RECRLab
Reconfigurable Computing Research Laboratory

A Reconfigurable Fixed-Point Architecture for Adaptive Beamforming

DANIEL LLAMOCCA AND DANIEL ALOI

Electrical and Computer Engineering Department,

Oakland University

May 23rd, 2016

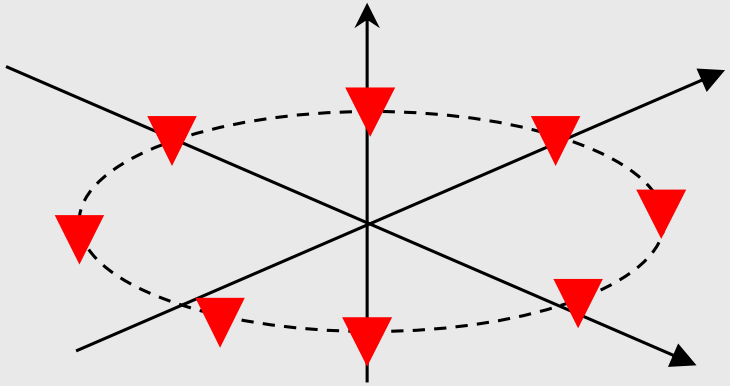
OAKLAND UNIVERSITY

Outline

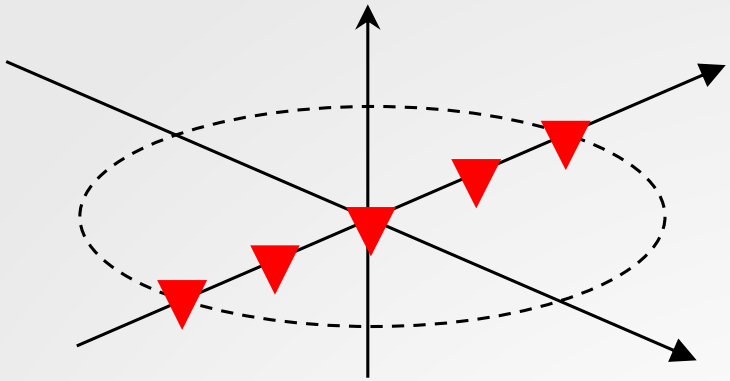
- Motivation
- Adaptive Beamformer
  - Algorithm
  - Architecture
- Experimental Setup
- Results
- Conclusions

Motivation

- *Adaptive beamformers are common in switched-beam **Smart Antennas** that contain fixed beam patterns that are switched on-demand. They can also be used in null steering.*

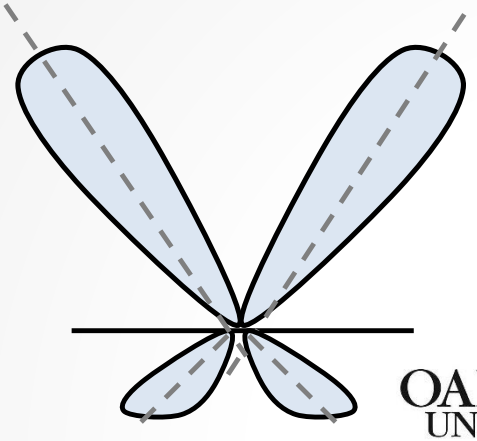


UNIFORM CIRCULAR ARRAY



UNIFORM LINEAR ARRAY

- *The adaptive nature of the beamformers makes them a suitable candidate for reconfigurable implementation.*
- *Beamformers are widely used in radar, sonar, speech, and mobile/wireless communications*



Adaptive Beamformer

- ***Description***:
  - *System that receives a vector of signals $\vec{y}_n$ from multiple antenna elements.*
  - *The Beamformer output is described by: $z[n] = \vec{w}_n^H \times \vec{y}_n$.*
  - *We want to emphasize a signal from a desired Direction of Arrival (DOA) and suppress undesired signals. This is accomplished by adaptively adjusting the weights $\vec{w}_n^H$.*

- *This requires large amounts of data to be processed in parallel. In addition, we need resource efficiency. Here, dedicated **fixed-point hardware implementations** are desired.*

- *We present a reconfigurable beamforming architecture validated on a Programmable System-on-Chip.*

