FPGA-based Discrete Ambiguity Function for Stochastic Linear Time-Variant Channels

D. Márquez, A. Polo, D. Rodríguez, & M. Jiménez

Electrical & Computer Engineering Department
University of Puerto Rico at Mayagüez
Mayagüez, PR 00680

Institute for Computing and Informatics Studies

MWSCAS 2012
Aug 7, 2012

AIP LAB

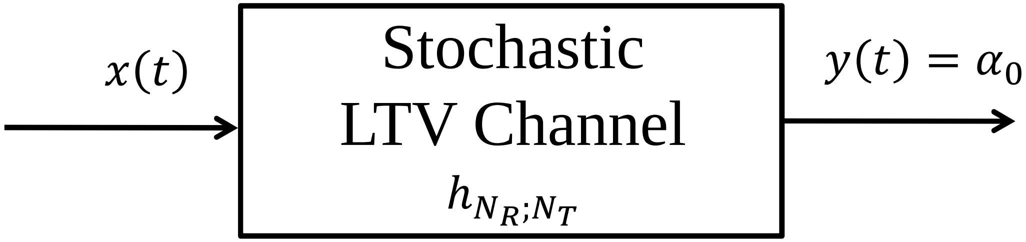
Outline

- Justification
- Stochastic Linear Time-Variant Channels (LTVC)
  - Characterization using the Discrete Ambiguity Function (DAF)
- DAF Formulation & Implementation
- FPGA Implementation Results
- Conclusions

Justification

- The Discrete Ambiguity Function (DAF)
  - Instrumental for the characterization of the scattering function of randomly time-varying linear underwater acoustic channels (UWACs).
  - The scattering function is used to model the delay-Doppler spread function.
- This work presents a novel signal processing algorithm framework for the FPGA implementation of the DAF.

Stochastic Linear Time-Variant Channels



$$y(t) = \alpha_0 x(t) + \alpha_1 x(t - \tau_1) e^{+j2\pi f_1 t} + n(t)$$

*LTV: Linear Time-Variant

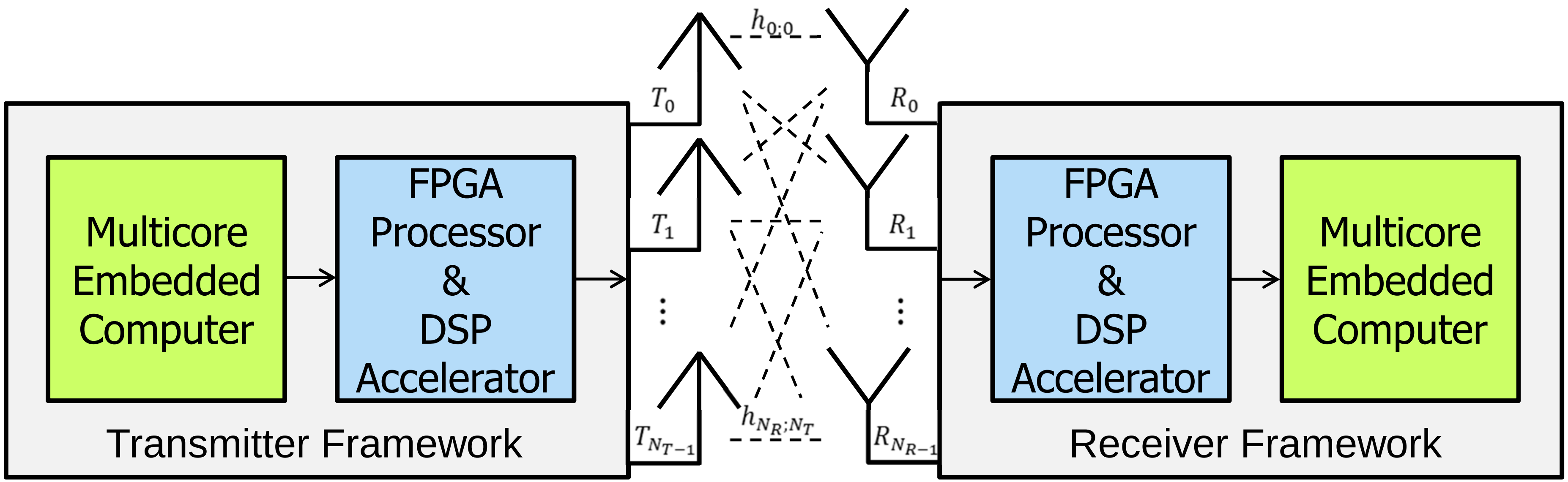
Parameter estimation:
τ_1:Delay spread
f_1:Doppler spread

The parameters τ_1, f_1 describe the state of an stochastic LTV channel at an specific instant of time.

