Energy Efficient Video Encoding Using the Tegra K1 Mobile Processor

Kristoffer Robin Stokke, Håkon Kvale Stensland, Carsten Griwodz, Pål Halvorsen

Simula Research Laboratory & University of Oslo, Norway
{krisrst, haakonks, griff, paalh}@ifi.uio.no

ABSTRACT

Energy consumption is an important concern for mobile devices, where the evolution in battery storage capacity has not followed the power usage requirements of modern hardware. However, innovative and flexible hardware platforms give developers better means of optimising the energy consumption of their software. For example, the Tegra K1 System-on-Chip (SoC) offers two CPU clusters, GPU offloading, frequency scaling and other mechanisms to control the power and performance of applications. In this demonstration, the scenario is live video encoding, and participants can experiment with power usage and performance using the Tegra K1's hardware capabilities. A popular power-saving approach is a "race to sleep" strategy where the highest CPU frequency is used while the CPU has work to do, and then the CPU is put to sleep. Our own experiments indicate that an energy reduction of 28 % can be achieved by running the video encoder on the lowest CPU frequency at which the platform achieves an encoding frame rate equal to the minimum frame rate of 25 Frames Per Second (FPS).

Categories and Subject Descriptors

C.1.3 [**Other Architecture Styles**]: Heterogeneous (hybrid) systems;
C.1.4 [**Parallel Architectures**]: Mobile processors;
I.4.0 [**General**]: Image processing software;
J.2.0 [**Physical Sciences and Engineering**]: Electronics

General Terms

Experimentation, Measurement

Keywords

Demonstration, video encoding, real-time, energy, power

MMSys'15, Mar 18-20, 2015, Portland, OR, USA
ACM 978-1-4503-3351-1/15/03.
http://dx.doi.org/10.1145/2713168.2713186.

1. INTRODUCTION

Energy consumption is an important aspect for the usability of mobile devices, where the evolution in battery technology has not kept pace with the increasing power usage of the devices [4]. This has not gone unnoticed by hardware producers, who are developing more power-efficient and flexible SoC architectures. For example, NVIDIA's Tegra K1 [6] is a highly flexible mobile multicore SoC equipped with one low-performance, low-power CPU cluster, one high-performance, high power CPU cluster as well as a GPU with 192 CUDA cores. The platform gives full control of these capabilities to the developer. For example, heavy, parallel computing operations can be offloaded to the GPU while executing on the power efficient CPU cluster to save energy. The challenge is to understand how these choices impact runtime performance and energy consumption, and whether any energy can actually be saved depending on the choices of the programmer.

In this demonstration, we let the participant tweak the hardware settings of a Tegra K1 to minimise the energy consumption of live video encoding. The challenge is to meet a performance requirement of 25 FPS. The participant is free to adjust the CPU and GPU core frequency, select the active CPU clusters and control the number of active cores (see Table 1). The encoder, called Codec 63 (C63), is a highly simplified MPEG-inspired codec created for teaching purposes (see Section 2.2 for further details). The effects of the participant's hardware settings in terms of the achieved frame rate, power usage and CPU/GPU frequency are presented over time on a dedicated laptop. The laptop also handles live video decoding (see Figure 1). Although C63 cannot compete with the performance of a dedicated hardware encoder, the point is here to study the effects of the

Name	Description
CPU Frequency	CPU operating frequency.
CPU Cluster	Encode using the high-power or low-power CPU cluster.
CPU Cores	Controls the number of active cores to be used while actively encoding.
GPU Frequency	GPU operating frequency.
GPU Offloading	Offload nothing, some or all frames to the GPU.

Table 1: Parameters that can be set by the participant. See Section 2.2 for further description.

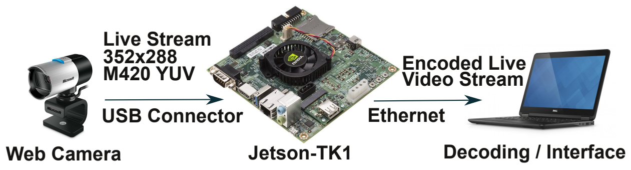

Figure 1: Demonstration set-up.