- *We analyze trade-offs among resources, accuracy, and hardware parameters.*

Adaptive Beamformer

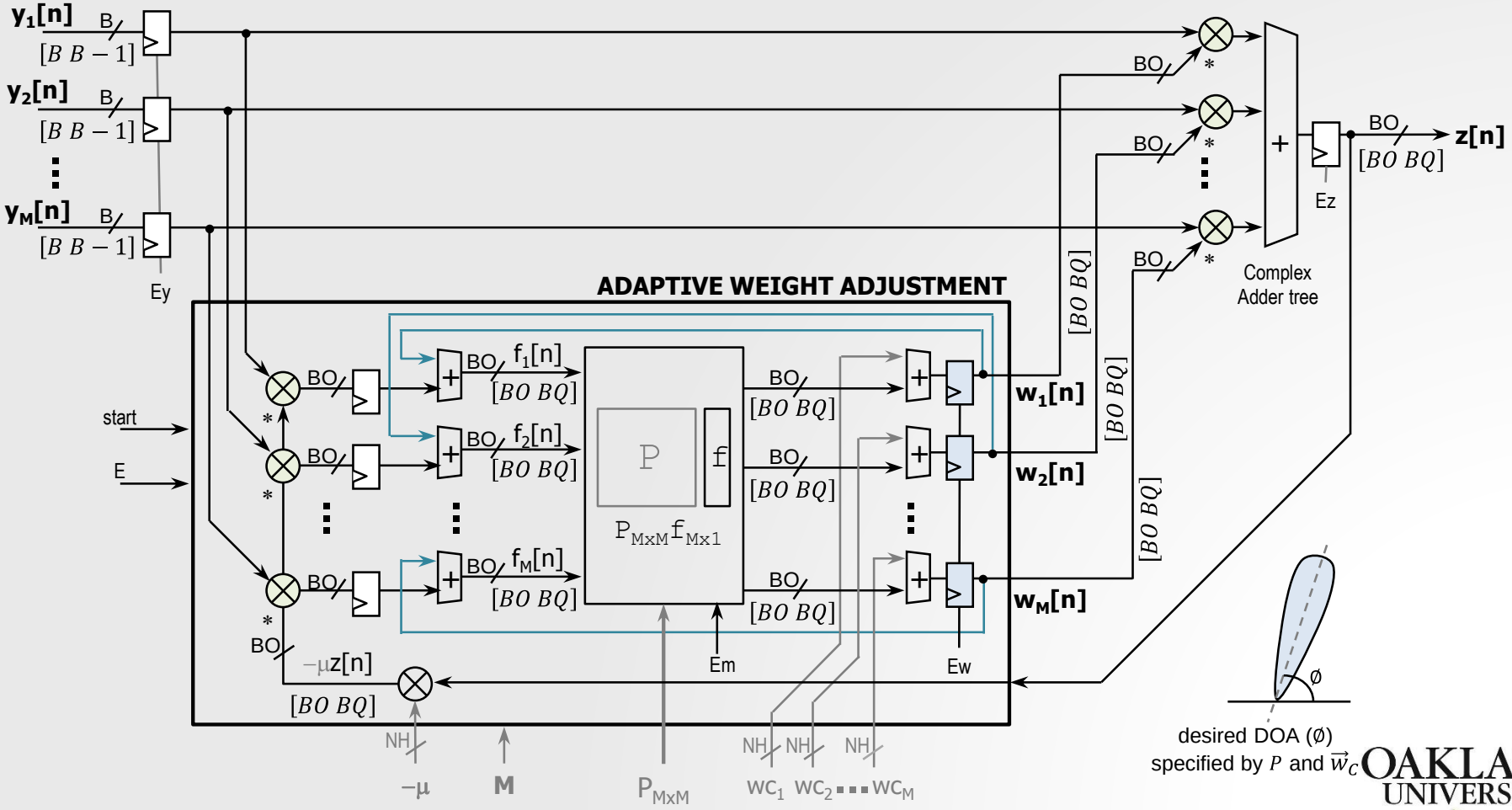
- ***Frost's Adaptive Algorithm***
  - *$\vec{y}_n$: column vector of M complex input samples at time n. M: number of sensors. N: number of **snapshots** (collection of M samples at time n).*
  - *$\vec{w}_n$: column vector of M complex weights at time n ($\vec{w}_1$=$\vec{w}_C$).*

$$
\begin{aligned}
&for\ n = 1{:}N \\
&\quad z[n] = \vec{w}_n^H \times \vec{y}_n \\
&\quad \vec{w}_{n+1} = \vec{w}_C + P \times (\vec{w}_n - \mu z^*[n]\vec{y}_n) \\
&end \\
&P = I - C^H(CC^H)^{-1}C,\ \vec{w}_C = C^H(CC^H)^{-1}c
\end{aligned}
$$

  - *Constrained optimization problem subject to $C.\vec{w}_n = c$. C: constraint matrix. c: column of constraining values.*
  - *$\vec{w}_n$ accentuates signals coming from some direction and/or suppress jammers. Thus, C and c steer the beamformer in a desired direction (i.e., switch to a desired beam pattern).*
  - *This is a relatively simple yet numerically robust algorithm. We steer the beamformer by updating the constraints C and c.*

