FIXED- VERSUS FLOATING-POINT IMPLEMENTATION OF MIMO-OFDM DETECTOR

Janne Janhunen, *Perttu Salmela, †Olli Silvén, Markku Juntti

Centre for Wireless Communications
*Department of Computer Systems, Tampere University of Technology
†Computer Science and Engineering Laboratory, University of Oulu

ABSTRACT

In this paper, we investigate the opportunities offered by floating-point arithmetics in enabling an assembly and intrinsics free high-level language based development. We compare the characteristics of floating- and fixed-point arithmetics by simulating a MIMO-OFDM soft output detector in a 3G LTE link level simulator. The hardware complexity and energy dissipation are analyzed by implementing three programmable processors supporting 32- and 12-bit floating-point and 16-bit fixed-point arithmetics. The processors are based on the transport triggered architecture (TTA) that has a very low programmability overhead. The analysis shows that at the same goodput rate a floating-point implementation can achieve a lower gate count and better power efficiency than a fixed-point design.

Index Terms— fixed-point, floating-point, MIMO, SSFE, TTA

1. INTRODUCTION

Wireless communication systems have experienced tremendous development during the last two decades. The 4th generation wireless communication systems and networks will encounter several implementation challenges due to ever increasing capacity and flexibility requirements. The latter ones are the driver for software defined radio (SDR) technologies that in turn are expected to enable the creation of cognitive radios. As a result any radio solution could be invoked on demand on any platform. Designing such solutions is a huge effort that should result in an straightforwardly reusable code legacy for the future.

So far the demands for high gate and energy efficiencies in baseband processing designs have been interpreted as requirements to employ fixed-point arithmetic. In part this expectation comes from the wide single instruction multiple data (SIMD) or vector type digital signal processors (DSP) intended for radio implementations. On those architectures, fixed-point arithmetic is efficient, but achieving efficiency requires utilizing in-line assembly or intrinsic functions that can be very difficult and laborious to port to another execution platform.

Floating-point arithmetic has generally been considered power hungrier and much less gate efficient than fixed-point designs. We see this as a questionable assumption that has not been justified by detailed implementation and tool chain level analyses. The benefits of floating-point arithmetic for software developers are undeniable: high-level languages can be used without intrinsics and compilation based tool chains can be employed when porting to new platforms.

In the present study, we investigate the complexity in gate equivalent and power efficiency rankings of fixed-point and floating-point arithmetics implementations using a multiple-input multiple-output (MIMO) detector as a target case. The gate equivalent is defined as a technology-independent measure corresponding a two-input NAND gate in CMOS technology. To make the results genuinely comparable, the same programmable transport triggered processor architecture (TTA) has been employed [1] and fixed goodput rates have been targeted. The goodput is a measure which combines both the hardware limitations and the detection reliability, i.e., the minimum of the transmission throughput and hardware detection rate of information bits.

Our implementation experiments have been carried out for a detector named selective spanning with fast enumeration (SSFE) [2], a recently proposed soft output detector with favorable properties for software implementation. It is based on tree search like the well known K-best [3, 4] list sphere detector (LSD) but the symbol selection is done with a reduced complexity.

We implemented three programmable processors supporting 32- and 12-bit floating-point and 16-bit fixed-point arithmetic. To estimate the energy dissipation of the processors, each of the processors executes SSFE detection for $2 \times 2$ antenna system with 16-QAM. The implementation analyzes were performed with a low-power 130 nm CMOS process.

The rest of the paper is organized as follows. In Section 2, we summarize the computer simulation results which define the word lengths for implementations. Section 3 describes the processor implementations on TTA. The results are presented in Section 4. In Section 5, we summarize a brief comparison of energy dissipations in other implementations proposed in literature. Finally, in Section 6 we conclude the contributions of the study.