The parameters τ_1, f_1 are estimated through the maximum of the expression:

$$\left| A_{x,y}(t,f) = \int_{-\infty}^{+\infty} x(t) y^*(t+\tau) e^{-j2\pi f t} \right|$$

Stochastic LTV Channels Characterization



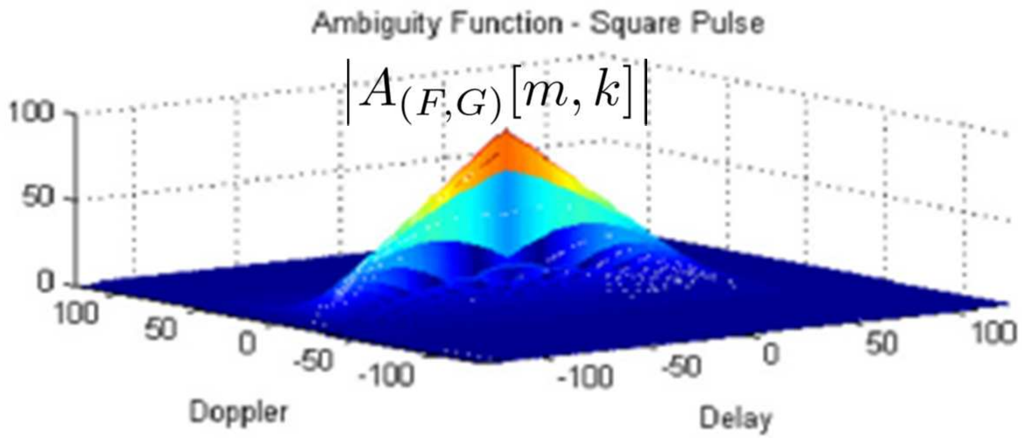
Discrete Ambiguity Function (DAF) Characterization

$$A : l^2(Z_N) \times l^2(Z_N) \rightarrow l^2(Z_N \times Z_N)$$

$$(F, G) \rightarrow A(F, G)$$

where,

$$A_{(F,G)}[m,k] = \sum_{n \epsilon Z_N} F[n] G^*[\langle n+m \rangle_N] e^{-j2\pi kn/N}; m,k \in Z_N$$



DAF Implementation Formulation

$$\mathcal{A}_{f,g}[m,k] = \sum_{n \in Z_N} f[n]\, g^*[\langle n+m \rangle_N]\, e^{-j\frac{2\pi}{N}kn}$$

$$\boxed{\mathcal{A}_{f,g}[m,k] \Leftrightarrow (I_N \otimes F_N)\, v}$$

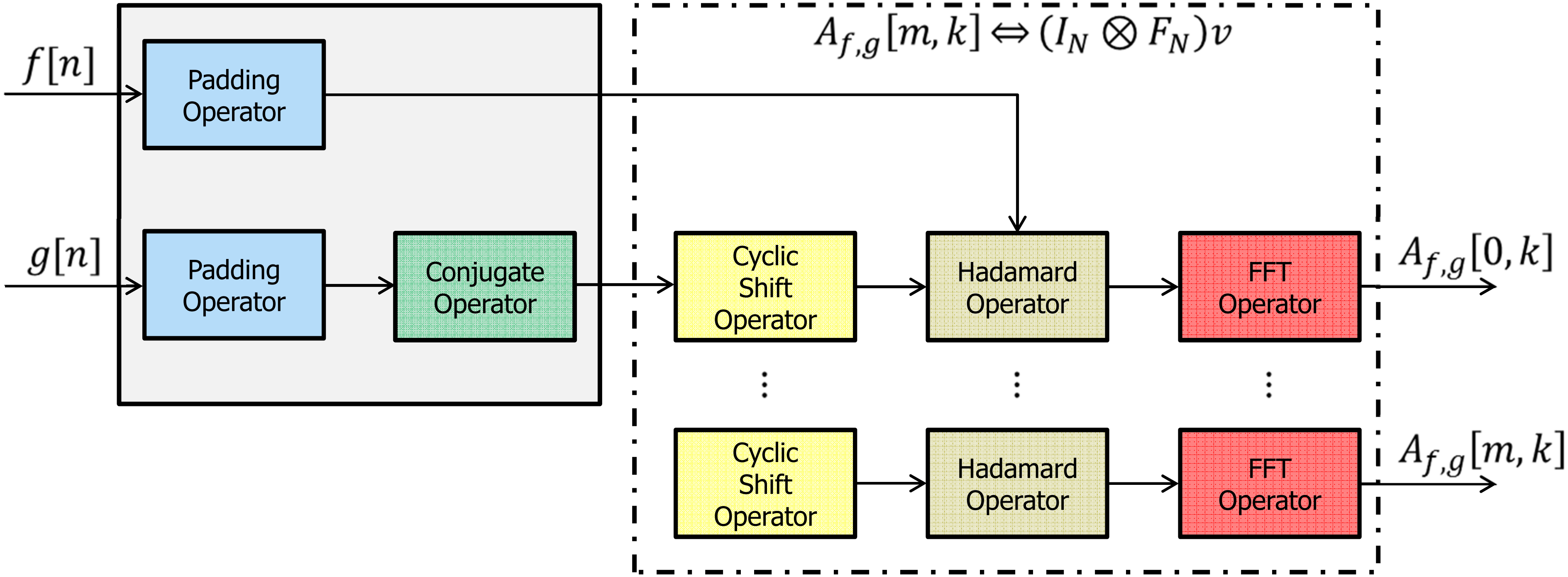
$$= \begin{bmatrix} F_N & & & \\ & F_N & & \\ & & \ddots & \\ & & & F_N \end{bmatrix}_{N^2 \times N^2} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-1} \end{bmatrix}_{N^2 \times 1} = \begin{bmatrix} F_N h_0 \\ F_N h_1 \\ \vdots \\ F_N h_{N-1} \end{bmatrix}_{N^2 \times 1} = \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_{N-1} \end{bmatrix}_{N^2 \times 1}$$

$$(I_N \otimes F_N)\, v \rightarrow \begin{bmatrix} H_0 & H_1 & \dots & H_{N-1} \end{bmatrix}_{N \times N} = \mathcal{A}_{f,g}[m,k]$$

