Task details - HWSW LU WS2013

Robert Najvirt, robert.najvirt@tuwien.ac.at, 5. 11. 2013

Reading the SD card (Task 1)

Your task is to read the video stream from the SD card (starting at address 0x00000000) with a minimum rate of 500 kB/s. You only need to implement a sufficiently large subset of the SD standard to be able to read the data (initialization and reading). You do not even need to perform error handling for the case something goes wrong. However, your implementation does have to be able to read the whole video file (ca 100 MB) in one pass requiring no more than three attempts (two resets). Note that (as far as I know) the SD standard allows an up to 25 MHz SPI clock (I do know the cards in the lab support it).

For the submission of the first task, you need to generate "checksums" of 512 byte blocks from the card. The checksum is simply the 8 least significant bits of the sum of all bytes in the block (even simpler, the 8 bit overflown sum). You should send the decimal or hexadecimal representation of the checksums either through JTAG UART or the hardware UART to the PC. You can compare your results with the files provided on the website. You can measure the speed either with a timer/counter and send it to the PC or you can simply toggle an I/O after having read each block and measure the speed using the logic analyzer or oscilloscope.

Note that a compressed video typically has temporal variations in datarate. For the video used in this course, the datarate over time (frame number) is depicted in Figure 1. Note the peaks that go over the red line which shows 500 kB/s. These speed requirements will need to be compensated for with a buffer. The faster you manage to read the SD card, the less buffering you will need. Refer to Table 1 for a comparison.

Buffer size	Reading speed
64 kB	440 kB/s
32 kB	590 kB/s
16 kB	710 kB/s
8 kB	780 kB/s
4 kB	830 kB/s
0	890 kB/s

Table 1: Required reading speed for various buffer sizes.

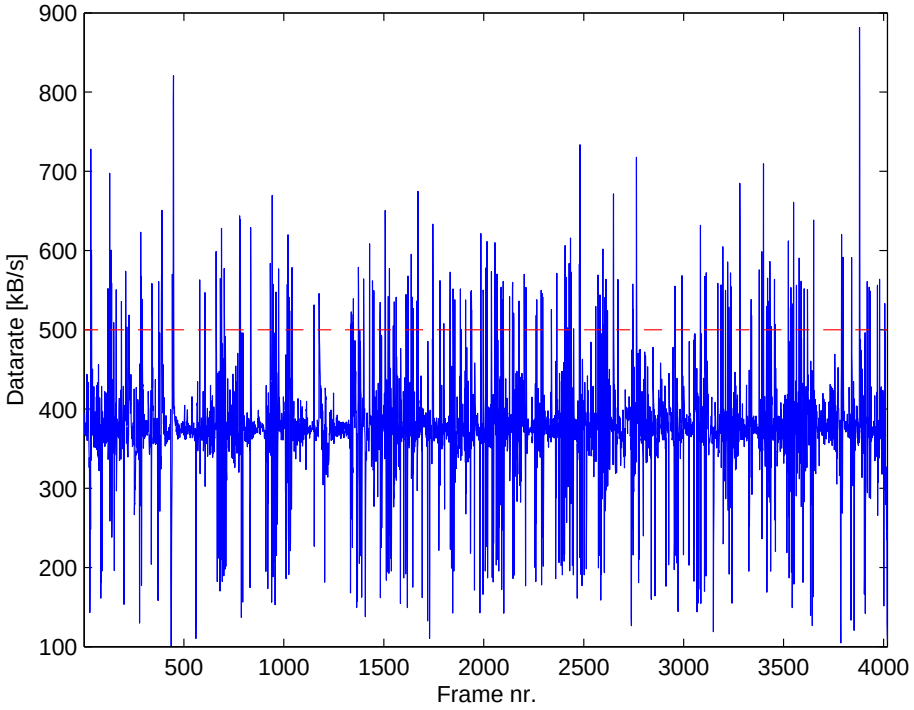

Figure 1: Video datarate.

Video decoding (Task 2)

This task contains the main part of the exercise. The assignment is simple: Decode the video stream on the SD card and display it on the attached LCD display.

The stream uses part of the Dirac video compression system. For a valid solution, you have to achieve at least a video playback with 10 fps, you will however see from the solution evaluation function that producing less fps (under 15) using less hardware resources will practically never give a better overall solution.

An important note about the requirements on your decoder: You only need to support parts of the Dirac specification that the video stream provided on the SD card uses. No other video stream we provide will use e.g. another wavelet filter, chroma sampling, etc. One more thing to know is that in addition to the "auxiliary data" and "padding" data units, you may also skip the "sequence header" data units since all parameters they define will always be the same for any video stream we provide. However, I do recommend that you decode a sequence header once (even by hand on a paper) to practice the interleaved exp-Golomb data format and navigating the Dirac specification document.

Everything you need to know about the stream is in the stream itself. For example, simply decoding all parse info headers and sending the value of *PARSE_CODE* to the PC will reveal what kinds of pictures you need to support (actually, the sequence header already reveals that). Every feature of the Dirac specification depends on some variable in the stream. Decode that variable everywhere it occurs in the stream to check whether or not you need to support some feature.

If you do not want to experience the good feeling when you decode yet another part of the stream, you can find a lot of information in the stream information file on the homepage. You may safely choose never to open the file – there is no other information than what can be read from the stream in it.

I recommend the following approach (all references address the Dirac specification):

- Start with Section 9 about the stream syntax. This already allows you to roughly see the stream contents.
- Next read Appendix A (except A.4) about data encodings. All data reading functions are defined here. Especially, the *read_uint()* and *read_sint()* functions. Using these functions and Section 10, you should be able to fully decode the sequence header.

- Now that you are familiar with navigating the specification and the data encodings, you can go to Section 11 to start decoding the picture data units, following the references to other sections of the specification. It is important to know what to expect from functions like *is_inter()*, *using_ac()* and others defined in Section 9.6.1.

Good luck!