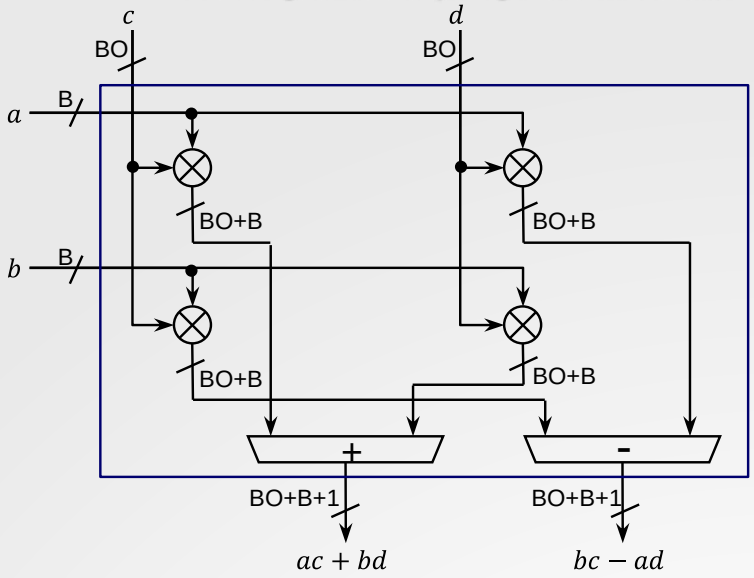
Adaptive Beamformer

- ***Parameterized fixed-point Architecture***
  - *Complex data: Input Format: [B B-1]. Output format [BO BQ]*
  - *$P, \vec{w}_C$: They steer the beamformer in a particular direction*

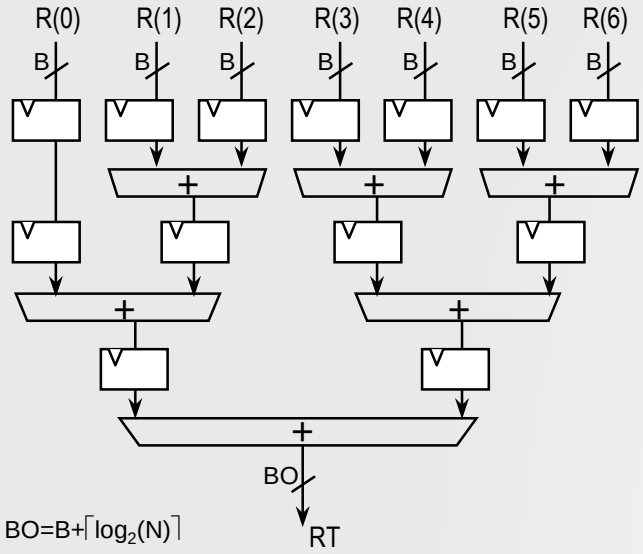


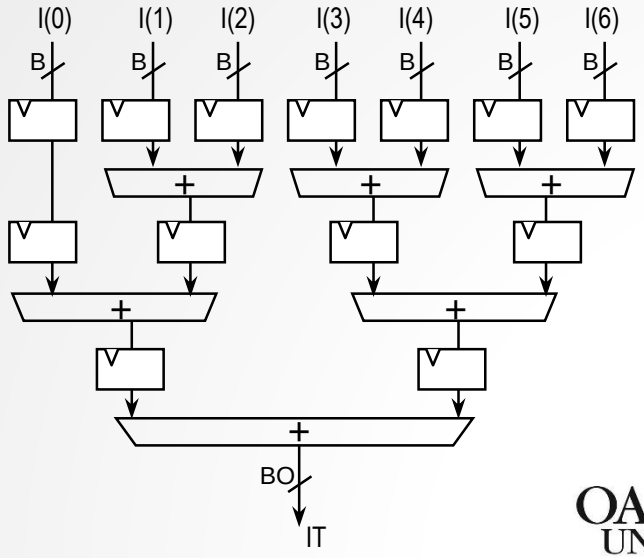
Adaptive Beamformer

- ***Components:***
  - *2M Complex Adders*
  - *2M Complex Multipliers: Each one requires 4 multipliers and two adders.*



