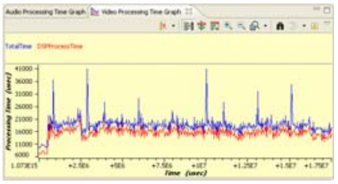
- This presentation provides a methodical approach to assess and measure both the performance and power of a DM6446 system running the video codec demos found in the DVEVM/Digital Video Software Development Kit.
- The performance/CPU loadings, memory usage and power consumption are measured for the H.264 audio/video demonstration software that is offered with the DVEVM.
  - H.264 algorithm demos were selected because they are relatively more complex and processing intensive compared to other video algorithms like MPEG4, MPEG2, etc.
- CPU loading measured using a new tool, the DM644x SoC (system-on-a-chip) Analyzer.

DM644x SoC Analyzer

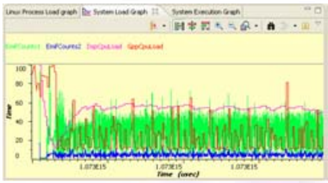
- DM644x SoC Analyzer is a real-time tool that provides system-level analysis and visualization of applications running on a DaVinci device.
- ARM⇔DSP interactions
  - Visualization of ARM interaction with the DSP for data processing
- Processing Time
  - Show time taken to process codec requests
- System Load
  - Overall CPU load and load required by individual tasks on ARM and DSP
- System Execution State
  - Running task on ARM and DSP



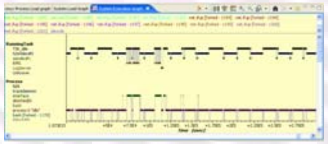
ARM⇔ DSP Interaction



Application Process Time



System Load



System Execution

Courtesy of Imtaz Ali

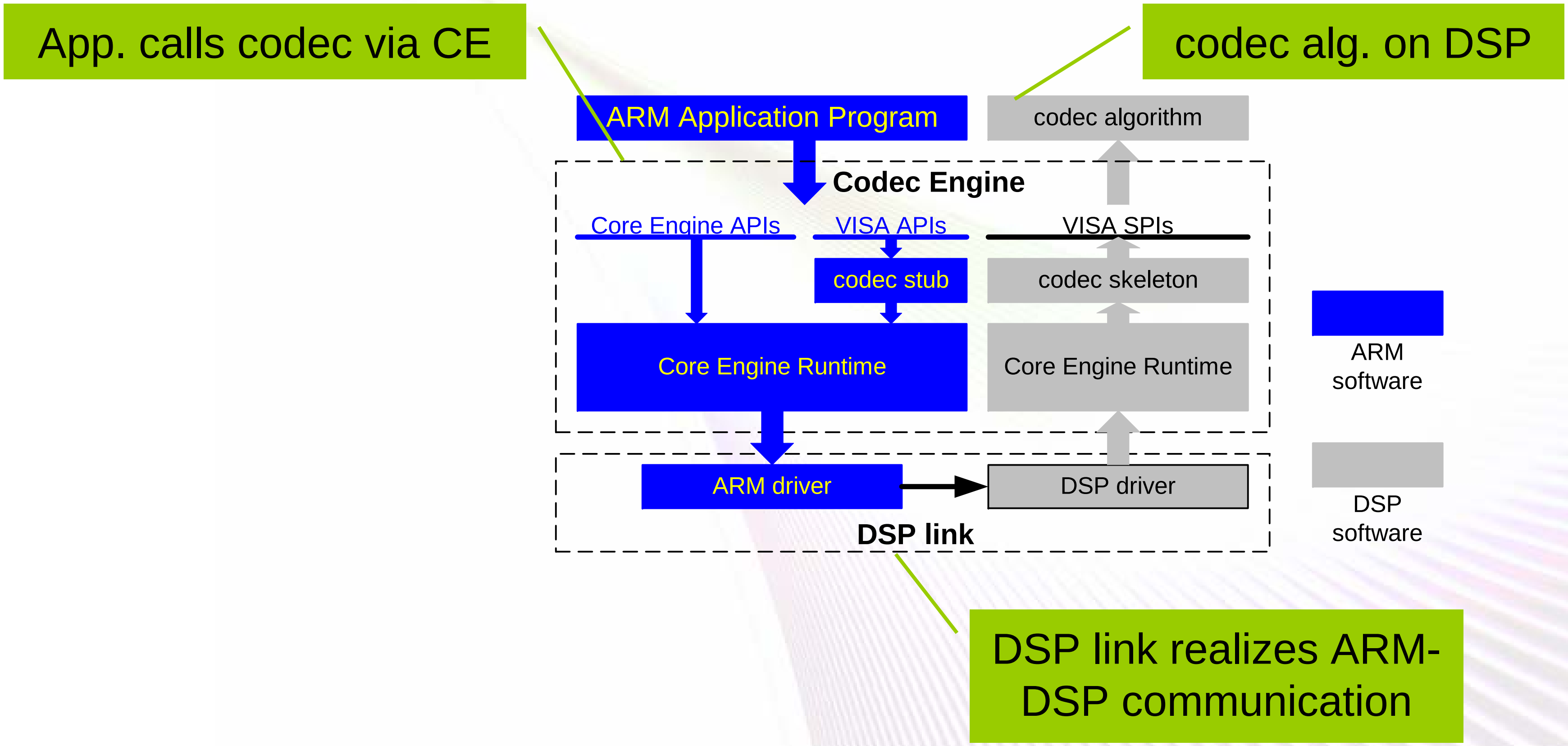
Reproducibility

The tools and methodology described here can also be utilized by a developer to evaluate and understand performance and power requirements in a system context for his or her own DM644x-based application software.

Presentation Agenda

- Introduction
- DVEVM software description
- Necessary equipment and software
- Measuring the processor loading on the demo programs
- Demo memory usage
- Power measurements on the demos
- H.264 decode demo
- Conclusion