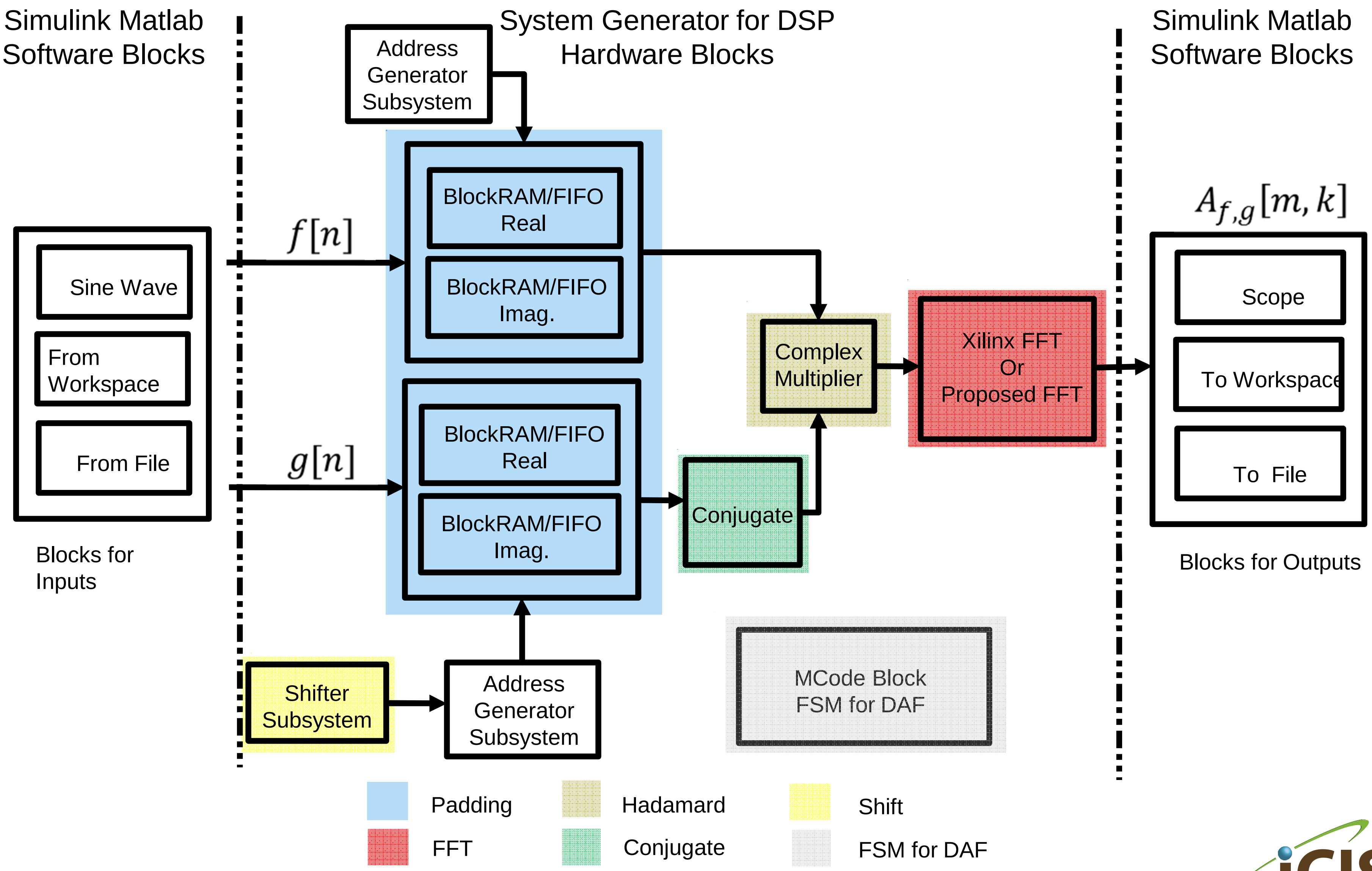
$$\text{Let } h_m[n] = f[n]\, g^*[\langle n+m \rangle_N] = v$$

$$\text{Then, } \mathcal{A}_{f,g}[m,k] = \sum_{n \in Z_N} h_m[n]\, W_N^{kn}, \text{ where } W_N^{kn} = e^{-j\frac{2\pi}{N}kn}$$