- *Complex Adder Tree: It requires 2 M-input pipelined adder trees.*



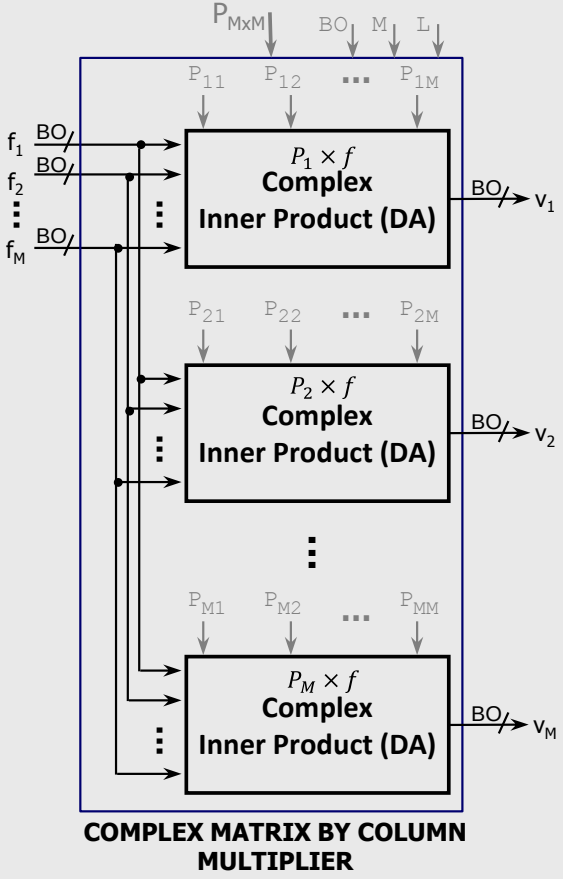


Adaptive Beamformer

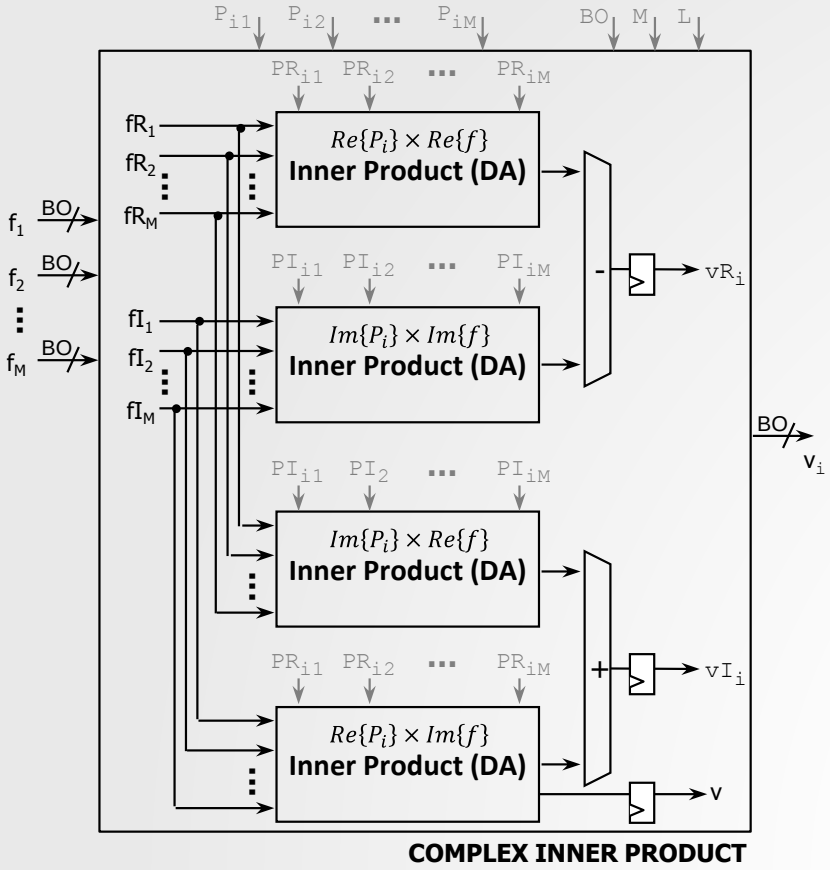
- ***Components:***
  - *Complex Constant Matrix by Column Product:*

$$P_{M\times M} \times f_{M\times 1}, f = \vec{w}_n - \mu z^*[n]\vec{y}_n$$

M inner products: For a particular Direction of Arrival, P is constant. We use Distributed Arithmetic to avoid multipliers