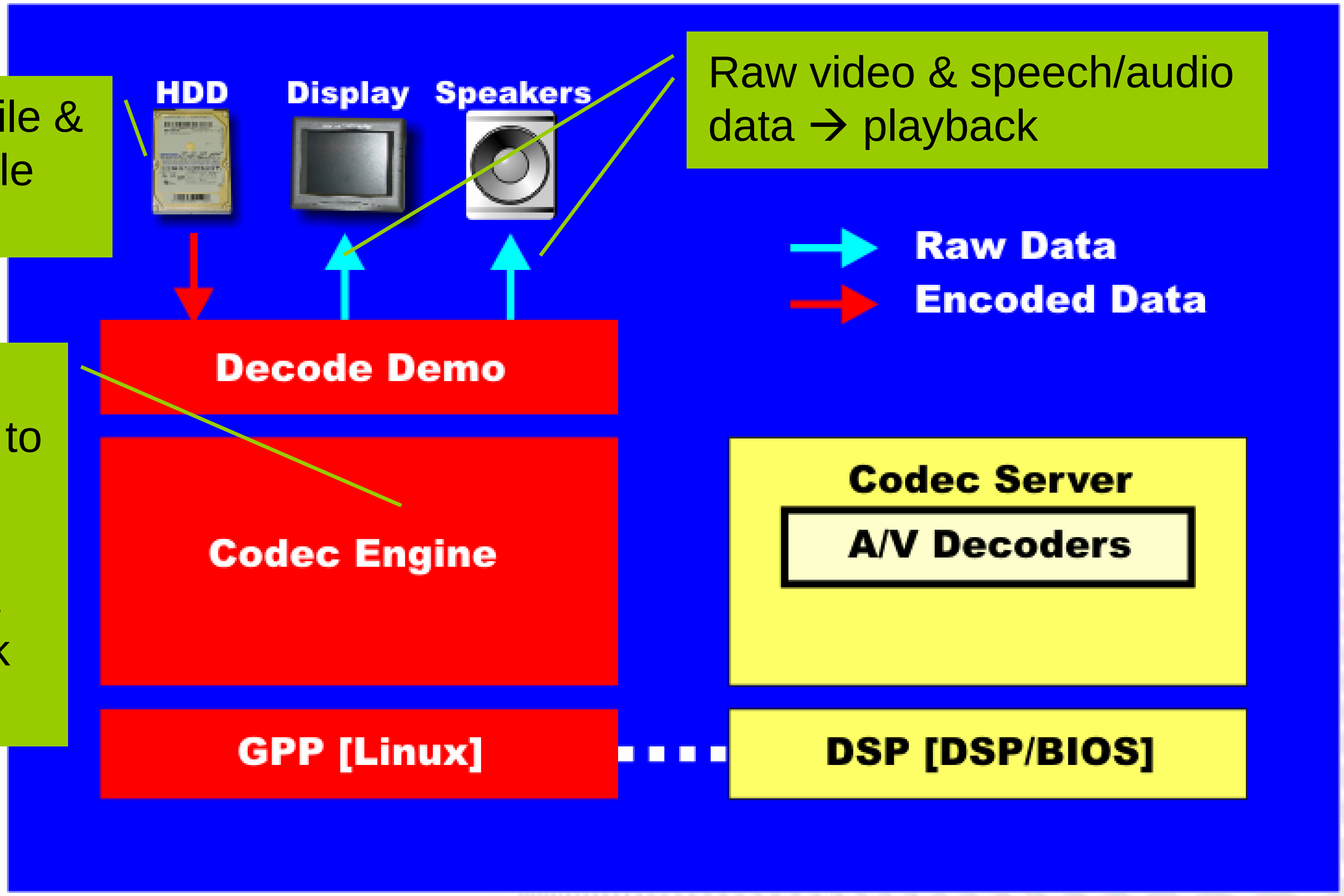
Software Architecture



Decode Demo

Decode video file & speech/audio file from HDD

Raw video & speech/audio data → playback



- ARM delivers encoded data to DSP frame by frame via CE
- Decoded data delivered back to ARM

Encode Demo

Encode video & speech channel

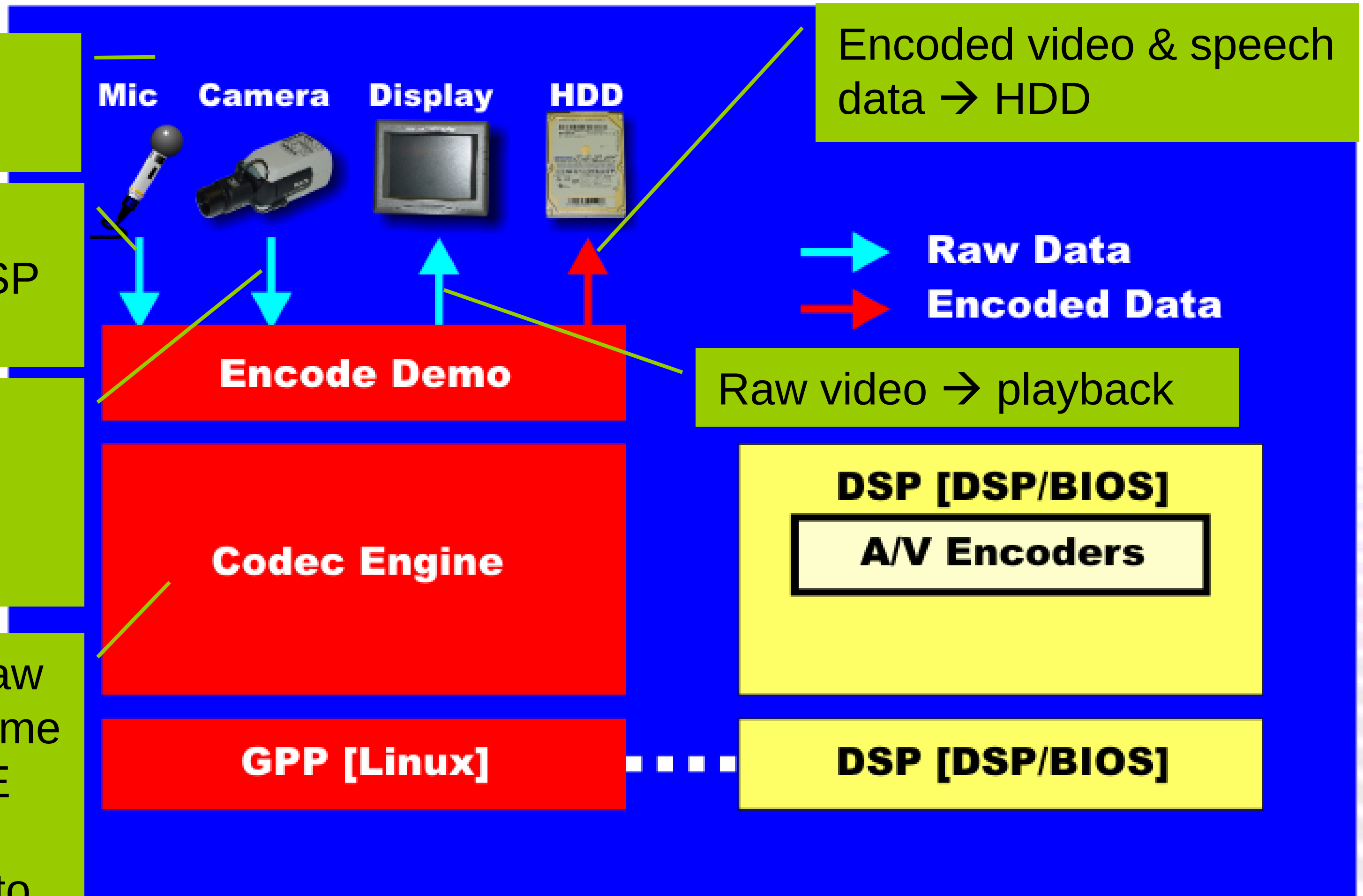
OSS driver gets samples from ASP interface

VPFE driver captures frames from composite interface

- ARM delivers raw data to DSP frame by frame via CE
- Encoded data delivered back to ARM