Tegra K1's hardware capabilities, and not the efficiency of the workload.

C63 can easily encode a frame in less than 40 ms, achieving a frame rate of at least 25 FPS. Although research shows that a frame rate of 12 FPS may be satisfactory with current mobile phones' screen size, we use 25 FPS as the minimum for the sake of display quality on the decoding computer. The challenge is for the participant to minimise the consumption of energy while meeting this frame rate requirement. For example, when we encoded 300 frames at 25 FPS in real-time, almost 30% of the energy was saved by minimising CPU frequency instead of running the processor on full speed to maximise the sleep period before the next frame (a "race to sleep" heuristic, see Figure 2).

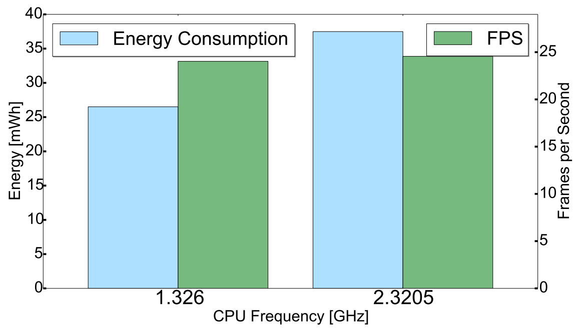

Figure 2: Our own experiments indicate almost 30% energy saving when minimising CPU frequency such that a target frame rate of 25 FPS is achieved.

2. SYSTEM OVERVIEW

Our demonstration is composed of three units (see Figure 1). A web camera is continuously transmitting a raw video stream to the encoder platform over a USB interface. The encoder platform is a Jetson-TK1 mobile development kit [5] featuring a Tegra K1 SoC. The Tegra K1 is continuously encoding the live-stream using the C63 encoder. As seen in Figure 3, we have also added a power measurement sensor that can log the energy consumption of the Jetson-TK1. Finally, the encoded stream is sent to a dedicated decoding laptop, which fulfils two tasks. First, it decodes the stream and plays it back live. Second, it displays the achieved performance- and power-related parameters such as the power usage, energy per frame and frame rate over time. The participant may use the decoding laptop to control the parameters in Table 1. Throughout the rest of this section, we will explain the system set-up in more detail.

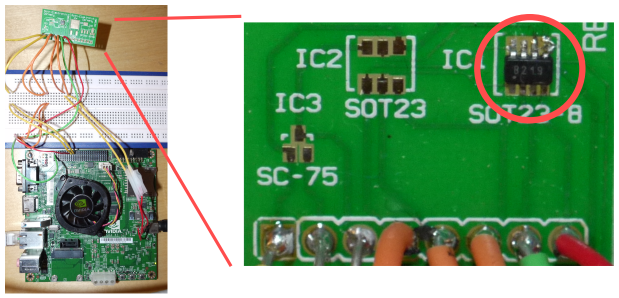

Figure 3: The figure shows our power measurement extension (INA219, circled) on the Jetson-TK1.

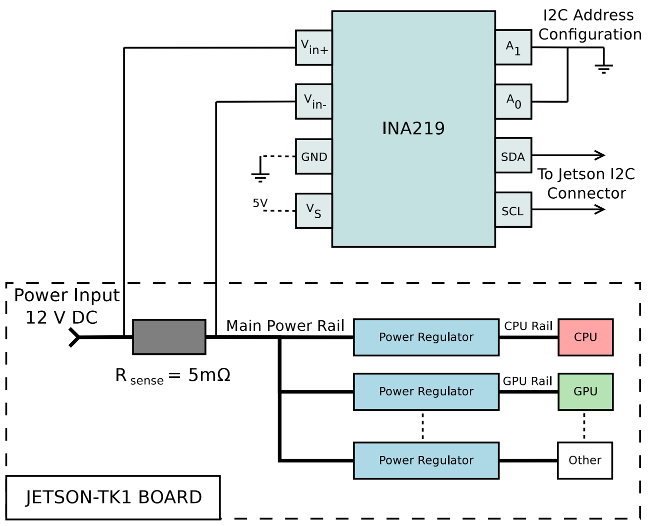

Figure 4: The INA219 measures total platform power usage over the Jetson's main power rail.