COMPLEX MATRIX BY COLUMN MULTIPLIER



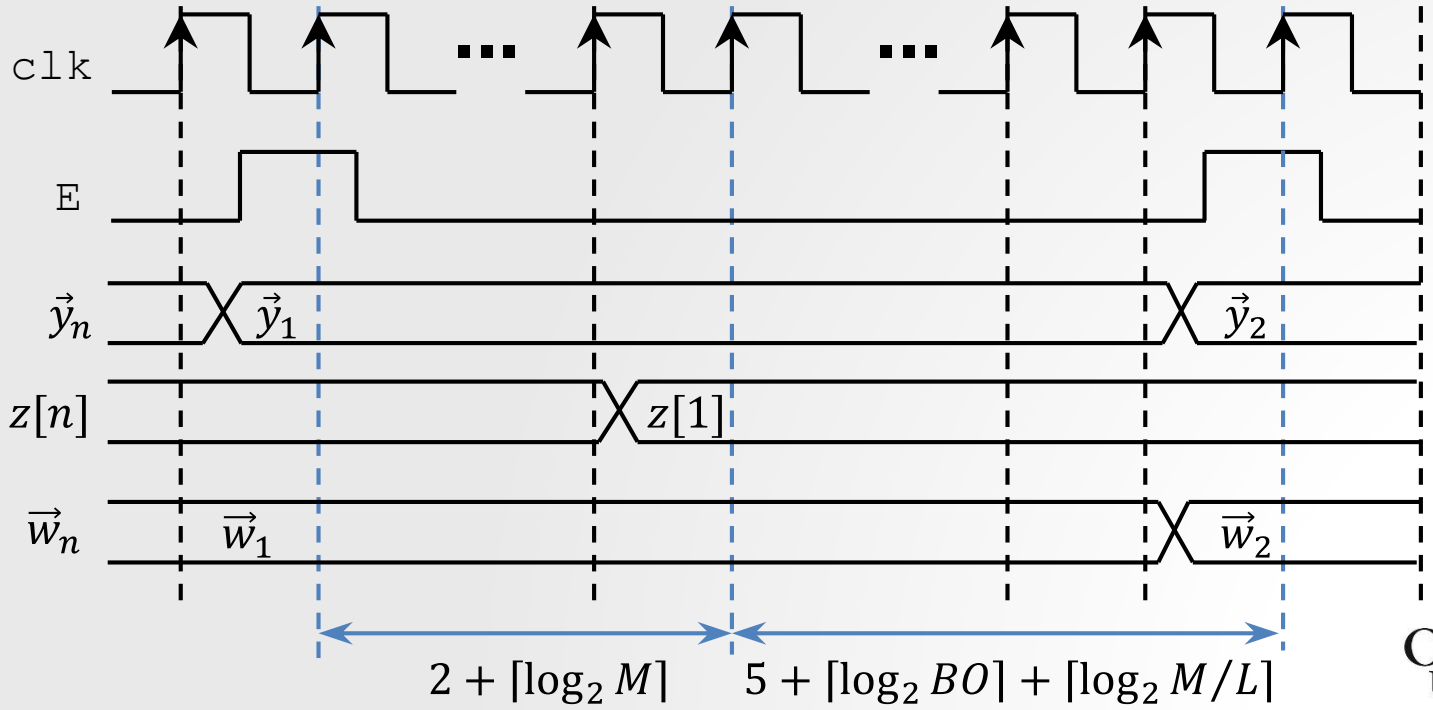
COMPLEX INNER PRODUCT

Latency: $\lceil log_2 BO \rceil + \lceil log_2(M/L) \rceil + 3$ *cycles*

Inner Product (DA): *Fully pipelined Distributed Arithmetic hardware than can process new data every cycle.*

Adaptive Beamformer

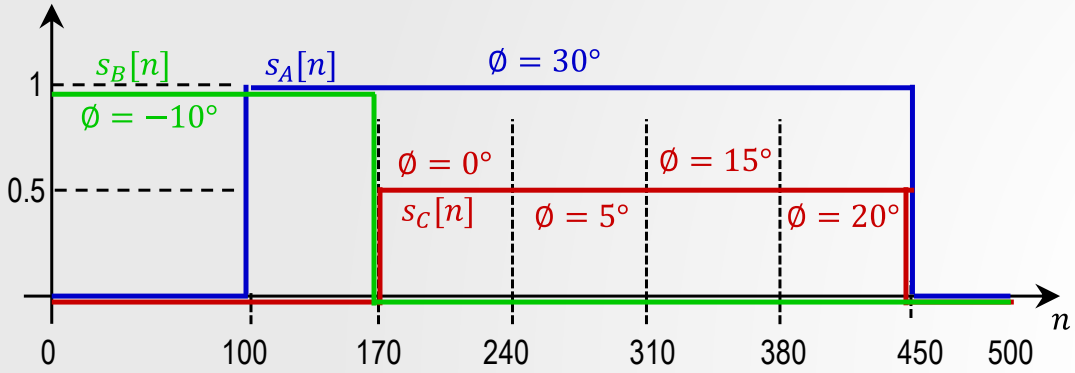
- ***Operation:***
  - *Dataflow controlled by an FSM.*
  - *E captures input snapshots at time n: $\vec{y}_n$. The complex weights $\vec{w}_n$ are available at this moment.*
  - *The output $z[n]$ is generated after $2 + \lceil \log_2 M \rceil$ cycles*
  - *Once $z[n]$ is ready, the next set of coefficients $\vec{w}_{n+1}$ is generated after $\lceil log_2 BO \rceil + \lceil log_2 (M/L) \rceil + 5$ cycles*



Experimental Setup

- ***Generation of input beamforming signals:***
  - *The snapshots $\vec{y}_n$ are retrieved from an array of sensors.*
  - *For this experiment, we use a Linear Array with M=6 sensors where N=500 snapshots are generated.*
  - $y_m[n] = s_m[n] + i_m[n] + r_m[n]$: *Samples at each sensor*
    $s_m[n]$: *Signal(s) of interest,* $i_m[n]$: *Interference (jammer),* $r_m[n]$: *noise.*