Encoded video & speech data → HDD

Raw video → playback

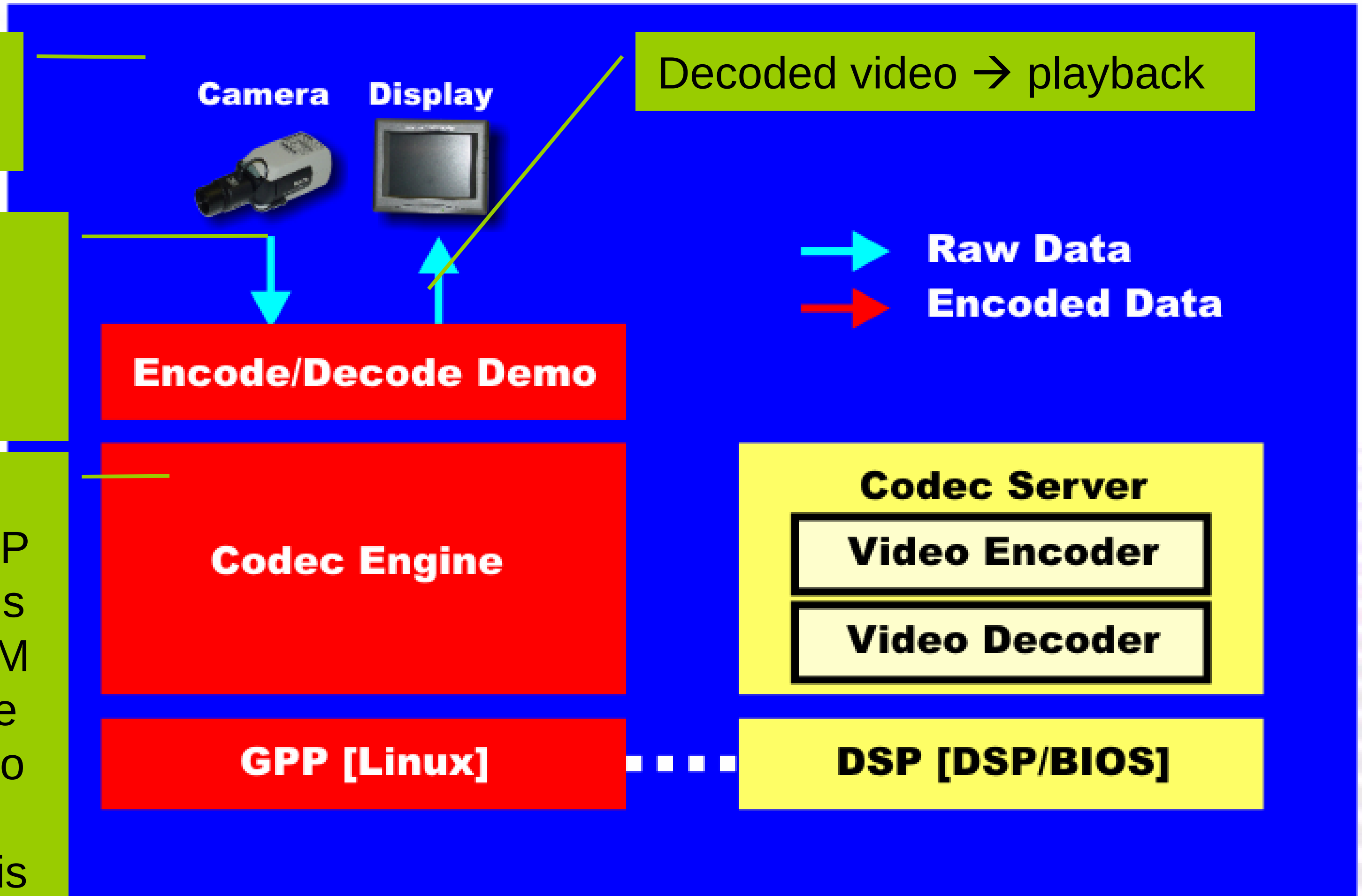


Encode/Decode Demo

Encode + decode video

VPFE driver captures frames from composite interface

- ARM delivers a raw frame to DSP
- Encoded frame is sent back to ARM
- ARM delivers the encoded frame to DSP
- Decoded frame is sent back to ARM



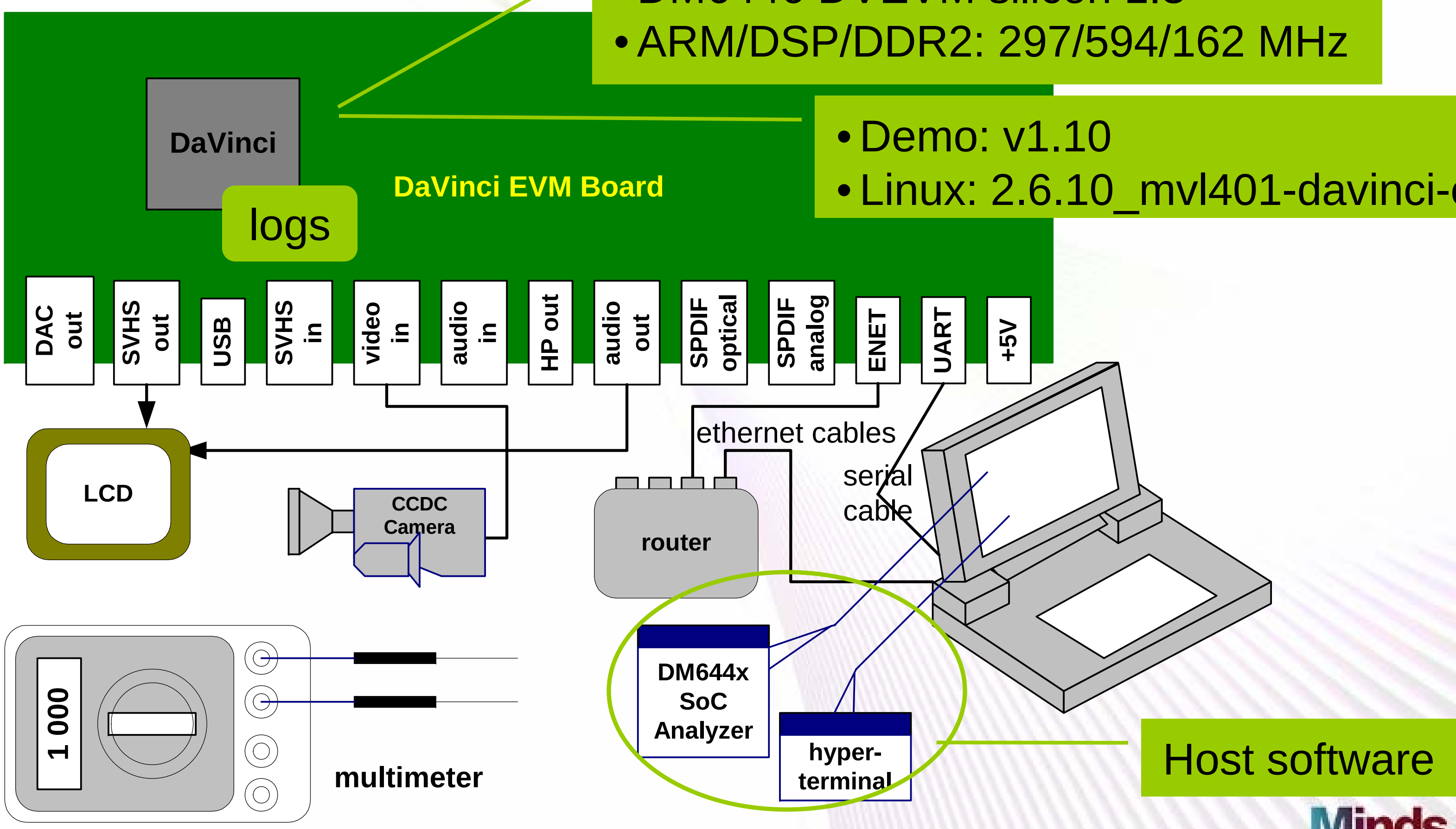
Presentation Agenda

- Introduction
- DVEVM software description
- Necessary equipment and software
- Measuring the processor loading on the demo programs
- Demo memory usage
- Power measurements on the demos
- H.264 decode demo
- Conclusion

Setup: Hardware

- DM6446 DVEVM silicon 1.3
- ARM/DSP/DDR2: 297/594/162 MHz

- Demo: v1.10
- Linux: 2.6.10_mvl401-davinci-evm



Setup: Software

- Data capture in real-time
  - BIOS logs, Linux Trace Toolkit (LTT), Codec Engine Trace
- Data transported via HTTP connection after measurement being done

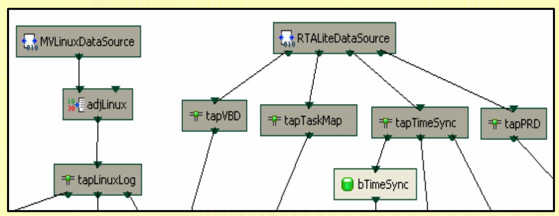
Build application to include
- Trace-enabled Linux
- Log-enabled BIOS
- Trace-enabled Codec Engine

Run application to capture logs

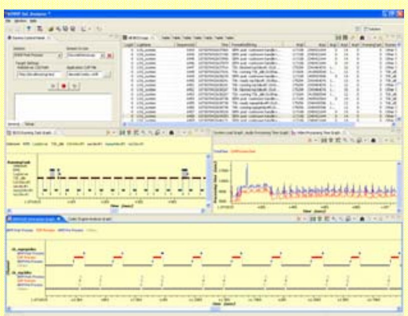
Run DVDP Analyzer to transport and visualize data