2. ERROR RATE PERFORMANCE

Fixed- and floating-point computer simulations for SSFE algorithm were carried out with 3G LTE [5] compliant link level simulator. We simulated a $2 \times 2$ antenna system with 16-QAM and a $4 \times 4$ antenna system with 64-QAM. The SSFE algorithm is characterized by the level update vector. It defines the number of child nodes spanned from each father node in the search tree and the size of the final candidate list. We refer to [2] for more detailed description of the SSFE algorithm. The first setup has a level update vector $m = [1224]$ and the latter $m = [11122223]$. Based on the computer simulations, we determined sufficient word lengths for the implementation. In general, the required word length depends on the channel conditions, the number of antennas, modulation order and detector algorithm.

We present the simulation results in Fig.1 and Fig.2. In both cases, the reduced word lengths are compared to the IEEE double precision floating-point result. For the $2 \times 2$ antenna system the difference in word length requirement between fixed- and floating-point arithmetic is only one bit. However, in the $4 \times 4$ antenna system the difference is already four bits. The 16-bit fixed-point and 12-bit floating-point arithmetics provide almost the same bit error rate and
both are close to IEEE double precision floating-point performance. The difference between double precision, 12-bit and 11-bit fixed-point arithmetics is approximately 0.3 dB (BER $10^{-3}$) and between 11-bit floating-point arithmetic little less. Using even shorter word lengths causes a detection breakdown. Due to fact that the processors are programmable and should support at least the 4 × 4 antenna system in real life, we decided to use word lengths which support both antenna cases. To provide the same bit error rate for fixed- and floating-point arithmetic, we use 16-bit (sign, 5-bit integer, 10-bit fraction) fixed-point and 12-bit (sign, 4-bit exponent and 7-bit mantissa) floating-point arithmetics in the implementations.

3. PROCESSOR IMPLEMENTATIONS

Transport triggered architecture is a programmable architecture template, in which the function units are triggered by data transports. TTA allows to design tailored processor with a chosen flexibility. The processor may resemble an ASIC design with minimum flexibility or the processor may be fully programmable. The application can be accelerated with special function units (SFU), which can be used same way as conventional FUs. The number of register files (RF) or the RF size are not restricted and they can be used as FUs. The program counter and the return address register, which is needed for jump or call operations is controlled by the global control unit (GCU). Sockets handle the data between FU ports and an interconnection network (ICN) and are controlled by the instruction word such that data are passed to and read from the correct bus. Our target is to compare performance, silicon complexity and energy dissipation between fixed- and floating-point arithmetics. To do this we have fixed the studied algorithm, simulation environment and function units in the processors.

We synthesized three different processors with a 130 nm low-power CMOS technology: 32- and 12-bit floating-point and 16-bit fixed-point processors. The processors have several FUs which can each support many operations. For instance, the logic unit supports AND, OR, inclusive OR operations. Other supported bitwise operations are shift (shift FU), greater than and equal to (compare FU). The load/store FUs are access to an external memory. The slice FU is a SFU designed to accelerate the SSFE detector execution. The arithmetic unit supports addition, subtraction, multiplication, negation and absolute value operations. There is a single Boolean (2 × 1 – bit) register file and several ”general purpose” register files. 32-bit floating-point word is divided into a sign, 8-bit exponent and 23-bit mantissa, whereas the 12-bit word has a sign, 4-bit exponent and 7-bit mantissa. The 16-bit fixed-point word has sign bit, 5-bit integer and 10-bit fraction part. The most interesting comparison is between the 12-bit floating-point and 16-bit fixed-point processors since these two cases have the same bit error rate.

The numbers of function units and their complexities in gate equivalents (GE) are summarized in Table 1. The fixed-point processor have no single cycle adders nor absolute value operation. FX denotes a fixed-point and FP a floating-point operation, respectively.