    $s_m[n] = s[n]e^{-j\vec{k}.\vec{x}_m}$, $i_m[n] = i[n]e^{-j\vec{k}.\vec{x}_m}$, *where* $\vec{k}.\vec{x}_m$ *depend on the angle of arrival of each signal and the array geometry.*
  - *Example: The following signals have different angle of arrivals (AOIs) and amplitudes, and appear at different time intervals.* $s_A[n]$: *AOI: 30 °,* $s_B[n]$: *AOI: -10 °,* $s_C[n]$: *AOIs: 0 °, 5 °,15 °, 20 °*



Experimental Setup

- ***Matrix C and constant c:*** *They control the beam pattern of the antenna array. In our experiment, we consider two Scenarios, each with different AOI (signals from previous figure).*

$a^H(\emptyset) = [e^{j\vec{k}.\vec{x}_1}\ e^{j\vec{k}.\vec{x}_2}\ \dots\ e^{j\vec{k}.\vec{x}_M}]$, $\mu = 0.01$ *(step size)*

TABLE I. SCENARIOS CONSIDERED IN OUR EXPERIMENTAL SETUP. FOR BOTH SCENARIOS, $C = [a^H(30°)\ \ a^H(-10°)\ \ a^H(0°)\ \ a^H(20°)]^T$

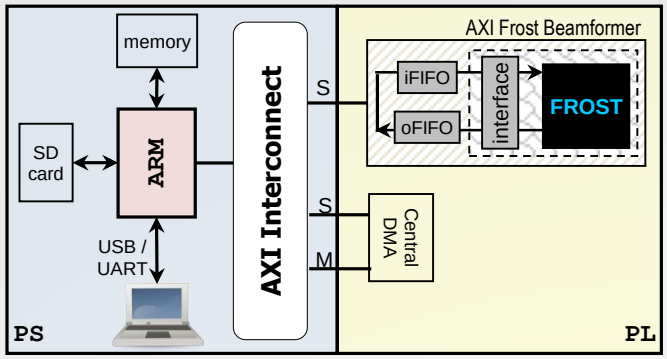
	SCENARIO A	SCENARIO B
SOI ($s[n]$)	$s_A[n], \emptyset_{SOI} = 30°$	$s_B[n], \emptyset_{SOI} = -10°$
Jammer ($i[n]$)	$s_B[n] + s_C[n]$	$s_A[n] + s_C[n]$
c	$[1\ 0\ 0\ 0]^T$	$[0\ 1\ 0\ 0]^T$

- ***Hardware Parameters:***
  - *B=8, M=6, N=500, NH=16 (bits per coefficient).*
  - *Five different fixed-point output formats are selected: [BO BQ] = [32 30], [24 22], [20 18], [16 14], [12 10]*

Number of integer bits? Depend on the experimental values. We can saturate if overflow occurs.

