# Tutorial: DOA Estimation using MUSIC Algorithm

## Introduction

In this tutorial, we will design an IP for DOA Estimation via MUSIC algorithm on Uniform Linear Array(ULA) and Sparse Array(SAA). The AXI stream IP will be created via Xilinx High Level Synthesis Tool. The IP is integrated with the Zynq Processing System via the Vivado IP Integrator. The SoC peripherals are managed by Xilinx Software Developement Kit. The design is tested on Zedboard.

## GitHub Link: https://github.com/Somya-Sharma/SpatialSensing_MUSIC.git

## Procedure

First, via High-level Synthesis, the floating-point IP for MUSIC algorithm will be created for Uniform Linear Array(ULA) and Sparse Array Arrangement(SAA). Second, the IP will be integrated with Zynq PS and AXI DMA. Then the software application is developed to test the hardware on Zynq System-on-chip.

## Create HLS Project — Step 1

### 1-1. Start Vivado HLS and Create a new project.

1-1-1. Open Vivado HLS by selecting **Start** > **All Programs** > **Xilinx Design Tools** > **Vivado HLS 2019.1.**

1-1-2. Select **Create New Project** and enter the project name as $MUSIC\_ULA$ or $MUSIC\_SAA$ for ULA or SAA respectively. Select appropriate project location and click on **Next**.

1-1-3. Enter the top function as $ULA\_STREAM$ for ULA or $MUSIC\_SPARSE$ for SAA. Select **Add Files** and add the source files *ula.h* and *ula.cpp* from MUSIC_tutorial/ULA/HLS. For SAA, add files from MUSIC_tutorial/SAA/HLS. Click on **Next**.

1-1-4. In testbench files, Select **Add Files** and add the *ula_test.cpp* as testbench file from MUSIC_tutorial/ULA/HLS. For SAA, you need to add *sparse_test.cpp* from MUSIC_tutorial/SAA/HLS. Click on Next.

1-1-5. Select the part as **Zedboard Zynq Evaluation Kit- xc7z020clg484-1**. Click **Finish** to create the project.

### 1-2. C Simulation, Synthesis, Co-SIM and Export IP

1-2-1. Click on **C Simulation** button from the toolbar. The C Simulation output will be displayed on the console.

1-2-2. Click on **Solution** > **Solutions Settings**. In the general tab, set the core_config option for DSP48 with latency 4. This is required only for ULA. Click on **C Synthesis** button from the toolbar. The synthesis results for ULA and SAA is shown in Fig. 1 and 2.

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 10.00 | 8.685 | 1.25 |

**Latency (clock cycles)**

**Summary**

| Latency | | Interval | | |
|---|---|---|---|---|
| min | max | min | max | Type |
| 6273 | 6613 | 6273 | 6613 | none |

**Detail**

**Instance**

**Loop**

**Utilization Estimates**

**Summary**

| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | 5 | - | - | - |
| Expression | - | - | 0 | 1419 | - |
| FIFO | - | - | - | - | - |
| Instance | 16 | 125 | 37757 | 36447 | 0 |
| Memory | 18 | - | 256 | 24 | 0 |
| Multiplexer | - | - | - | 2241 | - |
| Register | 0 | - | 6185 | 928 | - |
| Total | 34 | 130 | 44198 | 41059 | 0 |
| Available | 280 | 220 | 106400 | 53200 | 0 |
| Utilization (%) | 12 | 59 | 41 | 77 | 0 |

Figure 1: ULA Synthesis Results

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 10.00 | 9.634 | 1.25 |

**Latency (clock cycles)**

**Summary**

| Latency | | Interval | | |
|---|---|---|---|---|
| min | max | min | max | Type |
| 8156 | 9704 | 3348 | 4896 | dataflow |