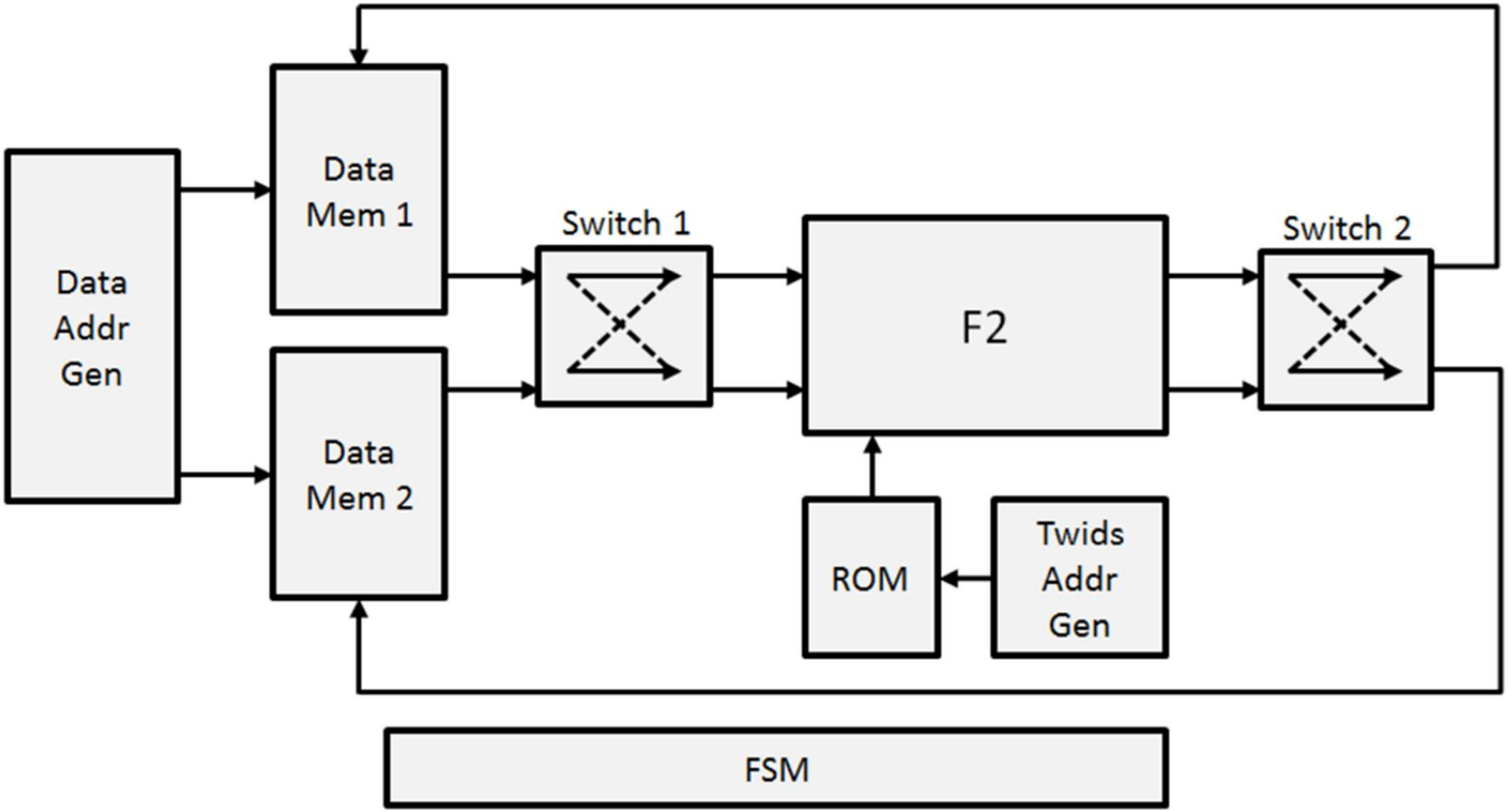
Algorithm for the DAF Implementation



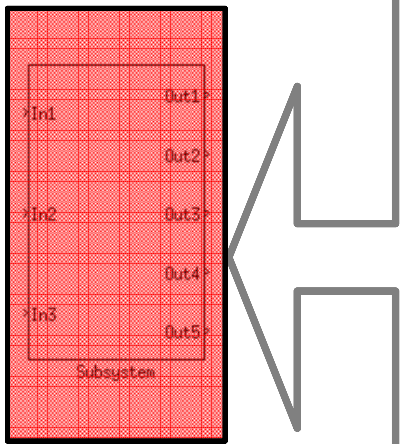
DAF Implementation in System Generator



Proposed FFT for the DAF Computation

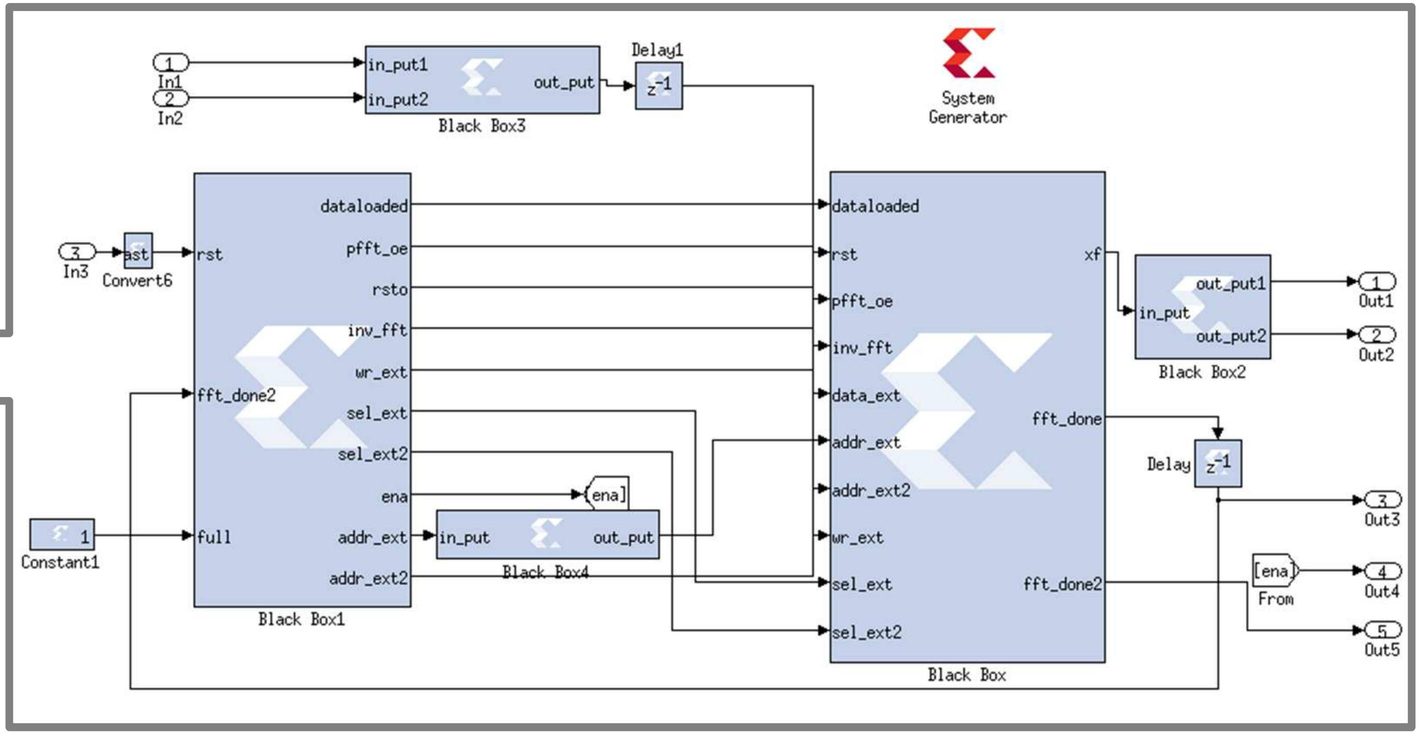