Experimental Setup

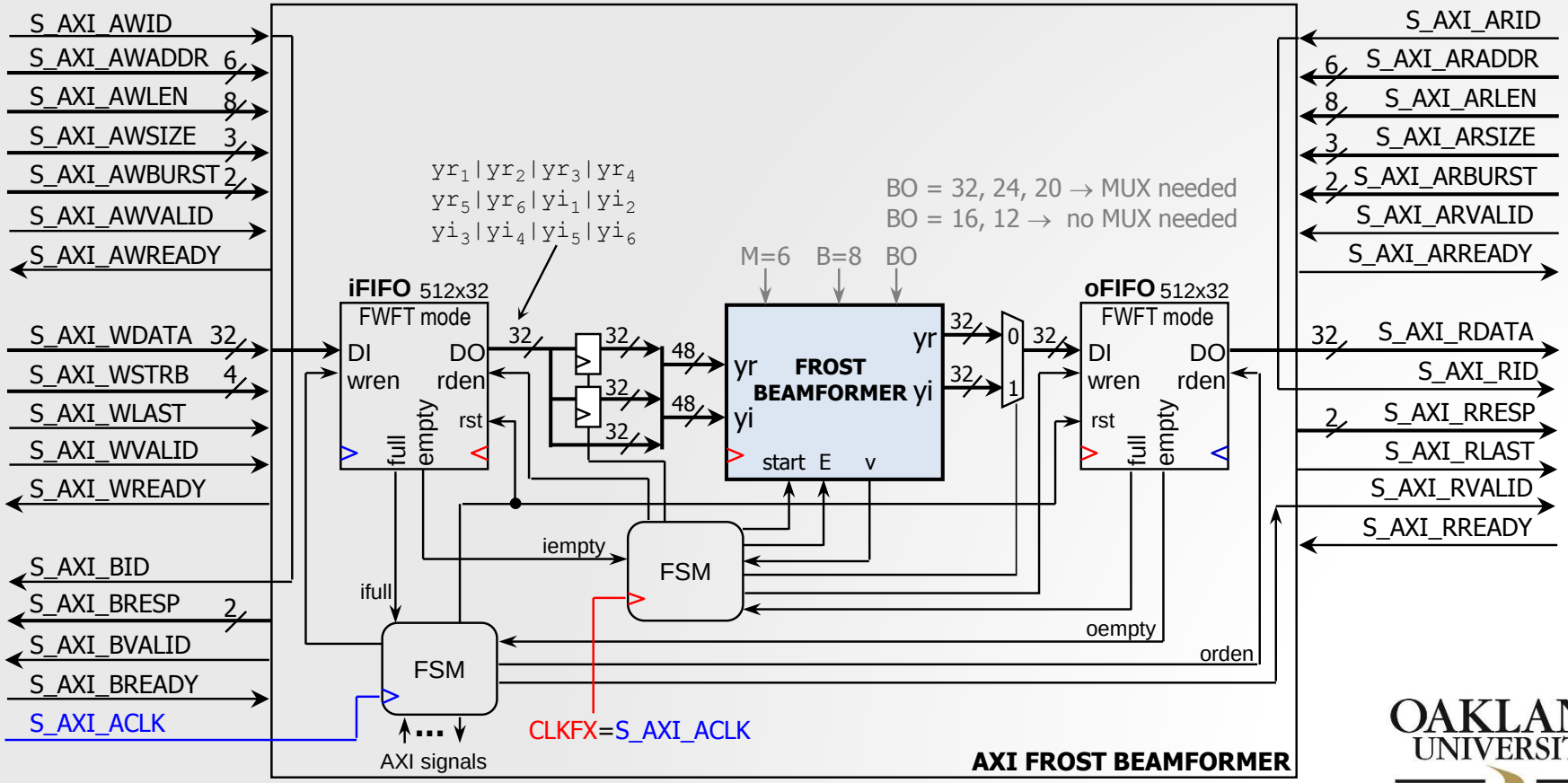
- ***Hardware validation:***
  - *The Frost Beamformer was included as a custom peripheral in an embedded system inside a Programmable SoC (Zynq-7000) in a XC702 Dev. Board.*
  - *Data is streamed and retrieved via an AXI4-Full Interface.*



- ***Accuracy Assessment (PSNR)***
  - *Complex output samples $z[n]$, we compare the power $|z[n]|^2$: fixed-point hardware results vs software routine with double floating point precision (MATLAB®). Two tests performed:*
  - *<u>Test 1</u>: FPGA and software (MATLAB) uses the quantized input samples ($B = 8$). This test assesses the quantization error incurred by the fixed-point architecture*
  - *<u>Test 2</u>: Only the FPGA uses the quantized input samples ($B = 8$) This test assesses the effect of both input quantization and fixed-point architecture on accuracy.*

Experimental Setup

- ***Hardware validation:***
  - *AXI Peripheral: It includes the Frost Beamformer and a 32-bit AXI4-Full Slave Interface (FIFOs, and control logic).*
  - *For M=6, B=8, a snapshot requires 48×2 bits (three 32-bit words). As for the output, for BO=16, 12 no MUX is needed.*



Results

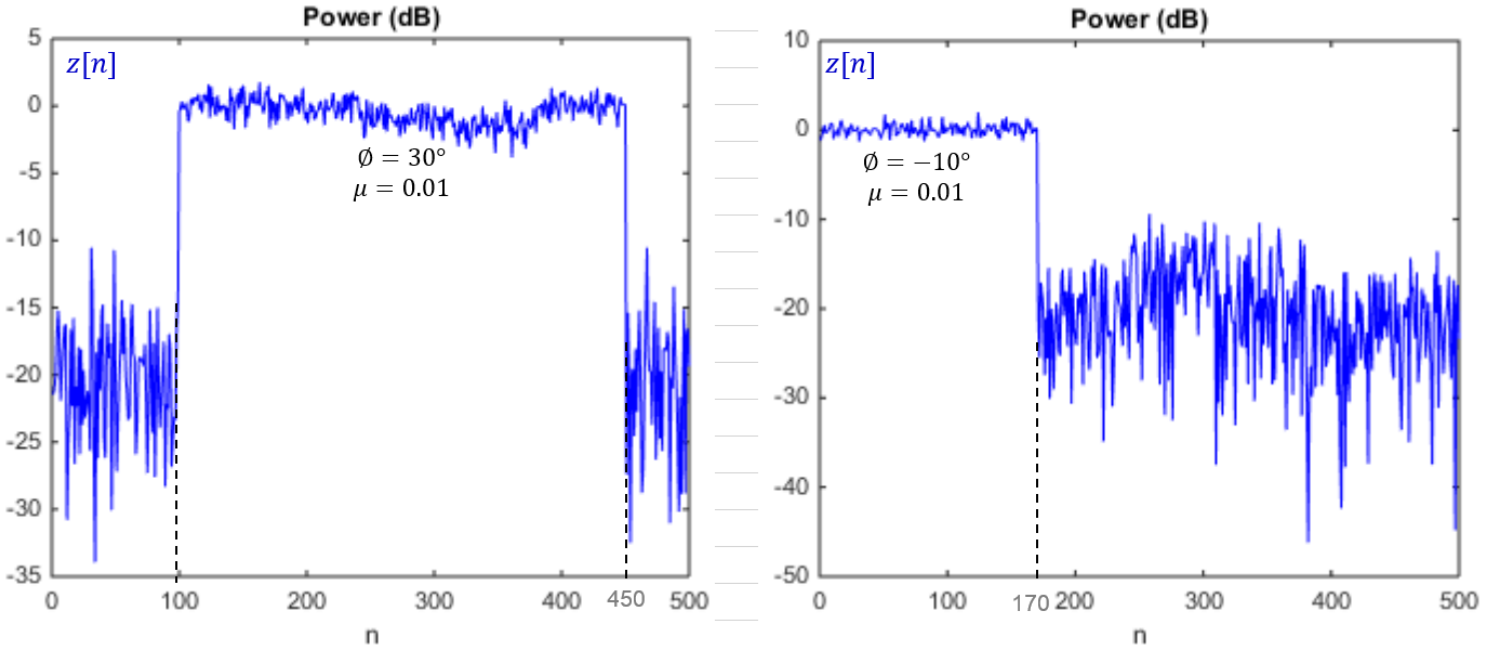
- ***Resources***
  - *The table shows resources (6-input LUTs, registers, and DSP48s) for the given design parameters: M=6, B=8, NH=16.*
  - *Resource consumption reasonable (<60% of the Zynq XC7020): fully parallel architecture requiring many multipliers.*