**Detail**

**Instance**

**Loop**

**Utilization Estimates**

**Summary**

| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | - | - | - | - |
| Expression | - | - | 0 | 154 | - |
| FIFO | - | - | - | - | - |
| Instance | 42 | 176 | 53353 | 51116 | 0 |
| Memory | 27 | - | 1216 | 94 | 0 |
| Multiplexer | - | - | - | 279 | - |
| Register | - | - | 31 | - | - |
| Total | 69 | 176 | 54600 | 51643 | 0 |
| Available | 280 | 220 | 106400 | 53200 | 0 |
| Utilization (%) | 24 | 80 | 51 | 97 | 0 |

Figure 2: SAA Synthesis Results

1-2-3. Click on RTL Co-simulation from toolbar to run RTL Co-simulation. The co-sim result should be PASS.

1-2-4. Click on **Export IP** from toolbar to export the design.

## Create Vivado Project — Step 2

**2-1. Start Vivado and Create a new project.**

2-1-1. Open Vivado by selecting **Start** > **All Programs** > **Xilinx Design Tools** > **Vivado 2019.1.**

2-1-2. Create a Vivado Project in working directory with the default options. In the part select, select board as Zedboard.

**2-2. Add HLS IP to the project.**

2-2-1. Add HLS IP to the Vivado User IP Repository. In the Flow Navigator Plane, under Project Manager, click on Settings.

2-2-2. In the Project Settings, expand IP and select Repository. In IP Repositories, click on add button and select the location of your HLS IP (MUSIC_ULA/solution0/impl/ip). For SAA, the location will be different(MUSIC_SAA/solution0/impl/ip). Click on OK.

2-2-3. Click OK. The IP $ULA\_STREAM$ in case of ULA and $MUSIC\_SPARSE$ in case of SAA will be added in the User IP Repository.

**2-3. Create Block Design.**

2-3-1. Click on **Create Block Design** and name the block design *system*.

2-3-2. In the tcl console enter source ULA_bd.tcl to create the block design for ULA and source SAA_bd.tcl for SAA. This script is provided in MUSIC_tutorial/ULA/Vivado and MUSIC_tutorial/SAA/Viva respectively.

2-3-3. Select the **Address Editor** tab. Expand the design heirarchy. Expand **Unmapped Slaves**, if any, and right-click and select **Auto-Assign Address**. Similarly, Expand **Excluded segment**, if any, and right-click and select **Include segment**.

2-3-4. Select **Tools** > **Validate Design.**

2-3-5. Select **File** > **Save block design.**

**2-4. Generate Bitstream and Export Hardware**

2-4-1. In the Sources tab, right click on system.bad and click on Generate Output Products. Click OK.

2-4-2. After the output products are generated, again right-click on system.bd and select Create HDL Wrapper. Click on OK.

2-4-3. Click on Generate Bitstream.

2-4-4. After the bitstream is generated, select File > Export > Export Hardware. In the dialog box, check Include bitstream.

2-4-5. Select File > launch SDK.

## Create the software application — Step 3

### 3-1. Create a Board Support Package.

3-1-1. In SDK, select **File** > **New** > **Board Support Package**.

3-1-2. Click **Finish** with the default settings (with standalone operating system). This will open the Software Platform Settings form showing the OS and libraries selections.

3-1-3. Click OK to accept the settings and create the BSP.

### 3-2. Create an application.

3-2-1. Select **File** > **New** > **Application project**.

3-2-2. Enter ULA as the Project Name, and for Board Support Package, choose Use Existing (standalone_bsp_0 should be the only option). If this option does not appear, make sure that the hardware specification is set to the hdf created above.

3-2-3. Click **Next**, and select *Empty Application* and click **Finish**.

3-2-4. Expand the **ULA** entry in the project view, right-click the *src* folder, and select **Import**.

3-2-5. Expand **General category** and double-click on **File System.**.

3-2-6. Browse to MUSIC_tutorial/ULA/SDK and click **OK**. Select *main.c, lib_xmULA.h, lib_xmULA.c, platform.h, platform.c, platform_config.h.* For SAA, browse to MUSIC_tutorial/SAA/SDK and click **OK**. Select *main.c, lib_xmSparse.h, lib_xmSparse.c, platform.h, platform.c, platform_config.h.* **click Finish** to add the file to the project.