2.1 Video Encoding Platform

We use a Tegra K1 multiprocessor SoC as our encoding platform (see Figure 3). This processor is especially interesting in terms of energy consumption, because it provides many hardware capabilities that can be used for power optimisation. The Tegra K1, as well as SoCs with similar capabilities, have been implemented in real devices such as NVIDIA's SHIELD tablet. The Tegra K1's capabilities are as follows:

- **CPU**: Two clusters in "4+1" core configuration[7, 8].
  - One high performance, high power (HP) cluster with four Cortex-A15 ARM cores. Three of the cores can be individually shut down.
  - One low performance, low power (LP) cluster with a single Cortex-A15 ARM core.
  - Hardware-supported migration of OS and applications between the clusters.
  - A 128-bit NEON single instruction, multiple data (SIMD) instruction set tailored for multimedia workloads.

  - 22 configurable CPU frequencies (between 51 MHz and 2.32 GHz).

- **GPU**: 192 programmable CUDA-cores based on the Kepler-architecture on-chip.
  - 15 configurable GPU frequencies (between 72 and 852 MHz).

The participant is free to change the CPU and GPU frequencies, as well as to enable and disable cores and switch CPU clusters. The relationship between the core frequency and power is given by the Dynamic Voltage and Frequency Scaling (DVFS) formula and is especially useful here [2]:

$$P_{core} = \alpha C V_{core}^2 f_{core} \tag{1}$$

In Equation 1, α is the core utilisation level, C is the core switching capacitance, V_{core} is the core voltage and f_{core} is the core frequency. The DVFS formula shows that higher performance in terms of processor frequency can be traded for increased power consumption, allowing fine-grained tuning of power and performance characteristics depending on application requirements.

The Jetson-TK1 platform is not pre-equipped with any power measurement sensors. Consequently, we have equipped our platform with an INA219 power measurement sensor [9] (see Figures 3 and 4) and developed a Linux kernel module that provides access to the device through the sysfs filesystem. The INA219 works by measuring the voltage drop, U_{sense}, over a sense resistor connected in series with the main power rail of the Jetson-TK1. The electrical resistance of the sense resistor, R_{sense}, must be very small to avoid affecting the main circuitry ($R_{sense} << R_{jetson}$). The INA219 amplifies the voltage drop U_{sense}, converts it to the digital domain, and calculates the board's current drain I_{jetson} as follows:

$$I_{jetson} = \frac{U_{sense}}{R_{sense}} \tag{2}$$

The power consumption, P_{jetson}, at any time, is given by:

$$P_{jetson} = I_{jetson} U_{jetson} \tag{3}$$

where U_{jetson} is the main rail bus voltage potential. The set-up is similar to that found in PowerScope [3], but more complex because the INA219 is a small, surface mounted device that requires a high degree of manual configuration. There is also no separate logging machine; power logging is done by the Tegra K1 itself, eliminating as much latency as possible between the sensor and the Jetson-TK1. The INA219 itself interfaces over an I2C bus adapter. Reading directly from the CPU, we achieve a rate of 4000 power measurement samples per second using this set-up.

2.2 Video Encoding Framework

As workload for our demo we use our own implementation of the C63 video encoder, which is suitable for parallelisation using the Jetson-TK1's hardware capabilities. C63 is capable of encoding raw YUV 4:2:0, and the encoding operations are similar to H.264 or Google's VP8. The operations are as follows (see Figure 5):

- *Motion vector search:* The encoder divides the current frame into a set of macroblocks, and attempts to estimate their displacement between frames.
- *Motion compensation:* The encoder compensates the current frame by removing information for each macroblock where a motion vector could be found. This reduces the amount of information that must be stored (only the vector needs to be stored).
- *Discrete Cosine Transform (DCT) and inverse DCT (iDCT):*. The DCT transforms each macroblock to the frequency domain. The inverse DCT transforms it back. We use the "Fast-DCT" algorithm [1] on the CPU.
- *Quantisation and de-quantisation:* Quantisation reduces the value of the highest frequency components of each macroblock, to which the human eye is less sensitive to. De-quantisation transforms it back.
- *Variable-length coding (VLC):* The last step writes each frame to persistent storage using variable length huffman encoding.