[BO BQ]	DSP48	%	FF	%	LUT	%
[12 10]	50	23	12231	12	11584	22
[16 14]	50	23	16167	15	15673	30
[20 18]	50	23	20585	19	20477	38
[24 22]	50	23	25411	24	24334	46
[32 30]	100	46	33808	32	32613	61

- ***Execution Time (performance bounds)***
  - *The hardware IP can process N snapshots in:*
    $(\lceil \log_2 BO \rceil + \lceil \log_2 (M/L) \rceil + \lceil \log_2 M \rceil + 7) \times N$ *cycles*
  - *For M=L=6, N=500, and for a frequency of 100 MHz:*
  - $BO = 12, 16$: *Execution time: 70 us. Throughput: 7.14×10^6 processed snapshots per second.*
  - $BO = 32, 24, 20$: *Execution time: 75 us. Throughput: 6.66×10^6 processed snapshots per second*

Results

- ***Accuracy***
  - *We show $|z[n]|^2$ (power (dB) of the output signal) for N=500 snapshots. This is data retrieved from the fixed-point hardware with [BO BQ] = [16 14]. Note how the Frost's beamformer is steered towards the desired directions.*
  - *Scenario A (left): We steer the Beamformer towards 30 °*
  - *Scenario B (right)): We steer the beamformer towards -10 °.*
  - *Note that the jammers are not present (there is no gain in the interval where the SOI is present).*



Results

- ***Accuracy (PSNR):***
  - *Test 1: High accuracy values (> 70 dB) for formats larger than [16 14], → fixed-point architecture is robust.*
  - *Test 2: Only the FPGA uses the quantized input samples. Accuracy is decent (> 60 dB) for formats larger than [16 14].*
  - *Increasing fractional bits from 14 to 30 only marginally increases accuracy. However, accuracy drops for the format [12 10] (~50dB).*

		SCENARIO A		SCENARIO B	
		Test 1	Test 2	Test 1	Test 2
[BO BQ]	[12 10]	53.1077	52.8447	54.4502	54.3013
	[16 14]	72.2134	64.4908	74.5182	65.3317
	[20 18]	73.0448	64.5916	80.2639	65.6460
	[24 22]	73.0650	64.5922	80.2631	65.6429
	[32 30]	73.0671	64.5924	80.2649	65.6429

  - *For our experiment, we found the fixed-point output format [16 14] to be optimal: a larger format increases resource usage with a negligible improvement in accuracy, and a smaller format results in a large drop in accuracy.*

Conclusions

- *Successfully validated a fixed-point Beamforming hardware that exhibits high throughput and reasonable resource requirements.*
- *Accuracy results suggest that fixed-point results are close to an implementation with double precision. Also, we experimentally verified that the Frost algorithm mitigates numerical errors: high PSNR values are obtained for small fixed-point formats.*
- *A drawback of fixed-point architecture is the number of integer bits: we can saturate, but the optimal number of integer bits depend on the dataset.*
- *Currently working on a self-reconfigurable version for a large set of hardware configurations and other array geometries. The goal is to implement a smart antenna that adapts to different beam patterns on-demand.*