Project Description

CSSE4010 - Digital System Design
School of ITEE, University of Queensland, Semester 2 2022
Due date: Monday, 7 November 2022 10AM AEST

Custom Computing Hardware Architectures for a Digital Multi-Beamformer

Introduction

This project explores the use of approximate algorithms and approximate hardware in designing low-complexity hardware accelerators with an example application in antenna array based digital beamforming. Beamforming refers to selective filtering of propagating radio or acoustic waves based on their direction of arrival (DOA) on an array of sensors. When multiple signals are impinging on an array of antennas/sensors, digital signal processing (DSP) can be employed to "spatially" filter one or many signals with a desired DOA, while rejecting other interfering (i.e. jamming) signals with undesired DOAs and noise. Thus, the underlying DSP system essentially produces one or many "beams", which are directional sensitivity patterns that can be electronically steered by changing the parameters of the DSP algorithm. A system producing multiple such beams simultaneously is known as a multi-beamformer, which can be used to selectively filter different signals arriving at multiple directions (see Fig. 1(a)).

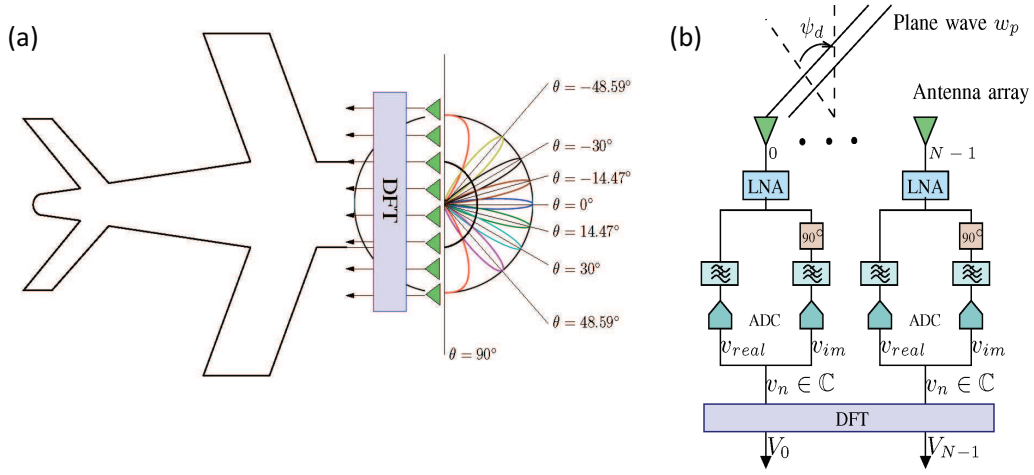

Figure 1: (a) An antenna array based receiver producing multiple beams. (b) Overview of the multi-beam receiver with antenna elements, RF front-end and DSP back-end. Images are taken from [1] without permission.

As shown in Fig. 1(b), a digital beamformer typically consists of an array of antenna elements, a front-end consisting of signal conditioning and analog to digital conversion and a DSP back-end implementing the beamforming algorithm. This project focuses on developing FPGA based hardware architecture for the DSP back-end. The particular DSP algorithm considered in this project is a *spatial* discrete Fourier transform (DFT) operation (i.e. a DFT performed along the spatial dimension). As shown in Fig. 1(b), we consider an antenna array having 8 elements (i.e. $N = 8$), where signal at each antenna is converted to digital following a down-conversion process. Therefore, at a given time index n_2, the input to the DSP algorithm is a frame of 8 complex valued signal amplitudes.

The main goal of this project is to come up with a suitable hardware architecture for this DSP back-end (i.e. the spatial DFT operation) using a combined approach of approximate algorithms and approximate hardware, as described next.

Task Description

1. Play with the given Matlab code, which generates a test input signal for the 8-element antenna array and implements the spatial DFT multi-beamformer using the Matlab FFT function. The given Matlab script obtains the multi-beam pattern of the beamformer by sweeping the input signal angle from 0-180 degrees. The software beam patterns (from the Matlab FFT function) are plotted both in Cartesian and polar forms to show the 8 beams produced by the algorithm. This base code will be explained in tutorial class on week 12.

   Note that the multiplier complexity of ideal N-point DFT is N^2 and that of the fast Fourier transform (FFT) is $N\log(N)$. However, the computational complexity can be further reduced by employing an approximate algorithm for DFT which reduces the multiplier complexity to 0, while providing acceptable accuracy in the beam patterns. Such an approximate-DFT (ADFT) algorithm has been proposed in references [1, 2], which have been shared on the Blackboard site.

