**Format for Intermediate Reports on DSP Optimizations**

**Group Name** : Speech group 5

**Team member 1** : Koen Goetshalckx

**Team member 2** : Seppe Iven

Selfie(s). Illustrating who is who.

Seppe: Koen:

 

# Report 1

**Lab date : 22/04, 2016**

**Submission time : 23/04**

## Logs:

We have found and fixed some bugs that occurred when implementing another group’s crypto part. We installed and configured CCS 5 and went through the given slides. We made the code slightly more modular. We profiled the code running on the DSP-simulator for a first time. We tried to speed up convolve, but no significant change was realized (see discussion below).

## Profile results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Major Optimization** | **Functions Effected** | **Total Cycles\*** | | **Change (in %)** |
| **Before** | **After** |
| Base Code |  |  | 21,8M |  |  |
| Session 1 | Hardcoded filter in unrolled inner loop of convolve for filter2 and filter3 | Convolve 🡪 convolveFilter2Odd, convolveFilter2Even, convolveFilter3Odd, convolveFilter3Even | 21,8M | 21,7M | 0,5 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function Name | File Name | Total no. of calls | Exclusive Count Total\* | | | Inclusive Count Total\* | | | Optimization carried out |
| Before  (Previous Session) | After  (This Session) | % Change | Before  (Previous Session) | After  (This Session) | % Change |
| Convolve  🡪  Convolve,  convolveFilter2Odd, convolveFilter2Even, convolveFilter3Odd, convolveFilter3Even | Analysis.c | 3840 | 3,5M | 3,4M | 3 | 13,5M | 13,4M | 1 | Hardcoded filter coefficients in an unrolled inner loop |

## Discussion (important)

The inner loop of the convolve function is the most used code of the program. Before the DSP, this inner loop was already sped up by a factor of 2, using the profiler of Visual Studio and the default windows C compiler. Now, an attempt to speed it up even more was made by unrolling this inner loop and hardcoding the filter coefficients in it. Some noticeable gain was expected, considering unrolled code improves pipelining and the hardcoded coefficients should require less data accesses. However, only a very small gain in speed was realized. Therefore, and because of the unreadability of the manually unrolled code (manually to hardcode the coefficients), the changes were reverted. (Only filter2 and filter3 were hardcoded. This should be enough to test the impact of the change.)

The amount of cycles is for an input file of 2986 samples

# Report 2

Lab date: 28/4

Submission date: 29/4

Submission deadline: 29/4

## Logs:

Changed some shorts to chars (lengths and offset). This changed the total amount of cycles to 21.6M.   
Declared a lot of variables ‘near’, but this gave no noticeable change, so we reverted it for cleanliness.  
Used the keyword ‘const’ for variables that behave like constants. This gave almost no speed up at all, so we reverted it for cleanliness.  
Combined left and right in convolve -> less dependencies & overhead

## Profile results:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function Name | File Name | Total no. of calls | Exclusive Count Total\* | | | Inclusive Count Total\* | | | Optimization carried out |
| Before  (Previous Session) | After  (This Session) | % Change | Before  (Previous Session) | After  (This Session) | % Change |
| Convolve | Analysis.c | 1920 | 3.4M | 2.2M | 35% | 13.4M | 12.0M | 10% | Combine |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Major Optimization** | **Functions Effected** | **Total Cycles\*** | | **Change (in %)** |
| **Before** | **After** |
| Base Code |  |  | 21,8M |  |  |
| Session 1 | Hardcoded filter in unrolled inner loop of convolve for filter2 and filter3 | Convolve 🡪 convolveFilter2Odd, convolveFilter2Even, convolveFilter3Odd, convolveFilter3Even | 21,8M | 21,7M  REVERTED | 0,5 |
| Session 2 | Combine convolve | Convolve | 21,8M | 20.1M | 8% |

## Discussion (important)

As expected, combining left and right convolve nets a gain in cycles because both the dependencies and the overhead are reduced. A lot of the work for the left and right convolve (e.g. fetching filter coefficients) needs to be done only once. The compiler has more independent variables to use for parallelism/pipelining. A slightly larger gain was expected, however.  
Most of the small changes, such as declaring variables ‘near’ and ‘const’ are reverted because they gave almost no gain. It is not excluded that these changes may be implemented again in the future.

# Report 3

Lab date: 29/4

Submission date: 30/4

Submission deadline: 30/4

## Logs:

Changed type of ‘result’ in the inner loop of convolve to int.