Proposed FFT in System Generator for DAF Computation



Subsystem Block
Simulink Matlab

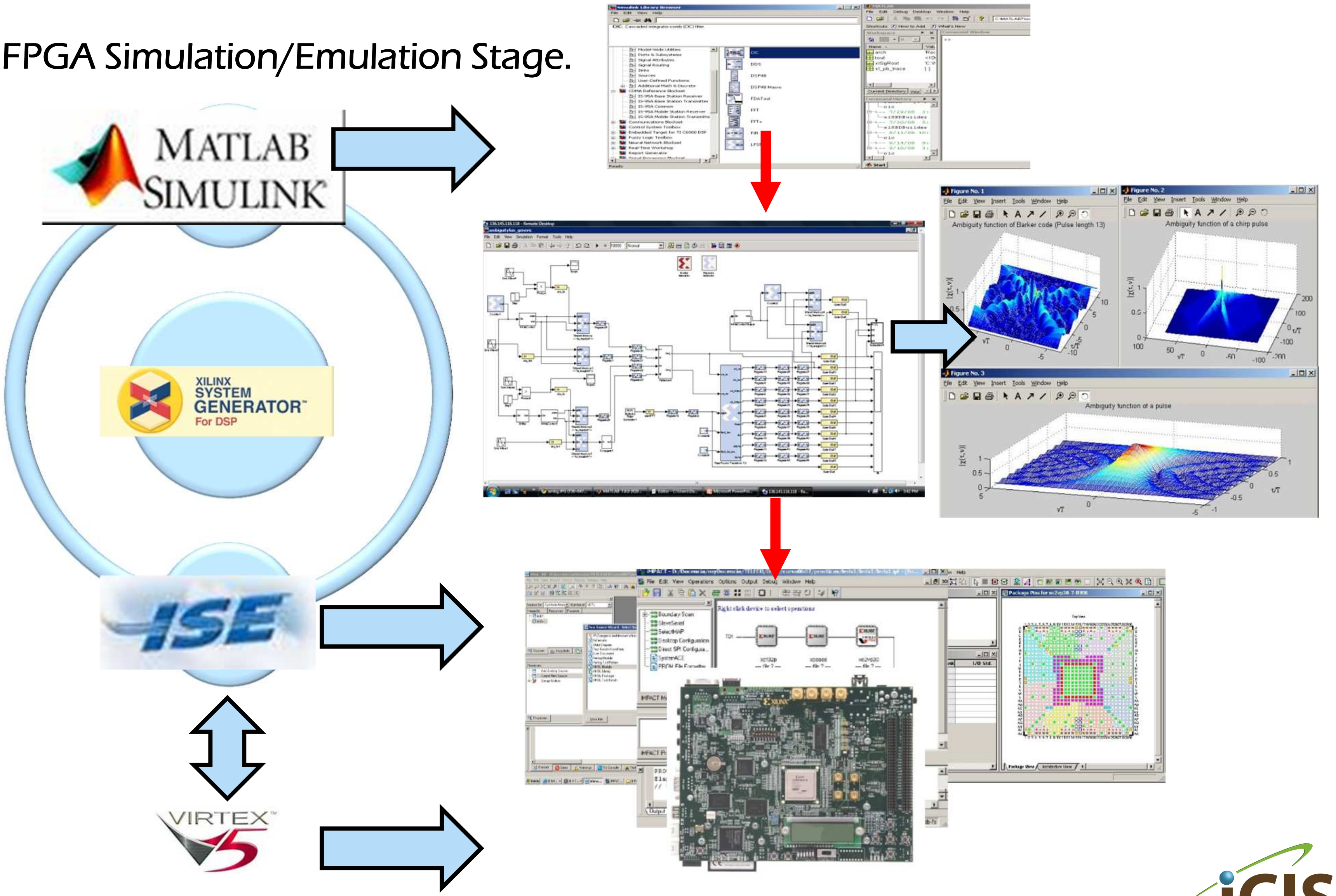
FFT



Set of BlackBox for proposed FFT
Xilinx System Generator for DSP

Integrated Hardware/Software Approach

- FPGA Simulation/Emulation Stage.