3-2-7. Increase the stack size to $0 \times 3000$ for ULA and $0 \times 6000$ for SAA in the linker script.

## Test the design — Step 4

### 4-1. Connect the board with micro-USB cable connected to the UART. Power On the board.

### 4-2. Start a terminal emulator program such as TeraTerm or HyperTerminal. Select an appropriate COM port (you can find the correct COM number using the Control Panel). Set the COM port for 115200 baud rate communication.

4-2-1. Start a terminal emulator program such as **TeraTerm** or **HyperTerminal**.

4-2-2. Select the appropriate COM port (you can find the correct COM number using the Control Panel).

4-2-3. Set the COM port for **115200** baud rate communication.

4-2-4. Right click on ULA in the project view and select Build Configurations > Set Active > Release.

**4-3. Run the application on Zedboard.**

4-3-1. Select Xilinx > Program FPGA.

4-3-2. Right click on ULA again, select Run As > Launch on Hardware Debugger.

4-3-3. The result as shown in Fig. 3 and 4 is displayed on the UART for ULA and SAA. Acceleration factor is the speedup for MUSIC algorithm on hardware as compared to its ARM implementation.

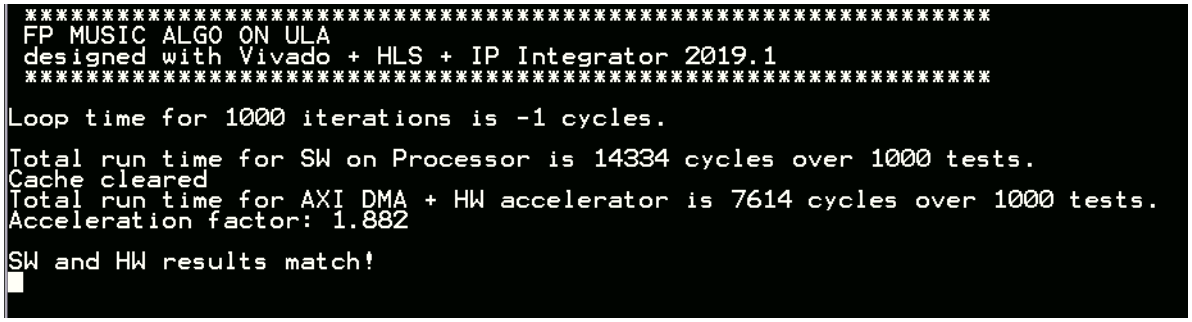


Figure 3: ULA Output

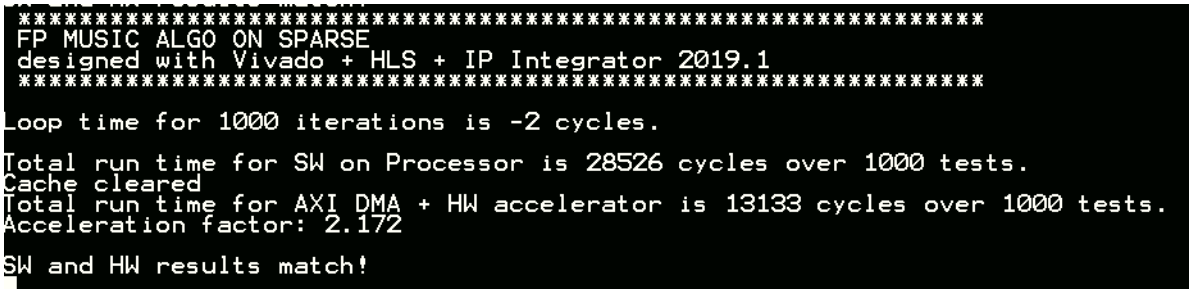


Figure 4: SAA Output