DOI 10.5162/ettc2018/4.1

Hardware Acceleration for Beamforming Algorithms based on Optimized Hardware-/Software Partitioning

René Schmidt, Stephan Blokzyl, Wolfram Hardt,
Technische Universität Chemnitz, Straße der Nationen 62, D-09111 Chemnitz, Germany,
{rene.schmidt, stephan.blokzyl, wolfram.hardt}@informatik.tu-chemnitz.de

Abstract:

Beamforming techniques are widely used in many fields of research like sonar, radar, wireless communication and speech processing applications. Beamforming algorithms are mainly used for signal enhancement and direction of arrival estimation. In applications for tracking mobile communication partners like speaker or aircrafts, beamforming algorithms are employed to estimate the direction of arrival.

Beamforming processing is always accompanied by high computational costs, which are challenging for embedded devices. Recent approaches process the calculation in frequency domain to reduce the processing time. However, in many cases the processing is still very slow and cannot be used for real-time processing. Alternative real-time solutions based on FPGAs face the drawback of long development processes and restricted communication interfaces.

This paper introduces a novel implementation approach based on System on Chip (SoC) technology with optimized Hardware-/Software Partitioning for real-time delay and sum beamforming. Basis for evaluation is a runtime measurement of software implementations to determine computation steps with high processing time and parallelization capabilities. Extracted computation steps are implemented on an associated FPGA with full pipelined architecture for high data throughput and fast processing speed. The complexity of the deduced architecture is evaluated regarding data length and data width with respect to computational accuracy.

Key words: Beamforming, System on Chip, Hardware-/Software Partitioning, Parallelization

Introduction

Nowadays, beamforming algorithms are used in numerous fields of applications. One field of application is the data generation as it is done in sonar systems to isolate ocean waveguide information [1]. As another example, the automotive industry uses beamforming algorithm for data fusion approaches to merge signals from distinct short range radar systems [2]. However, a main application for beamforming algorithms is signal quality enhancement. On this account, beamforming methods are used in wireless applications [3] and RF designs [4] to enhance the signal quality. Another field of application is spatial filtering focusing on separation of mutually effecting frequencies. In result the signal quality is increased as well [5]. Furthermore, beamforming algorithms are used in speech processing applications to improve speaker signal quality from a certain direction. In result, the signal quality increases and suppressing unintended influencing signals generated by sound sources in different directions like background noises [6].

However, beamforming algorithms can be used in opposite direction. The previous fields of application mainly focused on signal enhancement while another main application of these algorithms is direction of arrival estimation [7]. On basis of beamforming the direction of arrival is identified and used for speaker localization and tracking [8] or determining the position of aircrafts [9]. For this applications steered beamformers are used accompanied by high computational effort challenging embedded devices.

In the following, possible hardware acceleration techniques for beamforming algorithms determining the direction of arrival are analyzed and advantages and disadvantages of the distinct approaches are discussed. Basis for further considerations is the analysis of runtime and calculation complexity of a common delay and sum (DAS) beamformer. For elucidation of the theoretical values an exemplary implementation of the beamformer is used. On this basis, a novel System on Chip (SoC) design will be derived. The novel approach will be described in detail and evaluated with respect to runtime and hardware utilization. Finally, the results are presented and critically discussed in conjunction with a summary of the paper and possible improvements in future works.

Related Work

As previously mentioned, the main problem of beamforming algorithms is the high computational complexity. One possibility to realize a beamforming algorithm for direction of arrival estimation is the calculation in time domain providing simple implementation effort, but reducing the angle accuracy drastically. This raises from the direct proportional dependency of sampling rate F_s and angle resolution represented by following equation:

$$\Delta_i = \left\lceil F_s \frac{i*b\sin(\theta)}{c} \right\rceil \qquad (1)$$

with Δ_i representing amount of delayed samples, intermediated sensor distance b, input angle θ, signal speed c and $i \in \{1,2,..M\}$ where M is the amount of sensors. This implies the increasing of accuracy can just be obtained by increasing the sampling rate. For time domain calculation all input signal are delayed by Δ_i in respect to input angle θ_n with $n \in \{1,2,..N\}$ where N represents the measurement window size. In time domain each sample of the measurement window represent an angle according to equation (1) (comp.Fig.1). Afterwards the delayed signals according to θ_n are summed up. This procedure is repeated for all N. The maximum amplitude of the summed values represents the direction of arrival. Considering the described computation a calculation complexity of $O(N^2)$ can be identified.