Transport logs via TCP/IP

Store data in file for reuse



Data is decoded, analyzed, correlated


Visualization

Minds in Motion

Presentation Agenda

- Introduction
- DVEVM software description
- Necessary equipment and software
- Measuring the processor loading on the demo programs
- Demo memory usage
- Power measurements on the demos
- H.264 decode demo
- Conclusion

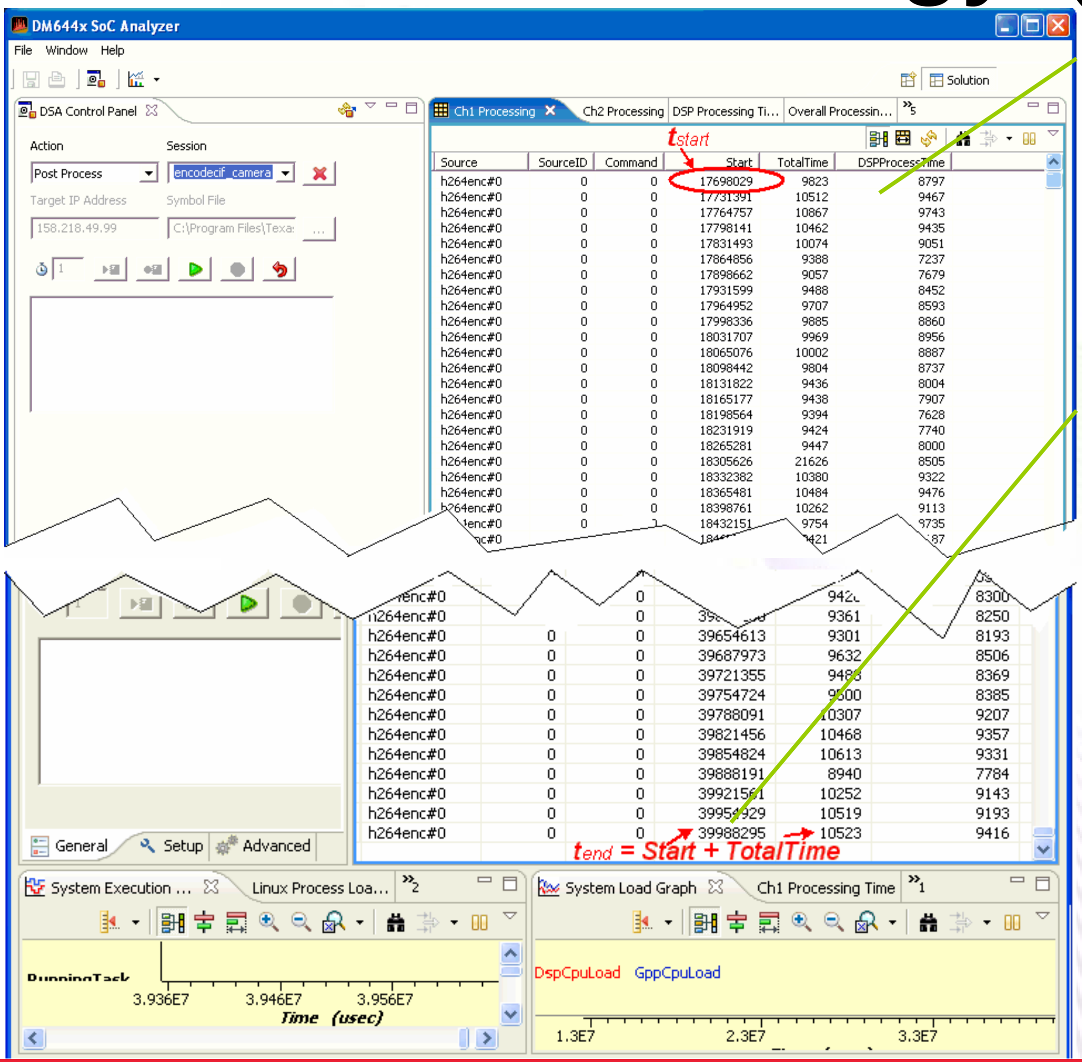
Demo Configuration (1 of 2)

- Linux kernel 2.6.10_mvl401-davinci_evm
- Demo version 1.10
- DM644x SoC Analyzer, beta version
- Decode demo
  - H.264 video decoder
    - File davincieffect_ntsc.264
    - NTSC D1 (720x480 pixels/frame), 4mbps, 30fps
  - AAC audio decoder
    - File davincieffect.aac
    - 160kbps, 48KHz, 1K samples/frame, 48fps
  - 20s

Demo Configuration (2 of 2)

- Encode demo
  - H.264 video encoder
    - NTSC D1 (720x480 pixels/frame), 4mbps, 30fps
    - Hand-waving, twice per second, 1 inch from camera
    - 20s
- Encode/Decode demo
  - H.264 video encoder + decoder
    - CIF (352x288 pixels/frame), 1mbps, 30fps
    - Hand-waving, twice per second, 1 inch from camera
    - 20s

Terminology (1 of 4)



t_{start}

- 1 channel: Ch1 processing table, 1st row, "Start"
- 2 channels: min{t_{start_ch1}, t_{start_ch2}}

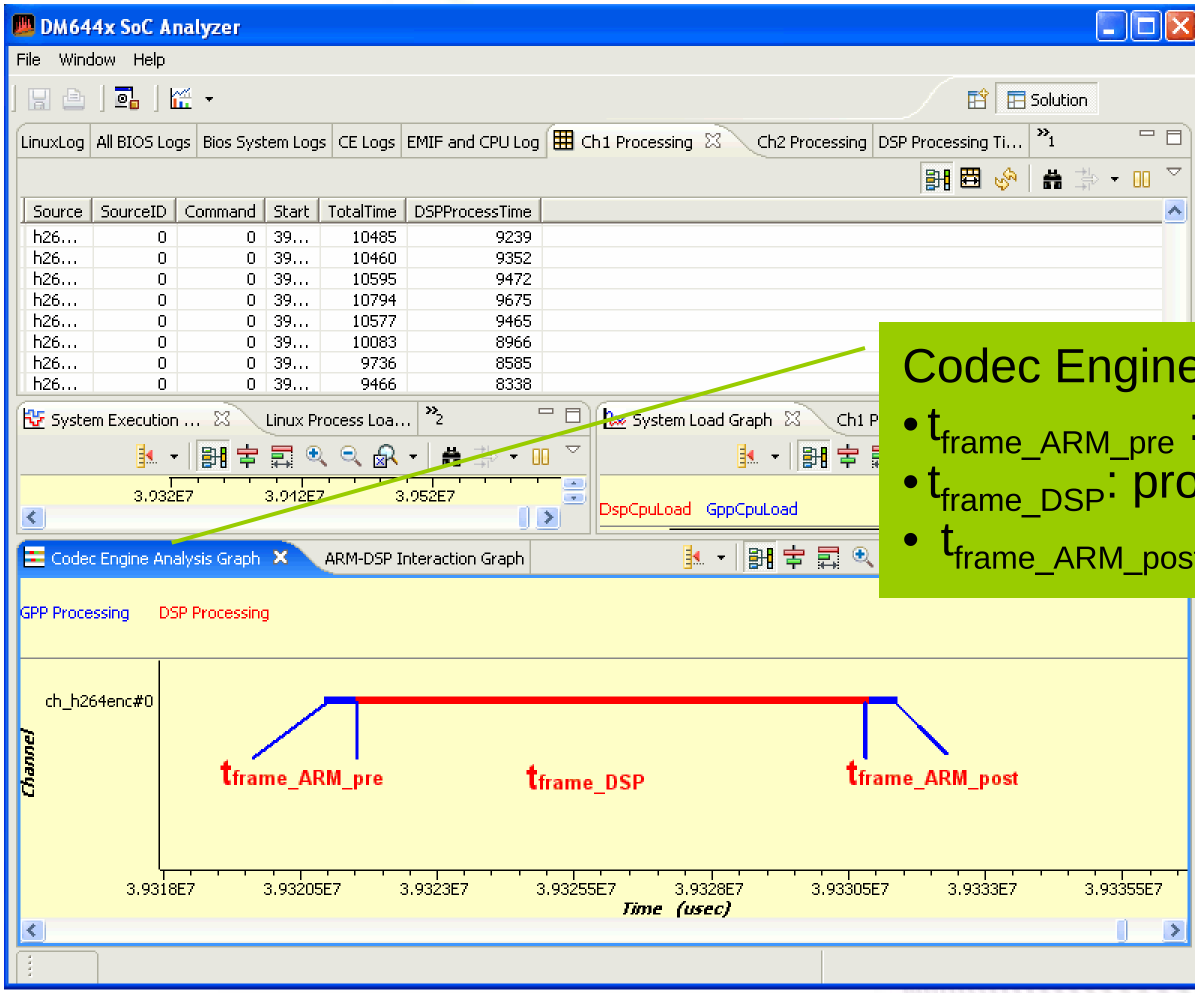
t_{end}

- 1 channel: Ch1 processing table, last row, "Start" + "TotalTime" column
- 2 channels: max{t_{end_ch1}, t_{end_ch2}}