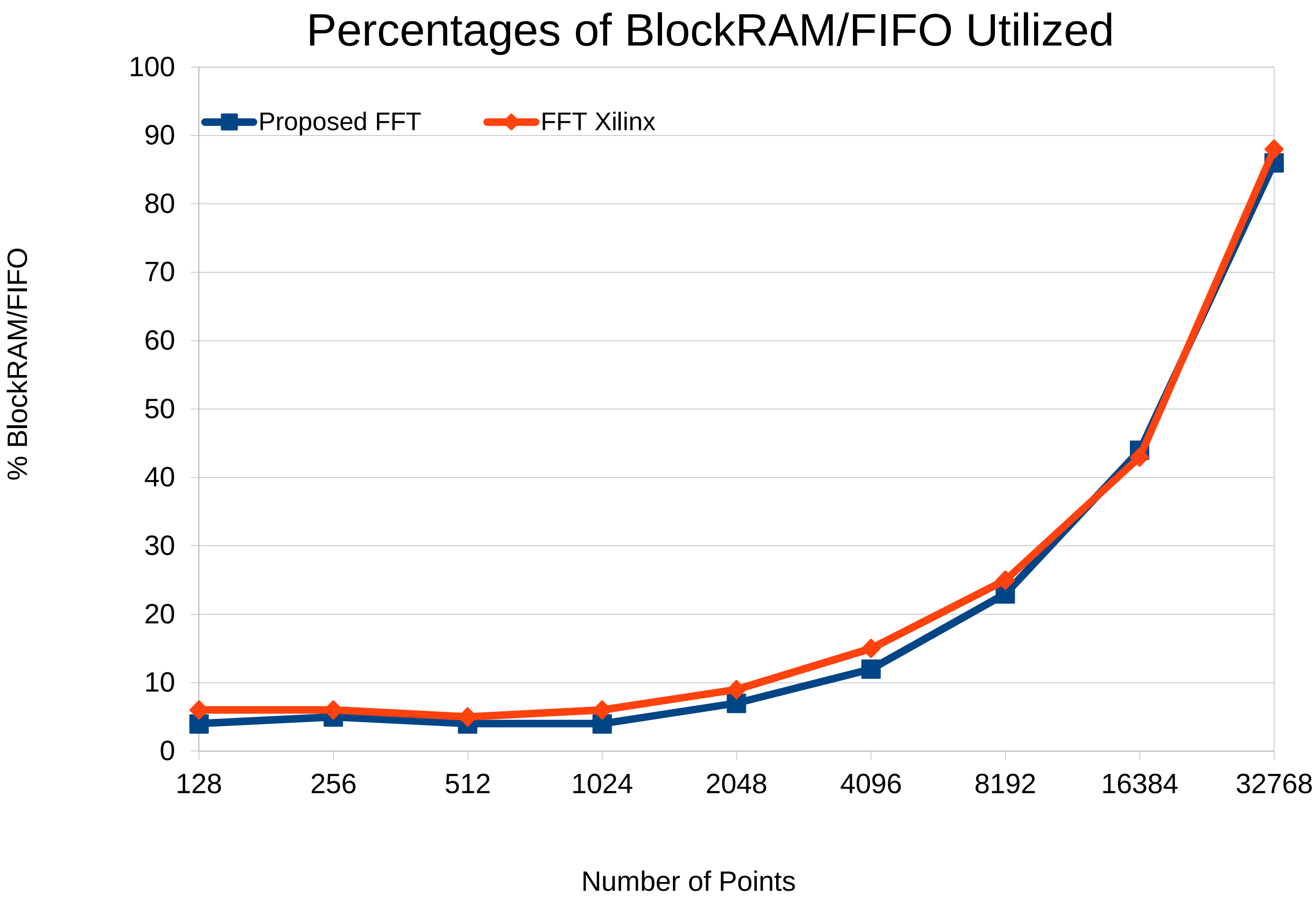
FPGA Implementation Results

LATENCY CYCLES FOR DAF USING XILINX FFT VS PROPOSED FFT			
Number of Points	DAF-Xilinx FFT	DAF-proposed FFT	Improvement %
128	717	650	9.34
256	1,415	1,363	3.67
512	3,011	2,908	3.42
1024	6,359	6,245	1.79
2048	13,547	13,422	0.92
4096	28,939	28,791	0.51
8192	61,728	61,568	0.26
16384	131,380	131,250	0.10
32768	278,860	278,670	0.07

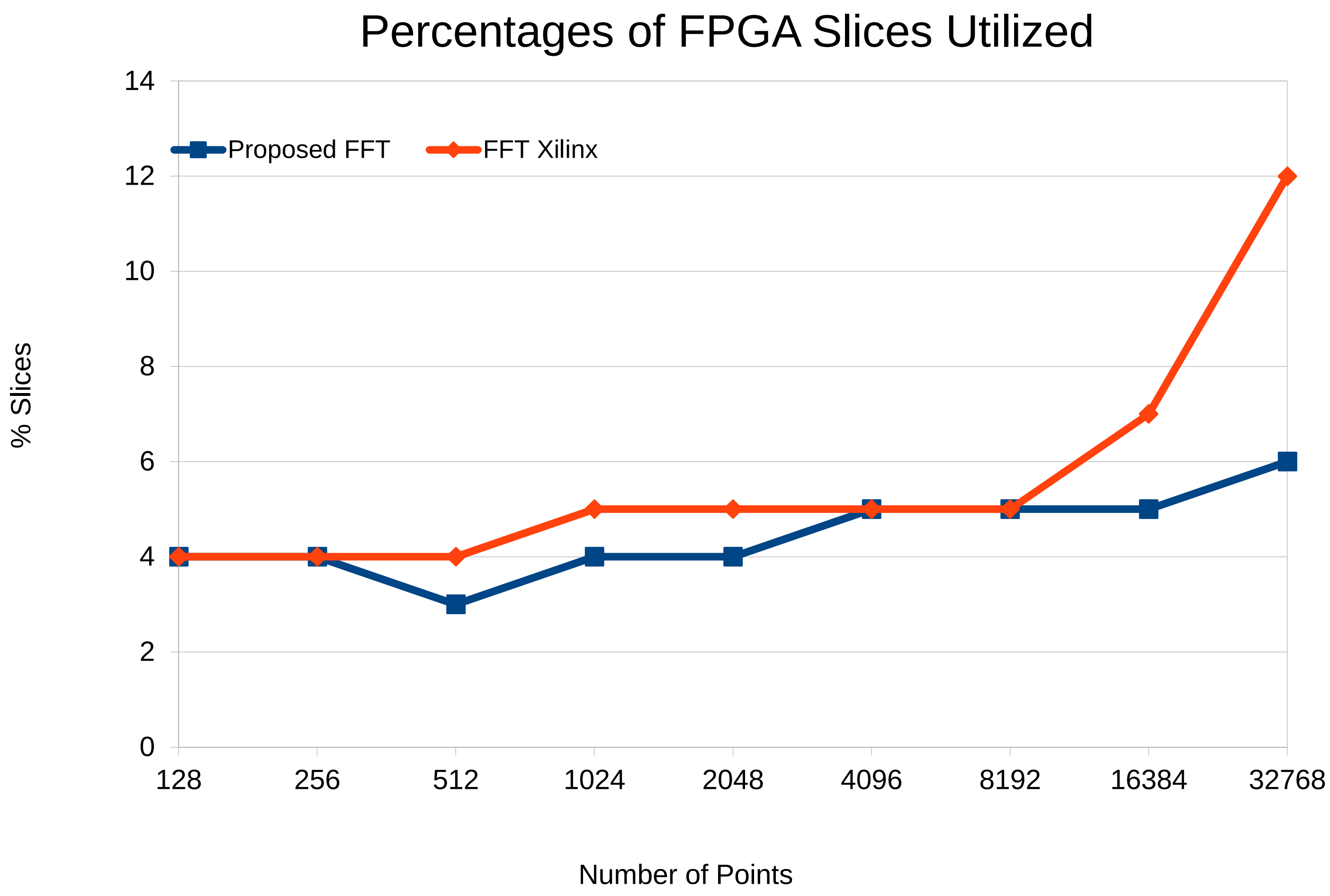
FPGA Implementation Results

COMPUTATION CYCLES FOR DAF USING XILINX FFT VS PROPOSED FFT			
Number of Points	DAF-Xilinx FFT	DAF-proposed FFT	Improvement %
128	106,382	90,503	14.93
256	424,206	379,911	10.44
512	1,796,622	1,616,391	10.03
1024	7,545,870	6,912,007	8.40
2048	31,909,902	29,571,079	7.33
4096	135,254,030	126,287,879	6.63
8192	572,702,730	537,837,578	6.09
16384	2,420,834,312	2,283,176,024	5.69
32768	10,210,902,032	9,668,460,540	5.31