The demo implementation attempts to reach a frame rate of 25 FPS by offloading none, some or all parts of the video processing to the GPU ("CPU-only", "GPU-only" or "hybrid"). For example, for the hybrid scheme, Y and U frames are transmitted to and encoded on the GPU, while the V frame is encoded on the CPU (see Figure 5). Frame writing is interleaved, such that the next frame starts processing while the last is currently being written to network. If the single frame encoding time is below the requirement (for example 40 ms for 25 FPS), the frame rate is met, and the CPU (and GPU) idles. In this case, the encoder sleeps for the remaining duration. Figure 6 illustrates an example:

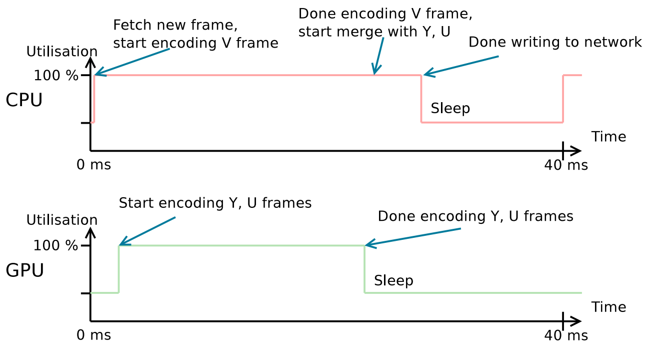
Fetch new frame, start encoding V frame
Done encoding V frame, start merge with Y, U
Done writing to network
Utilisation
100 %
CPU
Sleep
Time
0 ms
40 ms
Start encoding Y, U frames
Done encoding Y, U frames
Utilisation
100 %
GPU
Sleep
Time
0 ms
40 ms

Figure 6: An illusation of single-frame encoding under the "hybrid" processing scheme.

1. The CPU fetches a raw YUV frame from the web camera.
2. The CPU distributes the Y and U frames to the GPU, starts processing the V frame, and starts writing the previously encoded frame to the network.
3. The GPU finishes processing before the CPU is done, and idles.
4. The CPU finishes its processing before the 40 ms deadline. The CPU now sleeps for the remaining duration.
5. The cycle continues for the next frame.