Because of the high computational costs accompanied by low accuracy the calculation is transferred to frequency domain. In this context the convolution in time domain is reduced to a multiplication in frequency domain in conjunction with less computational effort. Another benefit is the representation of the time delay Δ as phase shift in frequency domain defined as steering vectors. In consequence an arbitrary angle resolution can be achieved restricted by computational power only. Therefore, the calculation complexity in frequency domain results in $O(N_A \cdot N_{FFT})$ where N_A represents the amount of angles with respect to angle resolution and N_{FFT} represents the amount of frequency components resulting of FFT usage and defining FFT resolution.

The previously discussed arguments are universally valid for beamforming algorithms. However, in last decades numerous versions of beamforming methods have been developed. Each of this algorithms rely on additional non trivial calculations effecting the calculation complexity. One example is Minimum Variance Distortionless Response (MVDR) beamformer depending on an inverse calculation of $M \times M$ matrix [10]. Another example is the Multiple Signal Classification (MUSIC) beamformer depending on calculation of eigenvalues of an equal sized matrix [11].

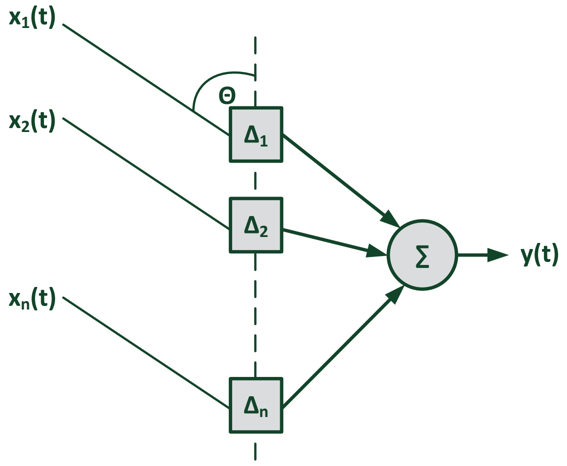

Fig. 1: Standard beamformer with delay Δ and input angle Θ.

Consequently, the transfer in frequency domain alone is not sufficient to solve the computational effort problem. On this account, the algorithms has been realized on specialized Hardware like Field Programmable Gate Arrays (FPGA) [12] or Graphics Processing Units (GPUs) [13] beside powerful PCs.

FPGA based realizations provide a high degree of flexibility with regards to correlated requirements as well as maximum parallelization capabilities. Another big benefit of FPGA usage is the low power consumption leading to a variety of possible applications. On the contrary, FPGA development is accompanied by long development times reasoned by specialized hardware implementation for the specific use case. Furthermore, there is a reduction of usability through additional specialized interfaces for data transfer reducing the throughput of FPGA designs.

On the other hand, GPUs have very high power consumption, but offer high performance and good parallelization characteristics with high throughputs and high processing speed.

In summary, a normal PC is powerful and can handle the specified algorithms in acceptable computing time, but is not suitable for mobile use due to the lack of mobility and high power consumption. FPGAs have low power consumption and high parallelization capabil-ities, but facing the drawback of long development times. GPUs, on the other hand, have high power consumption but offer enormous throughput and acceleration possibilities.

A compromise between power consumption, mobility and hardware acceleration is represented by the SoCs from Xilinx and Altera. The combination of ARM Processor and FPGA provides high flexibility and parallelization capabilities with low power consumption and provides novel possibilities for hardware/ software partitioning due to standardized flexible communication interfaces. For this reason, this architecture provides a suitable basis for accelerating beamforming algorithm for the usage on embedded devises.

For identification of potential task for hardware acceleration, the following chapter provides a detailed description and run time analysis of a standard beam scan algorithm represented by the DAS beamformer.

Algorithmen Analysis

To analyze the most popular representative of the beam scan algorithms, the DAS beamformer, the input signal of the sensors is defined as follows:

$$x_i(t) = g_i(t) * s(t) + n_i(t) \tag{2}$$

Where g represents the spatial response of signal source s with addaptive gausian noise n at time t. The beamformer output can be represented as:

$$y(t) = \omega^H x(t) \tag{3}$$

Where ω = [ω1, ω2,...,ωM] defines the weights of the input channels and H is the Hermitian transpose. Transforming the input signal x_i to frequency domain equation (2) has to be rewritten as:

$$X_i(k) = G_i(k)S(k) + N_i(k) \tag{4}$$

With X_i representing the result of applying a SFFT to the corresponding input signal x_i. Furthermore, S is the signal spectrum of signal source s and noise spectra N. This leads to beamformer output definition:

$$Y_i(k) = W^H(k)X(k) \tag{5}$$

With spatial filter W describing the time delay of equation (1) in frequency domain by a phase shift with respect to input angle θ.