   Your first task is to replace the ideal DFT/FFT operation in Matlab with one of the ADFT algorithms proposed in [1, 2] and verify the operation of ADFT-based multi-beamformer in software. i.e., obtain the beam patterns from the ADFT algorithm in software and visually compare them with the FFT beam patterns to verify the correct operation of the ADFT beamformer. **(2 marks)**

2. Implement the ADFT algorithm in hardware using Xilinx/AMD Model Composer tool flow using basic FPGA blocks such as adders, delays, and shifters, but with **no multipliers**. An example block diagram of the ADFT algorithm can be found in [1, Fig.10]. Note that the real and imaginary parts of the complex valued input is handled separately in [1, Fig.10] and you can follow a similar approach.

   (a) Verify the functionality of your hardware design for ADFT by visually comparing the beam patterns with those obtained from the software ADFT and the software FFT in step (1). Initially, for this step, your hardware design can be un-optimised with no pipelining and with arbitrarily large word lengths. Provide sufficient evidence for functional verification of this un-optimised hardware design. **(5 marks)**

   (b) Fully pipeline your design, show your work on how you pipelined the design and indicate the critical path before after pipelining. **(2 marks)**

   (c) Allocate a reasonable fixed-point precision to computations within your design and provide quantitative justification for your word length selection. You can use 8 bits as the input ADC word length. **(1 mark)**

   (d) For the optimised design after step 2-(c) (i.e. after pipelining and word length selection) provide a detailed functional simulation for the optimised hardware design by completing the table below. Also, provide a cartesian and polar plot showing 8 beams from the optimised ADFT hardware design. **(2 marks)**

Beam number k	Main beam direction (deg)		
	FFT Software	ADFT Software	ADFT Hardware Optimised
1			
2			
3			
4			
5			
6			
7			
8			

   (e) Synthesise and implement your optimised hardware design for ADFT and report FPGA resource consumption, critical path delay and the maximum possible frequency of operation. **(2 marks)**

3. The ADFT architecture having 0 multiplier complexity in (2) can be further optimised for complexity by replacing the accurate adders (i.e. Xilinx/AMD adder blocks) with approximate adders. Perform a literature search to find approximate adder architectures and select an approximate adder of your choice which can be described in VHDL. Some examples of approximate adders are provided in [3] and references therein. Implement your choice of approximate adder in VHDL using the required word length from part (2) above, and verify its operation in Vivado. You can use any level of abstraction to describe the approximate adder in VHDL. One possible way is to employ a data-flow description for an approximate full-adder block and then employ a structural description to create an N-bit parallel adder (ripple or carry-look ahead type). You are welcome to look at any open source VHDL descriptions for approximate adders from literature, however, the VHDL description you produce must be your own, with the appropriate word length you have selected in part (2). Any reference designs you have consulted should be properly cited in your report. Import your N-bit approximate adder as a blackbox to Model Composer [4] and use it to replace the accurate adders in your design from step (2). Simulate the resulting ADFT+approximate adder hardware design and examine its performance via beam pattern plots and by adding another column to the table in step (2)-(d). Report FPGA resource consumption, critical path delay and the maximum possible frequency of operation for this design. **(5 marks)**

4. Provide a short discussion and conclusion about your findings. **(1 mark)**

Submission

You are required to submit the following items as two separate submissions on Blackboard by the due date **(Monday 7 Nov 2022 10:00AM AEST)**:

1. A zip file containing your Matlab script file, properly named Xilinx Model Composer Simulink designs files, and VHDL source file for approximate adder. There is no need to include any of the files generated after synthesis and implementation - this will be a Blackboard submission

2. An electronically typeset PDF report containing the items outlined in task description above - this will be a Turnitin submission

Late penalties apply as per the ECP unless you have an approved extension. This project is worth 20% of the final course marks.

References

[1] S. Kulasekera, A. Madanayake, D. Suarez, R. J. Cintra, and F. M. Bayer, "Multi-beam receiver apertures using multiplierless 8-point approximate DFT," in *IEEE Radar Conference (RadarCon)*, 2015, pp. 1244–1249.

[2] V. A. Coutinho, V. Ariyarathna, D. F. G. Coelho, R. J. Cintral, and A. Madanayake, "An 8-beam 2.4 GHz digital array receiver based on a fast multiplierless spatial DFT approximation," in *2018 IEEE/MTT-S International Microwave Symposium - IMS*, 2018, pp. 1538–1541.

[3] B. S. Prabakaran, S. Rehman, M. A. Hanif, S. Ullah, G. Mazaheri, A. Kumar, and M. Shafique, "Demas: An efficient design methodology for building approximate adders for FPGA-based systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 917–920.

[4] "Xilinx/AMD Guides - Vitis Model Composer Blackbox Module," https://docs.xilinx.com/r/2021.2-English/ug1483-model-composer-sys-gen-user-guide/Black-Box.