Spent most of our time fixing a bug in the rewritten version of convolve (see below)

## Profile results:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function Name | File Name | Total no. of calls | Exclusive Count Total\* |  |  | Inclusive Count Total\* |  |  | Optimization carried out |
|  |  |  | Before  (Previous Session) | After  (This Session) | % Change | Before  (Previous Session) | After  (This Session) | % Change |  |
| Convolve | Analysis.c | 1920 | 2.2M | 1.1M | 50% | 12.0M | 2.0M | 83% | Long long to int |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Major Optimization** | **Functions Effected** | **Total Cycles\*** |  | **Change (in %)** |
|  |  |  | **Before** | **After** |  |
| Base Code |  |  | 21,8M |  |  |
| Session 1 | Hardcoded filter in unrolled inner loop of convolve for filter2 and filter3 | Convolve convolveFilter2Odd, convolveFilter2Even, convolveFilter3Odd, convolveFilter3Even | 21,8M | 21,7M  REVERTED | 0,5 |
| Session 2 | Combine convolve | Convolve | 21,8M | 20.1M | 8% |
| Session 3 | Long long to int | Convolve | 20.1M | 10.2M | 49% |

## Discussion (important)

We rewrote convolve: instead of doing the wrapping around to the beginning of an array in the inner loop, we now temporarily copy the array, with the copy being in the right order. This turns the inner loop into a classic multiply and add loop, which the compiler easily optimizes, at the expense of a small increase in memory usage. This only gave a small gain though.  
A much larger gain was realized by changing the intermediate result in the inner loop to an int, instead of a long long. This change may theoretically cause overflows, but implementing this exactly the same in MATLAB (our C and MATLAB code produce the exact same solution down to every bit) showed that the output signals remain unchanged.

# Report 4

Lab date: 3/5  
Submission date: 4/5  
Submission deadline: 4/5

## Logs:

Restricted several variables in quantize and dequantize. Fully inlined ‘synthesis’ and ‘analysis’ in the main. This gave about 300k cycles profit, but reverted this for cleanliness. We know that we can again implement this in the future to gain some cycles.

Changed some variable types in quantize and dequantize. This gained a lot of cycles, without changing the results (verified in matlab).

## Profile results:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function Name | File Name | Total no. of calls | Exclusive Count Total\* |  |  | Inclusive Count Total\* |  |  | Optimization carried out |
|  |  |  | Before  (Previous Session) | After  (This Session) | % Change | Before  (Previous Session) | After  (This Session) | % Change |  |
| Quantize | Quantize.c | 960 | 437k | 246k | 44% | 3.4M | 1.2M | 65% | Change variable types |
| Dequantize | Dequantize.c | 960 | 290k | 170k | 41% | 3.1M | 1.0M | 68% | Change variable types |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Major Optimization** | **Functions Effected** | **Total Cycles\*** |  | **Change (in %)** |
|  |  |  | **Before** | **After** |  |
| Base Code |  |  | 21,8M |  |  |
| Session 1 | Hardcoded filter in unrolled inner loop of convolve for filter2 and filter3 | Convolve convolveFilter2Odd, convolveFilter2Even, convolveFilter3Odd, convolveFilter3Even | 21,8M | 21,7M  REVERTED | 0,5% |
| Session 2 | Combine convolve | Convolve | 21,8M | 20.1M | 8% |
| Session 3 | Long long to int | Convolve | 20.1M | 10.2M | 49% |
| Session 4 | Restrict / change variable types | (De)quantize | 10.2M | 5.9M | 42% |

## Discussion (important)

After optimizing the convolve function, the new bottlenecks were quantize and dequantize, so we focused on these functions in this session. We removed all of the long long variable types. Changed some ints to shorts. This generally results in large improvements, as is to be expected. Inlining several functions into main.c did not net as much cycles as we were hoping for. We have (temporarily) reverted this because it cluttered main.c. This may in the future be implemented again.

# Report 5

Lab date: 6/5  
Submission date: 7/5  
Submission deadline: 7/5

## Logs:

-Restrict some variables  
-Port onto DSP  
-Merge with a crypto group  
-Check actual DSP results (!)

## Profile results:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function Name | File Name | Total no. of calls | Exclusive Count Total\* |  |  | Inclusive Count Total\* |  |  | Optimization carried out |
|  |  |  | Before  (Previous Session) | After  (This Session) | % Change | Before  (Previous Session) | After  (This Session) | % Change |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Major Optimization** | **Functions Effected** | **Total Cycles\*** |  | **Change (in %)** |
|  |  |  | **Before** | **After** |  |
| Base Code |  |  | 21,8M |  |  |
| Session 1 | Hardcoded filter in unrolled inner loop of convolve for filter2 and filter3 | Convolve convolveFilter2Odd, convolveFilter2Even, convolveFilter3Odd, convolveFilter3Even | 21,8M | 21,7M  REVERTED | 0,5% |
| Session 2 | Combine convolve | Convolve | 21,8M | 20.1M | 8% |
| Session 3 | Long long to int | Convolve | 20.1M | 10.2M | 49% |
| Session 4 | Restrict / change variable types | (De)quantize | 10.2M | 5.9M | 42% |