The Delay and Sum Beamformer determines the direction of arrival by successively scanning of the power spectrum. Input angle θ is defined by the maximum magnitude of the power spectrum defined by [14]:

$$P(\omega) = \frac{1}{k}\sum_{k=1}^{K}|X(k)^2| = W^H X(k)X(k)^H W \tag{6}$$

$$= W^H R_{xx} W \tag{7}$$

Where R_{xx} is the well-known covariance matrix of input signal $X(k)$ defined by $R_{xx} = X(k)X(k)^H$. Spatial filter W represents the steering vectors defined by [14]:

$$S(\Theta) = [s(\mu_1(k,\Theta)),\dots,s(\mu_M(k,\Theta))] = \tag{8}$$

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ e^{j\mu(1,\Theta)} & e^{j\mu(2,\Theta)} & \cdots & e^{j\mu(K,\Theta)} \\ \vdots & \vdots & \vdots & \vdots \\ e^{j(M-1)\mu(1,\Theta)} & e^{j(M-1)\mu(2,\Theta)} & \cdots & e^{j(M-1)\mu(K,\Theta)} \end{bmatrix}$$

In this context, term $e^{jq\mu(k,\Theta)}$ represents the phase shift of input signal X with respect to input angle θ, frequency bin k and sensor q. On this basis μ has to be defined as:

$$\mu(k,\Theta) = \frac{-2\pi}{\lambda_k} b\sin(\Theta) \tag{9}$$

Where λ is the wavelength of frequency k. In consequence the direction of arrival is described by the maximum amplitude of equation (7):

$$argmax\{P(\Theta)\} = argmax\{S(\Theta)^H R_{XX} S(\Theta)\} \tag{10}$$

The described algorithm results in the process flow depict in Fig. 2.

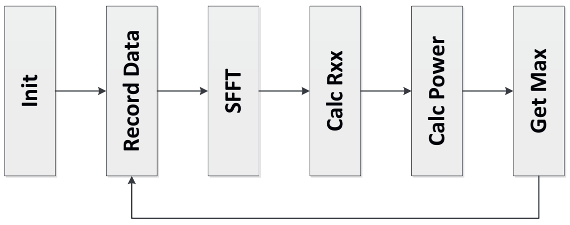

Fig. 2: Process flow for Beamforming computation.

At begin of the process all constants are determined including the steering vectors, since they just depend on angle granularity and FFT resolution. The “Init” process will be ignored in the further analysis, since the initiation costs just appear ones and effect the startup of the system only. The second step is the acquisition of data consisting of data sampling and copying in a predefined buffer. The runtime has complexity of $O(MN_{FFT})$ because every sample of the measurement window has to be copied ones for each input channel. Therefore, N_{FFT} is the length of measurement window defining FFT length and FFT resolution respectively. The following block is SFFT describing short-time Fourier transform of length N_{FFT}. If $N_{FFT} = 2^p$ with $p \in \mathbb{N}$ it can be shown that the complexity of SFFT is $O(N_{FFT}\log(N_{FFT}))$ and the accumulation for all channels results in $O(MN_{FFT}\log(N_{FFT}))$.

Afterwards, the calculation of the $M \times M$ matrix $R_{XX} \in \mathbb{C}$ takes part. According to definition of $R_{xx} = X(k)X(k)^H$, M² complex values has to be determined. For the determination of one component of matrix R_{XX} M complex multiplications and M-1 additions has to be executed leading to a runtime complexity of $O(M^4)$ for one frequency component. The process has to be repeated for each frequency bin successively conducting in runtime complexity of $O(\frac{N_{FFT}}{2}M^4)$. According to the master theorem, if $M \ll N_{FFT}$ the complexity for R_{xx} determination can be estimated by $O(N_{FFT})$.

The final power calculation from equation (7) is realized in the subsequent “Calc. Power” block. Therefore, the steering vectors given by equation (8) are multiplied with the determined matrix R_{xx} to archive the power level corresponding to input angle θ. The computation of equation (7) has to be repeated for each possible input angle and for each frequency component resulting in runtime complexity of $O\left(\frac{N_{FFT}}{2}N_A\right) \cdot O(W^H R_{xx} W)$. The computational costs for $W^H R_{xx}$ can be estimated by $O(M^3)$. The result of this matrix multiplication is a vector of length M which has to be multiplied by the steering vector as well. In consequent, the vector multiplication has to be estimated by $O(M^2)$. Subsequently, the absolute value of the result has to be determined. The calculation is accompanied by two multiplications for square computation, one addition and one square root determination resulting in $O(3n_b \log n_b{}^{2O(log*n_b)})$ with n_b defining number of bits [15]. In consequence the total runtime complexity is $O(\frac{N_{FFT}}{2}N_A) \cdot (O(M^3 + M^2 + 3n_b \log n_b{}^{2O(log*n_b)})$. Following the master theorem the complexity can be reduced to $O(N_{FFT}N_A)$.The last task is the maximum detection with a trivial runtime of $O(N_A)$.