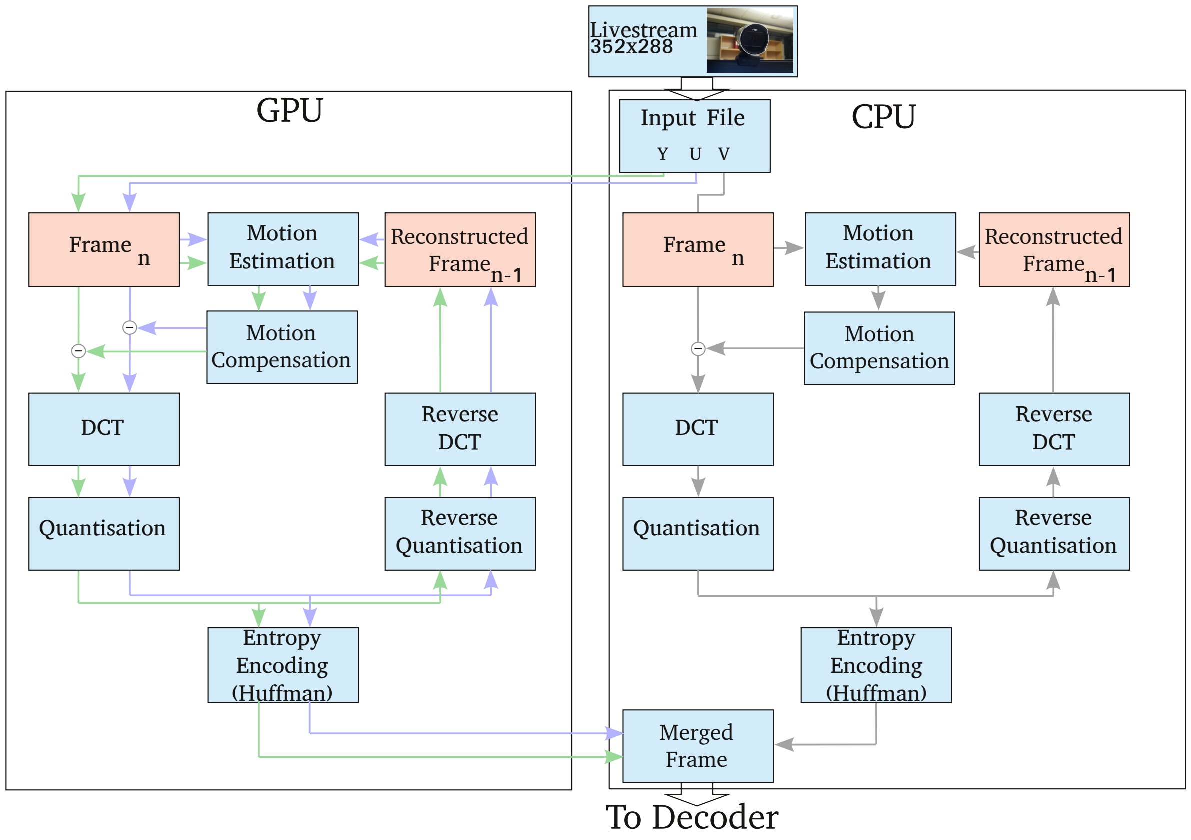

Figure 5: Our implementation of the C63 encoder operating in the "hybrid" processing scheme.

3. DEMONSTRATION

In this demo, we show how the energy consumption and performance of a video encoder is impacted by hardware and software configuration. Our demo supports runtime configuration of CPU and GPU core frequency, active CPU cluster, core shutdown and varying degrees of GPU offloading. The impact of these reconfigurations, that is power usage, frame rate, frequency and the decoded live video is displayed live to the participant as plots over time. Thus, the question is how energy-efficiently can you live-encode video?

4. REFERENCES

[1] L. V. Agostini, I. S. Silva, and S. Bampi. Pipelined fast 2d dct architecture for jpeg image compression. In *Integrated Circuits and Systems Design, 2001, 14th Symposium on.*, pages 226–231. IEEE, 2001.

[2] A. Castagnetti, C. Belleudy, S. Bilavarn, and M. Auguin. Power consumption modeling for dvfs exploitation. In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pages 579–586. IEEE, 2010.

[3] J. Flinn and M. Satyanarayanan. Powerscope: A tool for profiling the energy usage of mobile applications. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA'99. Second IEEE Workshop on*, pages 2–10. IEEE, 1999.

[4] K. Lahiri, S. Dey, D. Panigrahi, and A. Raghunathan. Battery-driven system design: A new frontier in low power design. In *Proceedings of the 2002 Asia and South Pacific Design Automation Conference*, page 261. IEEE Computer Society, 2002.

[5] NVIDIA. Jetson-TK1 Embedded Development Platform., 2014. http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html.

[6] NVIDIA. Tegra K1 Next-Get Mobile Processor., 2014. http://www.nvidia.com/object/tegra-k1-processor.html.

[7] NVIDIA. Tegra K1 Whitepaper., 2014. www.nvidia.com/content/PDF/tegra_white_-papers/Tegra-K1-whitepaper-v1.0.pdf.

[8] NVIDIA. Variable SMP., 2014. www.nvidia.com/content/PDF/tegra_white_-papers/Variable-SMP-A-Multi-Core-CPU-Architecture-for-Low-Power-and-High-Performance.pdf.

[9] Texas Instruments. INA219 Power Rail Monitor., 2014. http://www.ti.com/product/ina219.