$t_{measure} = t_{end} - t_{start}$

Minds in Motion

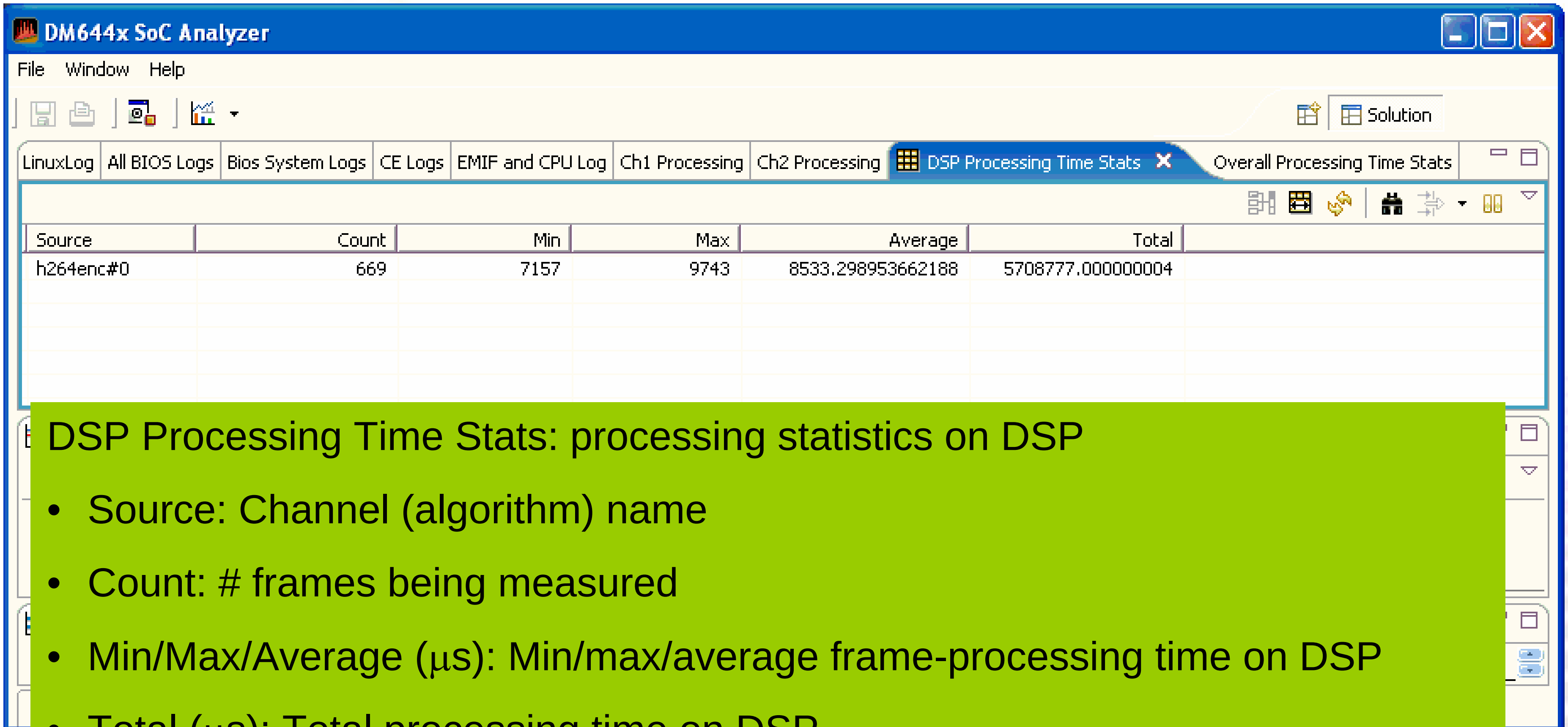
Terminology (2 of 4)



Codec Engine Analysis Graph

- $t_{frame\_ARM\_pre}$: preprocessing on ARM
- t_{frame_DSP}: processing on DSP
- $t_{frame\_ARM\_post}$: postprocessing on ARM

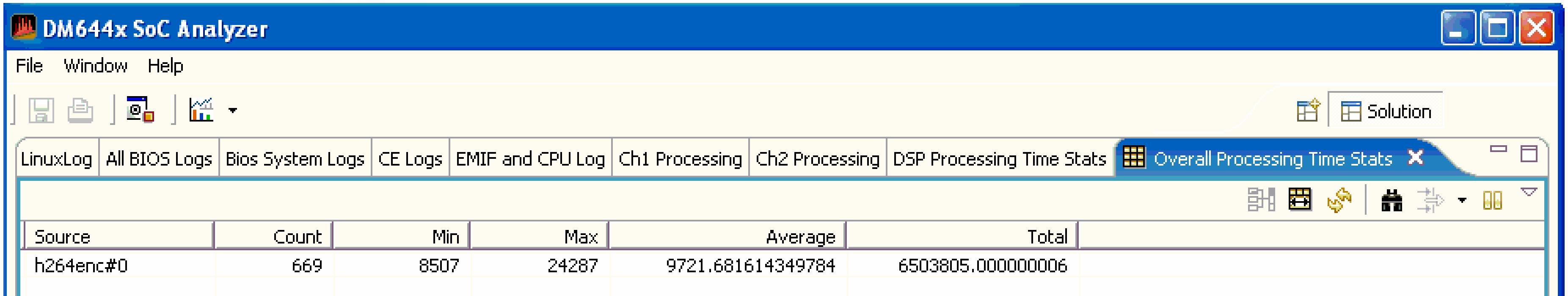
Terminology (3 of 4)



DSP Processing Time Stats: processing statistics on DSP

- Source: Channel (algorithm) name
- Count: # frames being measured
- Min/Max/Average (μs): Min/max/average frame-processing time on DSP
- Total (μs): Total processing time on DSP

Terminology (4 of 4)



Overall Processing Time Stats:

- Processing statistics including ARM pre-processing, DSP processing, ARM post-processing
- Source: Channel (algorithm) name
- Count: # frames being measured
- Min/Max/Average (µs): min/max/average processing time for each frame
- Total (µs): Total processing time

DSP Loading Calculation

- Use DSP Processing Time Stats table
- Video channel
  - $L_{min\_video\_DSP} = min\{t_{video\_frame\_DSP}\}/33,333$ us
  - $L_{max\_video\_DSP} = max\{t_{video\_frame\_DSP}\}/33,333$ us
  - $L_{avg\_video\_DSP} = t_{video\_frame\_avg\_DSP} / 33,333$ us
- Audio channel
  - $L_{min\_aac\_DSP} = min\{t_{aac\_frame\_DSP}\}/20,833$
  - $L_{max\_aac\_DSP} = max\{t_{aac\_frame\_DSP}\}/ 20,833$
  - $L_{avg\_aac\_DSP} = t_{aac\_frame\_avg\_DSP} / 20,833$
- $L_{avg\_DSP} = L_{avg\_ch1\_DSP} (+ L_{avg\_ch2\_DSP})$