## Discussion (important)

Found some other variables that can use the ‘restrict’ keyword. This did not net a significant gain. Then merged our code with a crypto group. This gave some errors because of memory management, but this was fixed by putting some variables outside the stack, increasing the stack size together, removing some unnecessary test variables of the crypto and decreasing their maximum message size.   
When comparing the DSP code to the simulator code and the MATLAB code, we found a significant problem! The simulator results are exactly the same as the MATLAB results (because we made it this way), but the actual DSP results differ from the simulator results! We ran the exact same code on the simulator and the DSP, and the results were slightly different. For some sound files, this gives a slight perturbation on the results. Although it is probably not hearable, the deviation from MATLAB makes it impossible to use MATLAB to test the effect of changes to for example parameters. There was also a file where the entire speech became zeros on the DSP but not in the simulator. More examination is necessary to determine if this has the same cause as the small deviations. Either way, the cause of the deviations should be found and preferably resolved. We will try to do this, and provide visual proof of the problem, in the next session, where we have access to a DSP.

# Report 6

Lab date: 11/5  
Submission date: 13/5   
Submission deadline: 12/5

## Logs:

Optimizations performed in the time before this session

* In quantize and dequantize: removed call to division in calculation of stepsize by writing the division as a binary search
* In quantize: removed call to division in calculating quantized\_difference by writing it as a linear search (linear instead of binary because only a few values possible)
* In convolve: replaced /(1<<amountToShift). This was written this way because just >>amountToShift rounds towards negative infinity instead of towards zero. It is now rewritten to use >> properly, together with some bit manipulation to get the desired rounding towards zero behavior.
* In convolve, combine and combineWithoutDelay: removed the modulo operation in the indexing by more simple increment and if statements

In the session: spent a lot of time fixing a bug which was highly likely due to not initializing some variable/array. After that, lost some time because our DSP was wrongly configured (the little switches). Finally went home with the DSP to try to get the real time working properly. We did not succeed in this.

## Profile results:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Function Name | File Name | Total no. of calls | Exclusive Count Total\* |  |  | Inclusive Count Total\* |  |  | Optimization carried out |
|  |  |  | Before  (Previous Session) | After  (This Session) | % Change | Before  (Previous Session) | After  (This Session) | % Change |  |
| Dequantize | Dequantize.c | 960 | 167k | 563k | -233 | 1007k | 582k | 42 | Wrote divisions as binary /linear search |
| Quantize | Quantize.c | 960 | 246k | 706k | -187 | 1120k | 725 | 35 |
| Convolve | Analysis.c | 1920 | 1108k | 827k | 25 | 2023k | 1083k | 46 | Division as shift (rounding to zero) |
| Convolve | Analysis.c | 1920 | 827k | 961k | -16 | 1083k | 961k | 11 | ~~Modulo in output index~~ |
| Combine | Analysis.c | 960 | 114k | 59k | 48 | 242k | 59k | 76 | ~~Modulo in indices~~ |
| combineWithoutDelay | Synthesis.c | 960 | 120k | 97k | 19 | 248k | 97k | 61 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Major Optimization** | **Functions Effected** | **Total Cycles\*** |  | **Change (in %)** |
|  |  |  | **Before** | **After** |  |
| Base Code |  |  | 21,8M |  |  |
| Session 1 | Hardcoded filter in unrolled inner loop of convolve for filter2 and filter3 | Convolve convolveFilter2Odd, convolveFilter2Even, convolveFilter3Odd, convolveFilter3Even | 21,8M | 21,7M  REVERTED | 0,5% |
| Session 2 | Combine convolve | Convolve | 21,8M | 20.1M | 8% |
| Session 3 | Long long to int | Convolve | 20.1M | 10.2M | 49% |
| Session 4 | Restrict / change variable types | (De)quantize | 10.2M | 5.9M | 42% |
| Session 5 | Removed calls to division/reminder functions | Convolve, quantize, dequantize, combine, combineWithoutDelay | 5.9M | 3.6M | 39% |

## Discussion (important)

See Logs. Summarized, we removed all calls (which were present in assembly) to division or reminder functions. This removes overhead and avoids using these slow functions. A large gain was realized in convolve, quantize and dequantize. Less important gains were realized in the combine functions. After all the code was optimized from 21,8M samples to 3,6M. Note that the original 21,8M was after we already sped up the code by a factor of 2 with Visual Studio when running on the win32 platform.