<table>
<thead>
<tr>
<th>FU (latency)</th>
<th># of FUs</th>
<th>32-bit FP</th>
<th>12-bit FP</th>
<th>16-bit FX</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD/ADD (2 cc)</td>
<td>8</td>
<td>1600</td>
<td>290</td>
<td>8090</td>
</tr>
<tr>
<td>ADD/SUB (1 cc)</td>
<td>4</td>
<td>1100</td>
<td>390</td>
<td>8090</td>
</tr>
<tr>
<td>SLICER (1 cc)</td>
<td>4</td>
<td>650</td>
<td>280</td>
<td>429</td>
</tr>
<tr>
<td>MUL (2 cc)</td>
<td>8</td>
<td>3420</td>
<td>350</td>
<td>1110</td>
</tr>
<tr>
<td>ABS (2 cc)</td>
<td>8</td>
<td>670</td>
<td>230</td>
<td>-</td>
</tr>
<tr>
<td>LSU (3/1 cc)</td>
<td>2</td>
<td>1230</td>
<td>1150</td>
<td>1140</td>
</tr>
<tr>
<td>RF (1 cc)</td>
<td>8</td>
<td>2750</td>
<td>1030</td>
<td>1380</td>
</tr>
</tbody>
</table>

3.1. Function Units

The floating-point processors include both floating-point and fixed-point adders and substracters as TTA uses fixed-point adders and substracters to generate memory addresses. Otherwise these operations are not used in the floating-point implementations. The slicer function is executed by a single cycle SFU supporting currently only 16-QAM. The logic inside the SFU compares input value to predetermined limits specified by the modulation method. Thus, the complexity of the unit is modest both in floating- and fixed-point arithmetic. The software implementation would cause expensive branches in execution, and thus, the slice operation is efficient to do with SFU.
The multipliers and adders have two clock cycle latency which reduces the silicon complexity and makes the critical path shorter such that the processor clock frequency can be higher. Table 1 shows the 16-bit floating-point multiplier to be a rather complex one. However, the 12-bit multiplier is almost ten-fold smaller. The 16-bit fixed-point multiplier supports a fractional mode which means that the output is already shifted to right scale.

4. RESULTS

We programmed the processors to execute the SSFE detector for 2 × 2 antenna system with 16-QAM. The level update vector is \( \mathbf{m} = [1224] \). The processor architecture has slicer SFUs to accelerate the SSFE detector but otherwise the core includes general purpose FUs and can be programmed to execute other algorithms too. Since the processors have the same function units and they are all synthesized with 160 MHz the comparison of arithmetic benefits is fair.

The latency of the 2 × 2 SSFE algorithm for floating-point processors is 75 clock cycles and 131 clock cycles for the fixed-point processor. The higher latency for the fixed-point processor is a consequence of achieving less optimal instruction schedule with the compiler. However, the same operations are executed in each processor during the algorithm and, therefore, the energy consumptions of alternative implementations are comparable as the consumed energy depends more on the number and type of different operations rather than on the schedule of operations. The floating-point processors provide a decoding rate of 17.1 Mbps at 160 MHz clock frequency.

Table 2 summarizes the number of operations during the algorithm execution. To avoid three clock cycle memory access latency, intermediate results are stored to registers during the execution. The algorithm requires rather high number of multiplications, addition and subtractions. Even though the number of slicer operations is not very high, a single cycle slicer has a significant role in fast algorithm execution.

Table 2. The number of executed operations in the SSFE algorithm

<table>
<thead>
<tr>
<th>Operation</th>
<th># of_ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>84</td>
</tr>
<tr>
<td>SUB</td>
<td>68</td>
</tr>
<tr>
<td>ADD (address gen.)</td>
<td>29</td>
</tr>
<tr>
<td>SUB (address gen.)</td>
<td>2</td>
</tr>
<tr>
<td>MUL</td>
<td>157</td>
</tr>
<tr>
<td>SLICER</td>
<td>28</td>
</tr>
<tr>
<td>RF reads</td>
<td>402</td>
</tr>
<tr>
<td>RF writes</td>
<td>230</td>
</tr>
<tr>
<td>LDW</td>
<td>24</td>
</tr>
<tr>
<td>STW</td>
<td>19</td>
</tr>
</tbody>
</table>

4.1. Processor Complexities

All the processors support 160 MHz clock frequency which provides a fair area comparison between the implementations. The 130 nm CMOS technology sets limits to the clock frequency but it has only a minor effect to our contribution, i.e., comparisons of implementations using alternative number formats. Table 3 summarizes the processor complexities in gate equivalents.