For clarification of the theoretical time estimation, the system has been implemented on a Cortex A9 ARM Processor with the purpose to measure the time for each described task. For evaluation a FFT resolution of 1024, an angle granularity of one degree and three input sensors have been chosen. The results are depict in Tab.1.

Tab. 1: Results of runtime measurement for DAS with 3 Sensoren, N_{FFT} 1024 und N_A 180 implemented on ARM Cortex A9 Prozessor.

Task	Complexity	Time (ms)
Record Data	$O(MN_{FFT})$	1,313
SFFT	$O(N_{FFT}\log(N_{FFT}))$	1,668
Calc Rxx	$O(\frac{N_{FFT}}{2}M^4)$	5,124
Calc Power	$O(N_{FFT}N_A)$	533,181

The results show that the runtime of $O(N_{FFT}N_A)$ in combination with the expensive constant costs for square root calculation effecting the overall runtime massively. In consequence the theoretical consideration and the exemplary time measurement identifying the power calculation as bottleneck in the localization process.

The following chapter describes a novel approach for hardware acceleration bases on SoC Architecture with the aim of dissolving the bottleneck.

System on Chip Architecture

To handle the identified bottleneck a Xilinx Zynq SoC is used. The SoC combines a Cortex A9 ARM Processor and FPGA with the standardized AXI-Interface as communication medium. The Aim of the design is to use the parallelization capabilities of the FPGA to dissolve the bottleneck and improve the throughput of the power calculation but also keep the most possible flexibility to compensate the FPGA development costs.

A localization system bases on beamforming algorithms usually depends on the amount of sensors M, the scanning region defined by angle range, the granularity of the angles resulting in number of steering vectors N_A and length of measurement window N_{FFT} given by FFT resolution. To ensure the greatest possible flexibility and reusability of the FPGA design, the realization should be independent from this parameters as fare as possible.

By considering equation (7) and the corresponding runtime complexity it is clear that all this parameters influencing the calculation. The amount of angles and angle granularity are indirectly given by the amount of steering vectors. The steering vectors depend on the frequency components and possible input angles only whereby a calculation in the initialization phase is sufficient. On the other hand, the calculation of R_{xx} has to be adapted with each measurement window. Nevertheless, the amount of steering vectors has a huge memory complexity since there is one complex steering vector for each frequency component, input angle and sensor. This leads to a memory usage of $O(\frac{N_{FFT}}{2}N_A M)$ which represent a big challenge for FPGAs and the limited Block RAM resources to store all steering vectors initially. For this reason, the steering vectors has to be transferred from the DDR RAM separately.

On this account, the task of the process flow depict in Fig.2 are mapped to the software part until task “Calc Rxx”, the other tasks are mapped to the hardware part connected by DMA controller and AXI-Lite interface. The resulting system architecture is show in Fig.3. The maximum detection has been moved to the hardware part to reduce communication costs. After power calculation N_A values have been generated and have to be transferred to the software by a DMA Controller with subsequently maximum detection. However, usually just the direction of arrival is needed which is represented by one angle with maximum magnitude. Based on this just one angle need to be transferred if the maximum detection is realized on hardware.

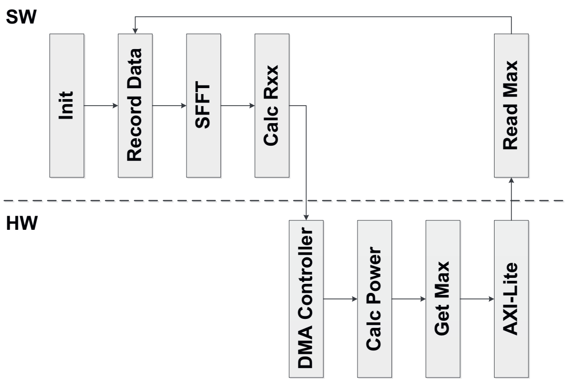

Fig. 3: System architecture with HW-/SW-Partitioning.

Since a huge amount of data has to be transferred between processor and FPGA a DMA controller is used to transfer all necessary data from DDR RAM to FPGA by direct memory access sequentially. Reversely, an AXI-Lite interface is used, since it provides less hardware utilization and simple usability for transferring low amount of data. After finishing the calculation process, the angel indicating the direction of arrival has to be transferred only, reasoning the usage of AXI-Lite interface. Additionally the AXI-Lite interface acts as configuration interface as well.

Reviewing equation (7) in detail it is evident, that the matrix multiplication can be divided in three parallel vector multiplications. This vector multiplications can be efficiently implemented with shift registers. The following vector multiplication proceeds in parallel with one clock cycle delay (see Fig.4).