Overall Loading Calculation

- Use Overall Processing Time Stats table
- Video channel
  - $L_{min\_video\_overall} = min\{t_{video\_frame\_overall}\}/33,333$
  - $L_{max\_video\_overall} = max\{t_{video\_frame\_overall}\}/33,333$
  - $L_{avg\_video\_overall} = t_{video\_frame\_avg\_overall} / 33,333$
- Audio channel
  - $L_{min\_aac\_overall} = min\{t_{aac\_frame\_overall}\}/20,833$
  - $L_{max\_aac\_overall} = max\{t_{aac\_frame\_overall}\}/ 20,833$
  - $L_{avg\_aac\_overall} = t_{aac\_frame\_avg\_overall} / 20,833$
- $L_{avg\_overall} = L_{avg\_ch1\_overall} (+ L_{avg\_ch2\_overall})$

ARM Loading Calculation

- Use overall loading and DSP loading results
- Video channel
  – $L_{avg\_video\_ARM} = L_{avg\_video\_overall} - L_{avg\_video\_DSP}$
- Audio channel
  – $L_{avg\_aac\_ARM} = L_{avg\_aac\_overall} - L_{avg\_aac\_DSP}$
- $L_{avg\_ARM} = L_{avg\_ch1\_ARM} (+ L_{avg\_ch2\_ARM})$

Decode Demo Loading Results

$L_{min\_H264\_DSP}$	$L_{max\_H264\_DSP}$	$L_{avg\_H264\_DSP}$
33.02%	67.59%	49.03%

$L_{min\_aac\_DSP}$	$L_{max\_aac\_DSP}$	$L_{avg\_aac\_DSP}$
3.12%	5.23%	3.51%

L_{avg_DSP}
52.50%

$L_{min\_H264\_overall}$	$L_{max\_H264\_overall}$	$L_{avg\_H264\_overall}$
36.41%	79.72%	52.27%

$L_{min\_aac\_overall}$	$L_{max\_aac\_overall}$	$L_{avg\_aac\_overall}$
7.79%	74.53%	10.67%

$L_{avg\_H264\_ARM}$	$L_{avg\_aac\_ARM}$
3.24%	7.09%

L_{avg_ARM}
10.33%

Encode Demo Loading Results

$L_{min\_H264\_DSP}$	$L_{max\_H264\_DSP}$	$L_{avg\_H264\_DSP}$
74.63%	103.10%	87.54%

$L_{avg\_DSP} = L_{avg\_H264\_DSP}$
87.54%

$L_{min\_H264\_overall}$	$L_{max\_H264\_overall}$	$L_{avg\_H264\_overall}$
77.73%	106.36%	90.74%

$L_{avg\_H264\_ARM} = L_{avg\_ARM}$
3.20%

Encode/Decode Demo Loading Results

$L_{min\_H264enc\_DSP}$	$L_{max\_H264enc\_DSP}$	$L_{avg\_H264enc\_DSP}$	$L_{min\_H264dec\_DSP}$	$L_{max\_H264dec\_DSP}$	$L_{avg\_H264dec\_DSP}$
26.51%	36.10%	31.88%	11.75%	17.24%	15.44%

L_{avg_DSP}
47.28%

$L_{min\_H264enc\_overall}$	$L_{max\_H264enc\_overall}$	$L_{avg\_H264enc\_overall}$	$L_{min\_H264dec\_overall}$	$L_{max\_H264dec\_overall}$	$L_{avg\_H264dec\_overall}$
29.69%	39.29%	35.08%	14.69%	20.22%	18.48%

$L_{avg\_H264enc\_ARM}$	$L_{avg\_H264dec\_ARM}$
3.20%	3.04%

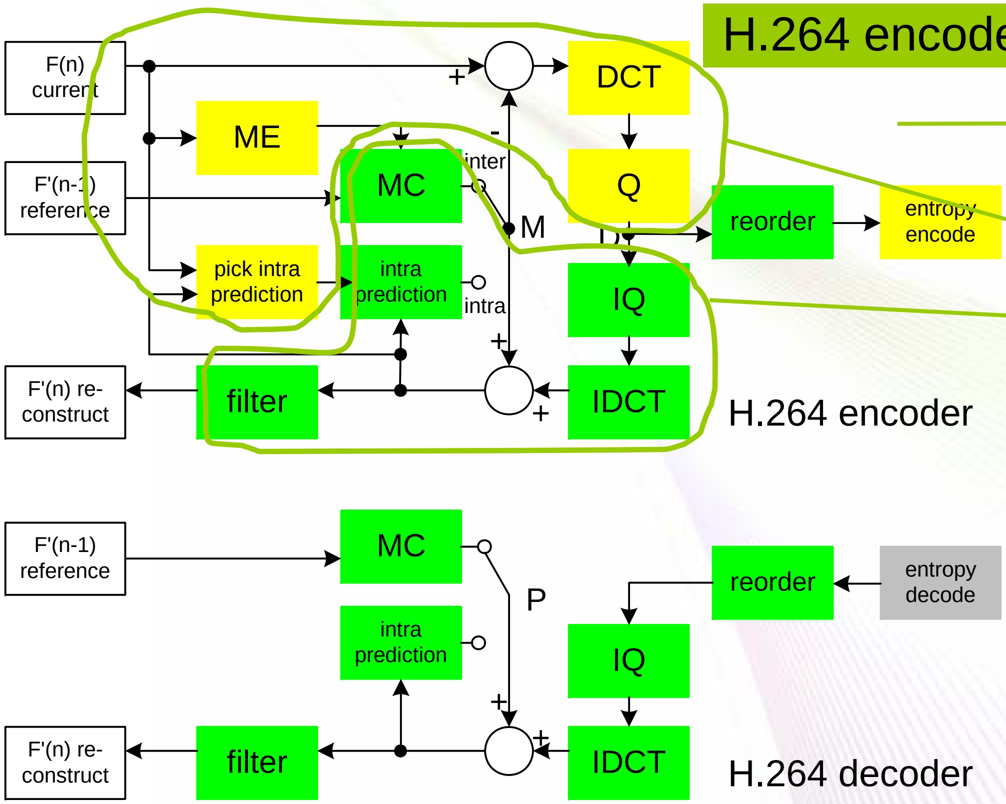
L_{avg_ARM}
6.24%

Loading Result Analysis (1 of 3)

- Overhead: 3% of video processing budget
  - An IPC → 1ms
  - 400μs from SoC analyzer: string parsing/formatting, timestamp reading/writing etc.
  - Other: cache invalidation/writeback
- > 100% loading
  - Processing time for a frame > real-time budget
  - Occasional real-time deadline violation has little impact because of buffering scheme
  - Several frames violate real-time deadline in sequence → Frame drops are noticed by the user

Loading Result Analysis (2 of 3)

H.264 encoder loading > H.264 decoder loading



MB by MB encoding

Forward path: encode MB to bits

Reconstruction path: Generate reconstructed frame. Same as decoder path.

ME tradeoff: more MIPS on ME → Higher quality

ME: motion estimation MC: motion compensation MB: macro block

(I)Q: (inverse) quantization (I)DCT: (inverse) discrete cosine transform

Loading Result Analysis (3 of 3)

- CIF:D1 = 1:4, but
  - $L_{enc\_encode/decode\_d1} > ¼ \times L_{enc\_encode\_cif}$
  - $L_{dec\_encode/decode\_d1} > ¼ \times L_{dec\_decode\_cif}$
- Two codecs in encode/decode demo → higher cache penalty.
- ME, entropy encoding/decoding do not scale linearly.