We broke up the significant parts of the TTA processor to show the gate equivalent distribution. The complexities of arithmetic, interconnection network, instruction decoder and instruction fetch are presented in gate equivalents and per cents of the total processor complexity. Arithmetic and interconnection network are the largest part of the processors. The interconnection network takes approximately third of the silicon, which encourages to optimize connections further. There is a large difference in arithmetic percentage between 32- and 12-bit processors. The size of the instruction decoder is a consequence of the multiple parallel function units, which leads to a wide instruction word.

4.2. Energy Dissipation

We summarize the power and energy dissipations in Table 4. For power we have separated cell internal and net switching power dissipations. The cell internal power is consumed when a cell input changes, but there is no change in output. The net switching power is dissipated when charging and discharging the load capacitance at the cell output. The global operating voltage for processors is 1.5 V. The energy dissipation is defined as,

\[
E = Pt,
\]

where \( P \) is power and \( t \) is the latency of the algorithm execution. As expected, the 32-bit floating-point processor has the highest energy dissipation, 32.0 nJ. Interestingly, the 12-bit floating-point processor consumes only 19.5 nJ, which is less than the energy dissipation of the 16-bit fixed-point processor with 23.6 nJ. We also present the energy dissipation per received bit which we can compare toward other implementations in literature. In 16-QAM system, the symbol is represented with four bits. Thus, in 2 × 2 antenna system eight bits are received at the same time.

The difference in energy dissipation between the 12-bit floating-point and 16-bit fixed-point processors is approximately 17 per cent. This observation shows that it is not always granted that a fixed-point implementation suits better for embedded digital systems. It can be expected and it is usually true that, e.g., double precision floating-point implementation does not lend itself for embedded systems due to high silicon area and energy consumption. However, if the analysis and comparisons between implementations using different number systems is extended to cover also non-standard floating point formats like the 12-bit format used in this study, it is possible that implementations using such formats may overcome fixed-point solutions.

5. COMPARISON

The energy dissipation of recent soft-output MIMO detector implementations on ASIC and TTA are compared in Table 5. We give an
overview of four SSFE implementations and a K-best implementation. Due to wide scale of implementations, the target is to compare only the energy per received bit for implementations and platforms. Note that the power dissipation between implementations are not totally comparable due to different CMOS technologies. Scaling a CMOS technology from 180 nm to 65 nm can reduce the design power dissipation up to 75 percent. To scale energy values we use a factor 1.5 between two successive technologies.

In spite of the differences in implementations, the energy dissipation estimates are in line with expectations. The ASIC designs are optimized to execute the particular detector, and, thus, their energy dissipation per received bit is low. The programmability causes 4.0–18 times overhead compared to optimized hardware implementations. However, in larger design the SDR can in general reuse the hardware which likely reduces the energy difference over hardware design.

A high resource utilization for SSFE algorithm execution can be achieved with Texas Instruments TMS320C6416 digital signal processor (DSP) [9]. This provides an interesting comparison to our TTA implementation. We selected the throughput corresponding to a level update vector $\mathbf{r} = [1124]$ which is the closest to the TTA implementation. Note that the DSP implementation is a complex-valued $4 \times 4$ antenna system with 64-QAM. The authors in [9] do not provide an energy dissipation but we did a rough estimate based on the power consumption summary provided by [10]. Compared to the DSP, the TTA implementation provides a significantly better energy per bit dissipation.

### 6. CONCLUSIONS

We studied how a recently proposed soft-output detector compares with the fixed- and floating-point arithmetic. We defined with the link level simulations the required minimum word-lengths for both arithmetics. The arithmetics with a reduced precision were compared to a double precision floating-point result. The silicon of the 12-bit floating-point processor is less than the area of 16-bit fixed-point processor and the processor consumes approximately 17 per cent less energy per detected bit. The results show that it is not always granted that a fixed-point implementation suits better for embedded digital systems. Specially, when the implementations using different number systems are extended to cover also non-standard floating-point formats like the 12-bit format used in this study, it is possible that implementations using such formats may overcome fixed-point solutions.

### 7. REFERENCES