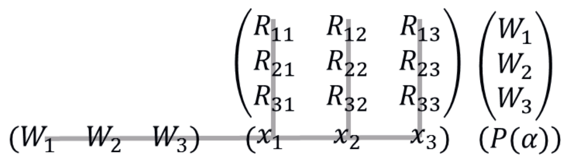
$\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}$
$\begin{pmatrix} W_1 \\ W_2 \\ W_3 \end{pmatrix}$
$(W_1 \quad W_2 \quad W_3)$
$(x_1 \quad x_2 \quad x_3)$
$(P(\alpha))$

Fig. 4: Schematic power calculation of equation (7) with parallel vector multiplication (gray) for M=3.

This calculation has to be carried out N_A times without changes of the R_{xx} values. For this reason, the values for R_{xx} are transmitted once for each frequency component to subsequently stream the steering vectors through the calculation architecture. This approach has been realized in the following FPGA design depict in Fig. 5.

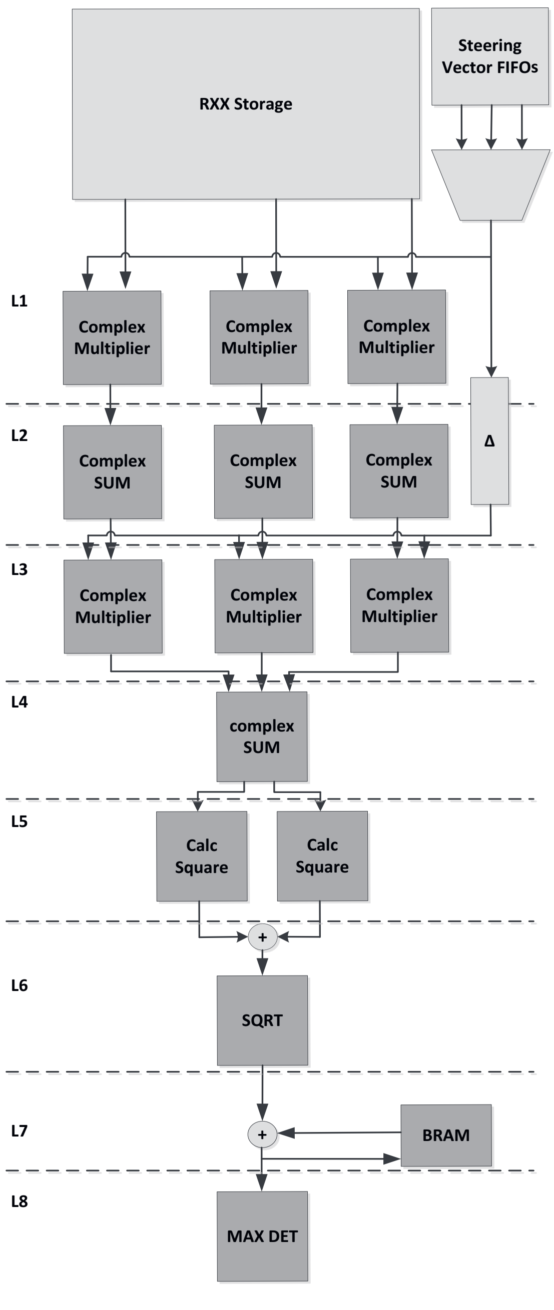

Fig. 5: Block diagram of FPGA Design for parallel computation of equation (7) with maximum detection for M=3. Calculation instances are displayed dark gray routing and storing instances light gray divided in computation level L1 to L8.

The data are transmitted to the FPGA by a simple streaming protocol, whereas the first M² data are identified as R_{xx} entries. The R_{xx} entries are stored in RXX Storage. All other incoming data are steering vectors which are stored in M separated FIFOs. The FIFOs representing one row of the matrix depict in equation (8). For communication reduction the first row is not transmitted since it is constant one. The DMA Controller supports 32bit and 64 bit interfaces. For this implementation, 64bits are used which is logical divided in two 32bit values. The two values representing the real and imaginary part of one complex number. For this reason an arbitrary amount of sensors can be supported since they can be identified by a simple modulo calculation.

After the R_{xx} values and the steering vectors have been transferred, the data according to formula (7) and the schematically represented calculation in Fig.4 are transferred into the calculation architecture by a multiplexer.

The first level of the calculation architecture (L1) is formed by M complex multipliers, which as the name implies, realizes a multiplication of complex numbers. To reduce the number of DSP cores, which are used to perform the multiplications, the calculation is realized as follows:

$$\mathrm{z1} = (\mathrm{a} + \mathrm{bi}) \quad \mathrm{z2} = (\mathrm{c} + \mathrm{di})$$

$$R(z1 \cdot z2) = ac - bd \tag{11}$$

$$I(z1 \cdot z2) = (a + b)(c + d) - ac - bd \tag{12}$$

The advantage over the traditional variant is the reduction of four multiplications to three multiplications leading to a reduction of M DSP cores per calculation level. The resulting calculation tree is depict in Fig.6.