Presentation Agenda

- Introduction
- DVEVM software description
- Necessary equipment and software
- Measuring the processor loading on the demo programs
- Demo memory usage
- Power measurements on the demos
- H.264 decode demo
- Conclusion

Memory Types

- *ARM static memory*: code (.text) + data section
  – Obtained by running the Linux host utility arm_v5t_le_size
- *ARM program stack*: allocated by Linux
  – Found in the status file for the corresponding process
- *DSP static memory*: code + data section
  – Found in the generated memory map file
- *Demo dynamic memory*
  – Allocated by audio/video-processing algorithms and related drivers
  – Collected from the demo source code
    - (1) Allocated and used by the ARM app. program
    - (2) Allocated by the ARM app. program but shared both by the ARM and DSP (input/output buffers)
    - (3) Allocated by various related drivers in Linux kernel space, i.e., VPFE, VPBE, OSD
- Additional memory allocated by DSP codec is not counted.

Decode Demo Memory Usage

ARM static memory (bytes)

Code	Data	Total
121,707	11,320	133,027

ARM Program Stack Size
84K bytes

DSP static memory (bytes)

Code	Data	Total
555,264	553,480	1,108,744

Dynamic memory (bytes)

Name	Type	Size	Comments
vReadBuffer	share	3,145,728	3MB saving encoded video data.
frameBuf	share	2,488,320	Saving up to three decoded video frames in PAL D1 size.
DisplayBuf	driver	2,488,320	Allocated by display driver in kernel. Up to three D1 frames.
OSDframeBuf	driver	1,658,880	Allocated by OSD driver in kernel. Up to two D1 frames.
OSDtransBuf	driver	184,320	Allocated by OSD driver in kernel for transparency functionality.
aReadBuffer	share	61,440	60KB, saving encoded audio data.
rawBuffer	share	10,240	10KB, saving five blocks of decoded audio samples.
Total		9,965,568	The total is about 9.5MB.

Presentation Agenda

- Introduction
- DVEVM software description
- Necessary equipment and software
- Measuring the processor loading on the demo programs
- Demo memory usage
- Power measurements on the demos
- H.264 decode demo
- Conclusion

Importance of Power Consumption Data

- Power supply design/battery life estimation
- Board thermal analysis
- Effective board design
  - Cost reduction
  - Increased reliability

Terminology

The overall average power consumption for a complementary metal-oxide semiconductor (CMOS) circuit is the sum of static and active power consumption.

$$P_{total} = P_{static} + P_{active}$$

$$P_{static} = V \times I_{leakage}(V, Temp)$$

Where V is the supply voltage and $I_{leakage}$ is the leakage current at that voltage and temperature. Static power is the power dissipated when transistors are not switching.

$$P_{active} \approx Core(A \times CV^2 f) + I/O(N \times C_{io} V_{io}^2 f_{io})$$

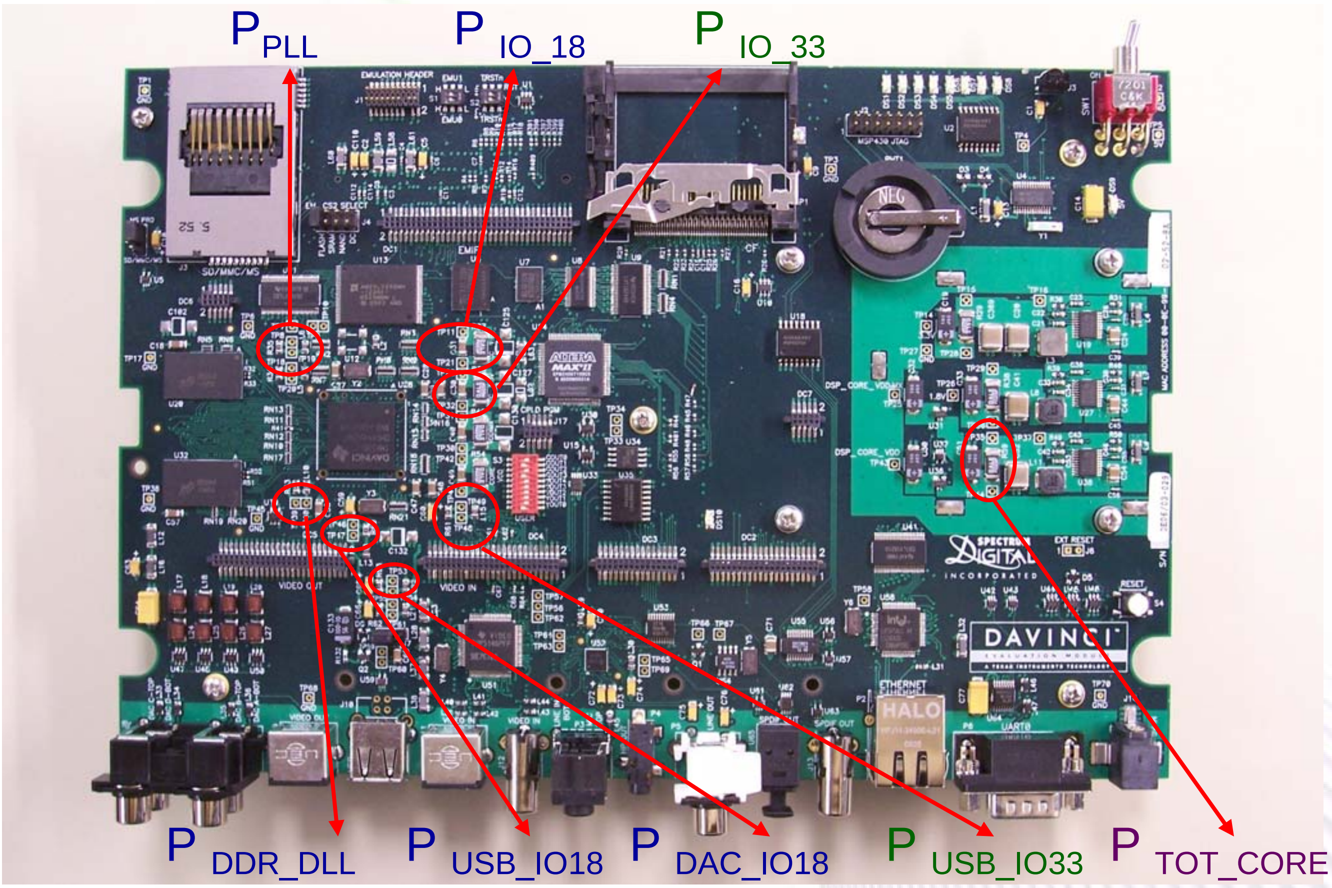
Core: C: node capacitance, V: core supply voltage, *f* : switching frequency, A: fraction of gates actively switching

I/O: C_{io}: pin/pad capacitance, V_{io}: I/O supply voltage, f_{io}: toggle rate and N: number of bits/pins switching

DM644x DVEVM Test Point Description

	Name	Test Points	Description
Core Supply Test Points	P_{TOT_CORE}	TP35-TP44	Power consumed by the core logic in the "Always On" domain, the DSP subsystem domain and DAC analog core power
1.8 V I/O Supply Test Points	P_{PLL}	TP19-TP8	Power consumed by the by PLL1 and PLL2
	P_{IO_18}	TP21-TP11	Power consumed by the 1.8V I/O, the DDR2 I/O, system oscillator and USB oscillator
	P_{DDR_DLL}	TP39-TP40	Power consumed by the DDR2 DLL
	P_{DAC_IO18}	TP53-TP54	Power consumed by the DAC analog I/O
	P_{USB_IO18}	TP46-TP47	Power consumed by the 1.8V I/O of the USB PHY
3.3 V I/O Supply Test Points	P_{IO_33}	TP32-TP22	Power consumed by all the 3.3 V I/O
	P_{USB_IO33}	TP48-TP49	Power consumed by the analog portion of the USB PHY