FPGA Implementation Results

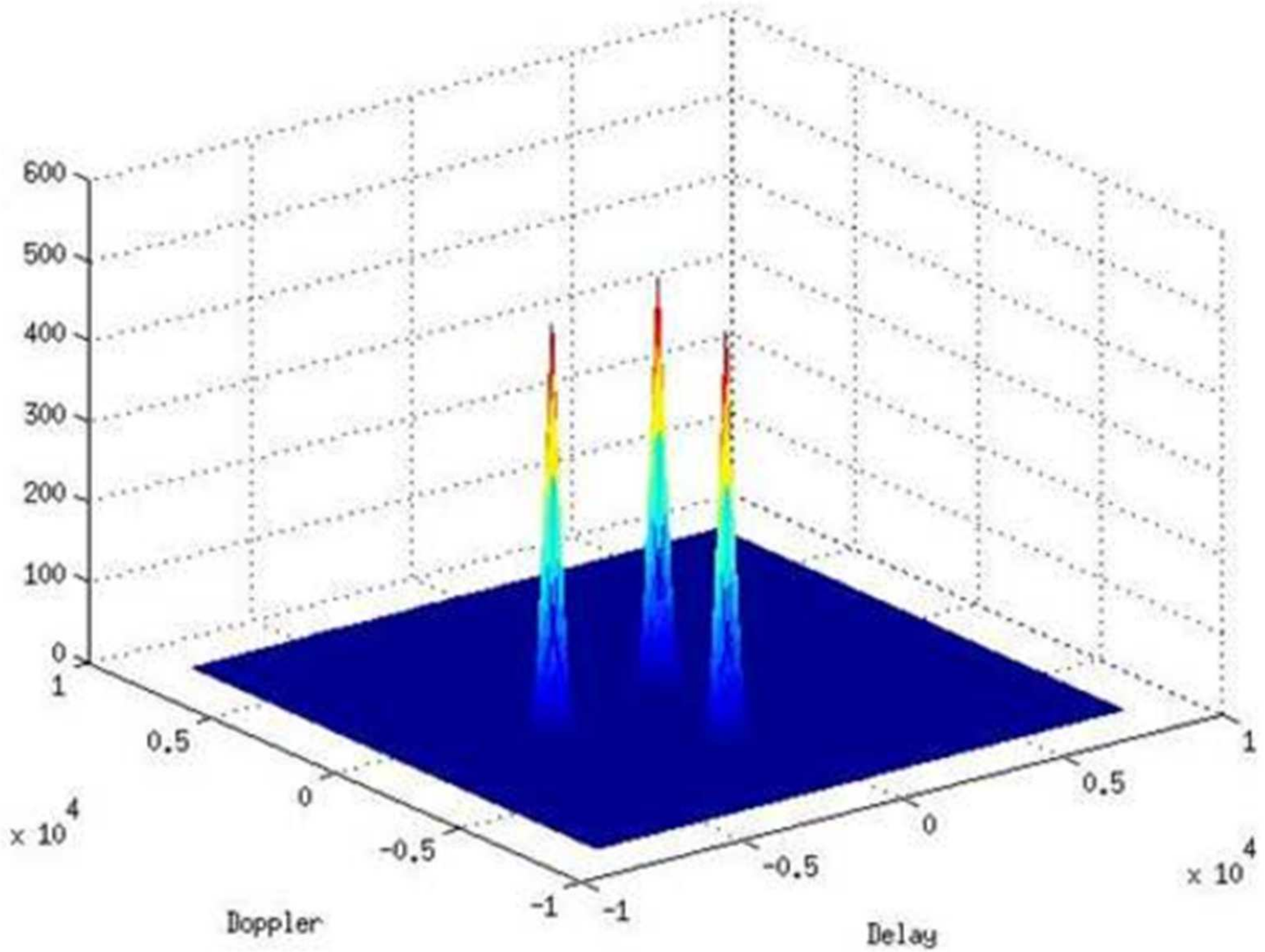


FPGA Implementation Results



FPGA Implementation Results

DAF Computation for Tracking Signals with 16384 Points



Conclusions

- The potential use of DAFs, to aid in the characterization of multiple input multiple output underwater time-frequency dispersive channels, also known as MIMO underwater doubly dispersive channels was demonstrated.
- An FPGA DAF implementation was described, comparing design results using Xilinx's FFT and a proposed scalable Pease's FFT. The results show a performance advantage of the scalable FFT DAF implementation over the DAF implementation using the Xilinx's FFT IP core.

Questions