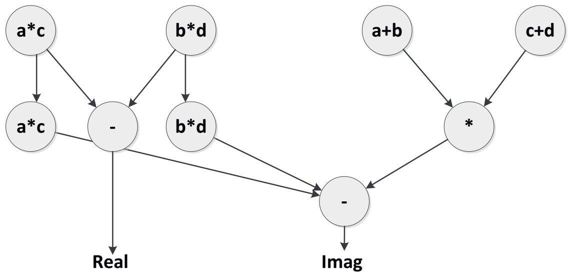

Fig. 6: computation flow for complex multiplication for $z1 \cdot z2$.

The realization requires three clocks for one complex multiplication until the result is obtained, but offers a full pipelined structure with maximum throughput.

The multiplexer passes the corresponding matrix components from Fig. 4 one after the other to the complex multipliers. The multiplication results are transferred to level 2 (L2). In L2 the summation of the intermediate results takes place ending the parallel matrix calculation shown in Fig.4. The result of L2 is the row vector (x_1, x_2, x_3) in Fig. 4. In L3 the multiplication of row vector x and steering vector takes place. For this reason the incoming steering vectors are delayed in such way so that they are present at the appropriate time at L3. The resulting summands are summed in level 4 leading to the sum representing the complex value of P(Θ). In L5 the square of the real and imaginary part transferred from L4 is calculated summed and forwarded to L6 for square root determination. For square root computation the Cordic-IP core is used, provided by the vendor. Result of L6 is the power value of the dedicated frequency component at input angle Θ. Since the results must be added over all frequency components, each value is stored in a BRAM. In each subsequent iteration the values are read and summed up in parallel. For this purpose, the number of angles N_A must be transferred once during the initialization phase via AXI-Lite interface. In the last step, the maximum is determined by storing the current maximum and the associated angle index for each iteration, which can be accessed via the AXI-Lite interface after completion of the calculation.

To determine the latency of the system, the respective latencies of the described blocks have to be added for each processing level. The multiplication in L1 and L3 requires three clock cycles while the summation in L2 and L4 takes one clock cycle only. Squaring the real and imaginary part in L5 is calculated in one clock cycle the subsequent summation requires one additional cycle. The calculation of the square root is more complex and requires 13 clock cycles until the result is available. The summation in L7 and the determination of the maximum in L8 are processed in one cycle respectively. In summery the latency of the design results in 25 clock cycles. The determination of the total runtime is given by the behavioral description of the system. To determine one power value, N_A steering vectors must be shifted through the calculation architecture. All N_A steering vector have to be active for M clock cycles since all M components of matrix R_{xx} have to be shifted through the complex multipliers. This results in a runtime of $N_A M + 25$ clock cycles for one maximum determination of a frequency component. The process has to be repeated $\frac{N_{FFT}}{2}$ times for each frequency component. The total runtime results in $\frac{N_{FFT}}{2}(N_A M + 25 + T_{com})$ with T_{com} defining the communication time for transferring the steering vectors from DDR RAM to FPGA. T_{com} can be estimated by N_A clock cycles, but includes additional communication costs for DMA controller processing leading to inaccuracies in the estimated total runtime. The conversion from clock cycles to time units depends on the individual configured FPGA base frequency.

The correctness of the design has been verified by simultaneous simulations in Matlab and Vivado simulator.

Results

For evaluation purpose of the design architecture, the design has been implemented on a Zynq ZedBoard according to Fig.3. Correspondingly, the data acquisition, the determination of the SFFT and the calculation of the R_{xx} matrix were implemented in software. On hardware the power computation and maximum detection have been realized with FPGA base frequency of 100 MHz.

Reviewing, the FPGA design from Fig. 5 is independent of the number of steering vectors, the definition of the steering vectors and the resolution of the FFT. However, the design architecture still depends on amount of sensors M since each column in Fig.5, at level 1 to 3, represents one Sensor. For the usage of M sensors M complex multiplier and M summations have to be implemented in L1 to L3 influencing the resource utilization massively. The following figure depicts the used Look Up Tables (LUT) and Flip Flops (FF) in respect to the amount of sensors.

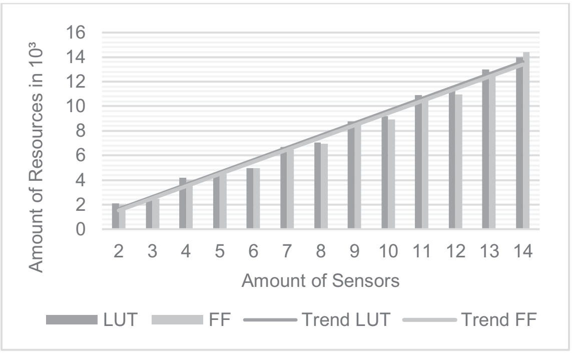

Fig. 7: Resource utilization of FFs and LUTs in dependency of used Sensors and linear trend (light and dark gray lines).