DM644x DVEVM Power Test Points



P_* : 1.2 V

P_* : 1.8 V

P_* : 3.3 V

Power Consumption on the Demos

	P_{TOT_CORE} (mW)	P_{PLL} (mW)	P_{IO_18} (mW)	P_{DDR_DLL} (mW)	P_{DAC_IO18} (mW)	P_{USB_IO18} (mW)	P_{IO_33} (mW)	P_{USB_IO33} (mW)	Total Power (mW)
Decode Demo (D1)	686	76	58	10	8	37	13	12	900
Encode Demo (D1)	782	76	58	11	8	37	13	12	997
Enc/Dec Demo (CIF)	686	76	50	11	8	37	13	12	893

DSP/ARM/DDR2: 594/297/162 MHz, CVdd/CVdddsp = 1.2V
Nominal silicon, room temperature

Power Summary

	Core Power 1.2V (mW)	1.8 V IO Power (mW)	3.3 V IO Power (mW)	Total Power (mW)
Decode Demo (D1)	686	189	25	900
Encode Demo (D1)	782	190	25	997
Enc/Dec Demo (CIF)	686	182	25	893

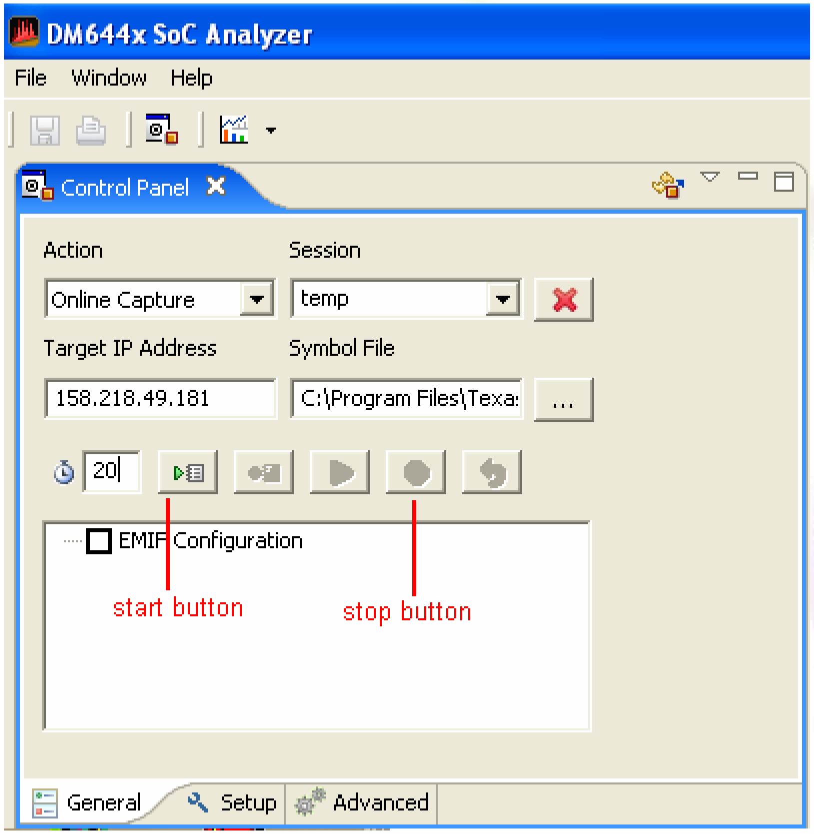
Things to Note

- Primary factors affecting the power consumption on the demos are:
  - ARM and DSP loading/utilization.
  - DDR2 IO activity (CIF/D1, buffering schemes, reads versus writes).
- High CPU loading does not mean high power. It depends both on:
  - Cycle efficiency (instructions per cycle).
  - Relative time the CPU(s) spend in active versus idle threads.
- VICP module if used will also contribute to additional power.
  - H.264 encode versus H.264 decode
- Fixed power consumption on the 3.3V I/O rails even when no 3.3V IO are in use:
  - USB PHY was not powered down.
  - Board dependent, pull-up/pull-down configuration on the 3.3V I/Os.

Presentation Agenda

- Introduction
- DVEVM software description
- Necessary equipment and software
- Measuring the processor loading on the demo programs
- Demo memory usage
- Power measurements on the demos
- H.264 decode demo
- Conclusion

Steps



- Start DM644x SoC Analyzer on host.
- Click the "Control Panel Tab."
- Select "Online Capture," give session name.
- Input target IP address.
- Select "Symbol File," i.e., encodeCombo.x64P for encode demo.
- Input measure interval (in seconds).
- Start demo on target.
- Click the "Start" button.
- Wait until the timer expires or click the "Stop" button … The logged data are pulled from target to host.
- Post-process and display results.

Reproducing Loading Results

- Decode demo
  - DSP loading depends on bit rate, motion in input stream.
  - Measure the same video/audio file from 15s to 35s.
- Encode demo
  - DSP loading depends on motion, background texture, brightness, noise, encoding bit rate, etc.
  - Different gesture → different loading.
  - Different camera → different loading.
- Encode/Decode demo
  - Same as encode demo.

Presentation Agenda

- Introduction
- DVEVM software description
- Necessary equipment and software
- Measuring the processor loading on the demo programs
- Demo memory usage
- Power measurements on the demos
- H.264 decode demo
- Conclusion

Conclusion

- This presentation provided ways to analyze performance and power using the hardware and software provided with the DM644x DVEVM/DVSDK.
- Performance was measured by examining the individual loading for each codec channel, and the overall loading for the demo was calculated for the ARM and DSP processors making use of the DM644x SoC Analyzer.
- Power consumption was measured by using various core and I/O power supply test points on the DVEVM while running the video codec demos.
- The tools and methodology described here can also be utilized by a developer to evaluate and understand performance and power requirements in a system context for his or her own DM644x-based application software.

References

- “DaVinci System Level Benchmarking Measurements,” SPRAAF6
- “Codec Engine Application Developer User’s Guide,” SPRUE67a
- “DVEVM Getting Started Guide,” SPRUE66
- DSA online documentation available from the help menu in the DM644x SoC Analyzer tool
- “TMS320 C6000 DSP Cache User’s Guide,” SPRU656A
- DaVinci EVM Technical Reference Guide (Spectrum Digital)

Encode Demo Memory Usage

ARM static memory (bytes)

Code	Data	Total
117,109	11,088	128,197

ARM Program Stack Size
84K bytes

DSP static memory (bytes)

Code	Data	Total
341,248	479,769	821,017

Dynamic memory (bytes)

Name	Type	Size	Comments
encodedBuf	share	691,200	Saving encoded frame, up to 1 uncompressed frame.
capCtrlBuf	ARM	16	Controlling capture buffer.
captureBuf	share	2,488,320	Allocated by capture driver. Saving up to three video frames in PAL D1 size.
DisplayBuf	driver	2,488,320	Allocated by display driver in kernel. Up to three D1 frames.
OSDframeBuf	driver	1,658,880	Allocated by OSD driver in kernel. Up to two D1 frames.
OSDtransBuf	driver	184,320	Allocated by OSD driver in kernel for transparency functionality.
Total		7,511,056	The total is about 7.2MB.

Encode/Decode Demo Memory Usage

ARM static memory (bytes)

Code	Data	Total
114,000	11,040	125,040

ARM Program Stack Size
84K bytes

DSP static memory (bytes)

Code	Data	Total
381,088	468,545	849,633

Dynamic memory (bytes)

Name	Type	Size	Comments
encodedBuf	share	1,658,880	Output buffer for encoder. Saving up to two uncompressed frames.
captureBuf	share	2,488,320	Allocated by capture driver. Saving up to three video frames in PAL D1 size.
capCtrlBuf	ARM	24	Controlling capture buffer.
frameBuf	share	506,880	Output buffer for decoder. Saving up to three decoded video frames in CIF size.
DisplayBuf	driver	2,488,320	Allocated by display driver in kernel. Up to three D1 frames.
OSDframeBuf	driver	1,658,880	Allocated by OSD driver in kernel. Up to two D1 frames.
OSDtransBuf	driver	184,320	Allocated by OSD driver in kernel for transparency functionality.
Total		8,985,624	The total is about 8.6MB.