It can be observed that a linear correspondence between resources and amount of sensors has been achieved. This can be explained by the linear relationship between the number of sensors and the required complex multiplier and summations. An additional observation is the constant relation between amount of LUTs and FFs, furthermore the amount of resources is almost equal.

For the realization of the multipliers DSP cores are used. The usage of the DSP cores depending on amount of sensors is shown in Figure 8. For the same reasons, a linear trend is identified for DSP usage. However, the number of DSP cores is twice as high as instantiated multipliers reasoned by DSP core input vector width. The calculation bases on the multiplication of 32 bit values while the DSP cores supporting a 25bit multiplication only. During synthesis, this is resolved by using 2 DSP cores reasoning the higher utilization.

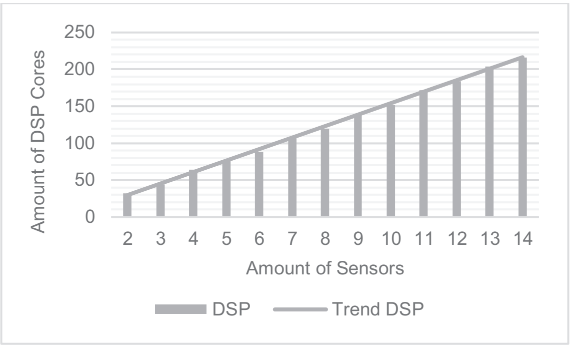

Fig. 8: Amount of used DSP Cores in dependency of used Sensors and linear Trend.

The parameters N_A and N_{FFT} have no influence on the hardware resources, but determining the overall processing time mainly. As derived in the previous chapter, a linear dependency of the runtime $\frac{N_{FFT}}{2}(N_A M + 25 + T_{com})$ is expected. To evaluate the processing time, the design from Fig. 5 was used to proceed varying amounts of data, to measure the necessary runtime. The procedure was repeated 500 times to ensure the statistical significance. The data variability is the result of the variation of the parameters N_A and N_{FFT}. The measurement results are illustrated in Figure 9 confirming the linear dependence on amount of steering vectors N_A and processing time. Figure 10 shows the measurement results for different FFT resolutions. The linear relationship between amount of frequency components and processing time can be confirmed by the measurements as well.

The number of sensors have a linear influence on the processing time of the calculation as well since it defines the size of matrix R_{xx}. However, the high influence of (M^3+M^2) of the sensor count has been cancelled out by the parallelization of the matrix calculation.

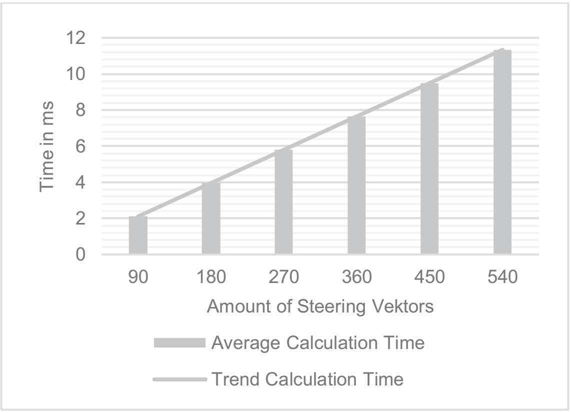

Fig. 9: Dependency between angle resolution and average processing time with trend line.

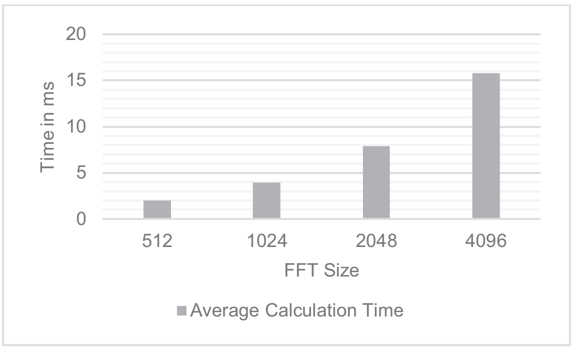

Fig. 10: Dependency between processing time and FFT resolution.

Even in a numerical comparison of runtimes, the design has significantly accelerated the processing time to the pure software solution. Compared to the software solution in Table 1, the processing time has been reduced from ~500 ms to ~4 ms. Even at four times higher resolution a processing time of ~15ms was achieved which is much faster than a standard embedded processor. For illustration, the used ARM processor has a frequency of 700 MHz and needs approximately 500 ms for calculation. This is equivalent to approx. 350 million clock cycles, to achieve the same processing time of 4 ms a processor with a frequency of 87,5 GHz has to be used. The explanation can be derived from the comparison of runtime estimations. For Software solution a runtime of $O(\frac{N_{FFT}}{2}N_A) \cdot O(M^3 + M^2 + T_{root})$ has been identified on the other hand a runtime of $O(\frac{N_{FFT}}{2}(N_A M + 25 + T_{com}))$ for hardware solution have been achieved. Comparing both results it is clear that the big factor of $O(M^3 + M^2 + T_{root})$ has been reduced to $O(M)$ reasoned by the parallelization of matrix calculation. Furthermore the calculation time of square root estimation has been canceled out reasoned by pipelining the root determination.

In summary, the presented design is flexible in terms of angular granularity, scan region and FFT resolution. Only a change in the number of sensors has an effect on the FPGA design. Furthermore, the processing time scales linear to all depending parameters.

Conclusion and Future Work

This paper presented a novel approach to hardware acceleration of beamscan algorithms for localization of signal sources. The algorithm class was analyzed and calculation bottlenecks identified. Based on the analysis, a Hardware-Software Co-Design design was derived resolving the bottleneck. The design works in the frequency domain and is independent of the FFT resolution, the angle resolution, as well as the scan region. Only changes in sensor count leads to an adaption of the FPGA design. By the novel integrating of the FPGA into the localization system, the complexity of the system could be reduced leading to a linear scalability of all hardware resources and total runtime.
Future work will deal with further reduction of DSP cores and optimization techniques to reduce the hardware resources.

Acknowledgment

The authors gratefully acknowledge funding by the DFG (GRK 1780/1).

References

[1] G.E. Allen, B.L. Evans, Real-time sonar beamforming on workstations using process networks and POSIX threads, *IEEE Transactions on Signal Processing 48.3,* 921-926 (2000); doi: 10.1109/78.824694

[2] K. Schuler, et al, Array design for automotive digital beamforming radar system, *Radar Conference, 2005 IEEE International. IEEE*, 2005; doi: 10.1109/RADAR.2005.1435864

[3] J. Litva, and T. K. Lo, Digital beamforming in wireless communications. *Artech House, Inc.,* 1996. ISBN:0890067120

[4] R. Kohno, Spatial and temporal communication theory using adaptive antenna array, *IEEE personal communications 5.1*: 28-35 (1998); doi: 10.1109/98.656157

[5] B. D. Van Veen, K. M. Buckley, Beamforming: A versatile approach to spatial filtering, *IEEE assp magazine 5.2*,4-24 (1988); doi: 10.1109/53.665

[6] J.-M. Valin and et al., Robust sound source localization using a microphone array on a mobile robot, *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol. 2*. IEEE, 1228–1233 (2003); doi: 10.1109/IROS.2003.1248813

[7] V. Krishnaveni, T. Kesavamurthy, and B. Aparna, Beamforming for direction-of-arrival (doa) estimation-a survey, *International Journal of Computer Applications, vol. 61, no. 11*, 2013; doi: 10.5120/9970-4758

[8] I. Markovi´c, et al., Speaker localization and tracking with a microphone array on a mobile robot using von Mises distribution and particle filtering, *Robotics and Autonomous Systems, vol. 58, no. 11*, pp. 1185–1196 (2010); doi: /10.1016/j.robot.2010.08.001

[9] U. Michel, History of acoustic beamforming, *1st. Berlin Beamforming Conference*. 2006.

[10] J. Capon, High-resolution frequency-wavenumber spectrum analysis, *Proceedings of the IEEE,* 1408-1418 (1969); doi: 10.1109/PROC.1969.7278

[11] R. Schmidt, Multiple emitter location and signal parameter estimation, *IEEE transactions on antennas and propagation*, 276-280 (1986); doi: 10.1109/TAP.1986.1143830

[12] A. Ahmedsaid, A. Amira, and A. Bouridane, Accelerating MUSIC method on reconfigurable hardware for source localization, *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on. Vol. 3,* IEEE, 2004; doi: 10.1109/ISCAS.2004.1328760

[13] V. P. Minotto, et al., GPU-based approaches for real-time sound source localization using the SRP-PHAT algorithm, *The International Journal of High Performance Computing Applications,*291-306 (2013); doi: 10.1177/1094342012452166

[14] S. N. Bhuiya, F. Islam, M. A. Matin, Analysis of Direction of arrival techniques using uniform linear array, *International Journal of Computer Theory and Engineering*, 931 (2012).

[15] M. Fürer. Faster integer multiplication. *SIAM Journal on Computing*, 979-1005 (2009); doi: 10.1137/070711761