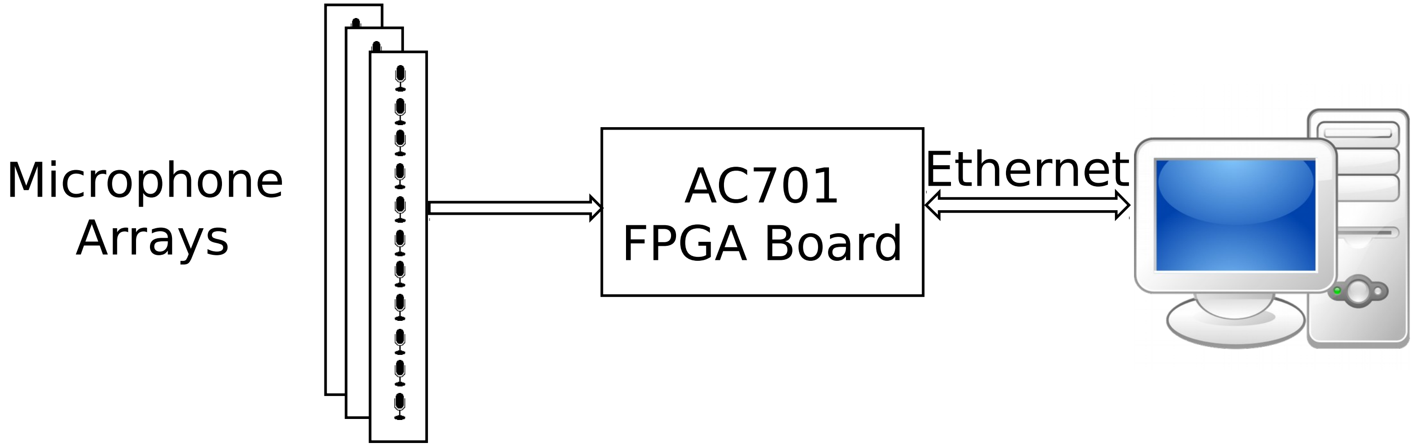
DAT096 Reference Project

Bhavishya Goel

CHALMERS
UNIVERSITY OF TECHNOLOGY

Hardware Design for DAT096

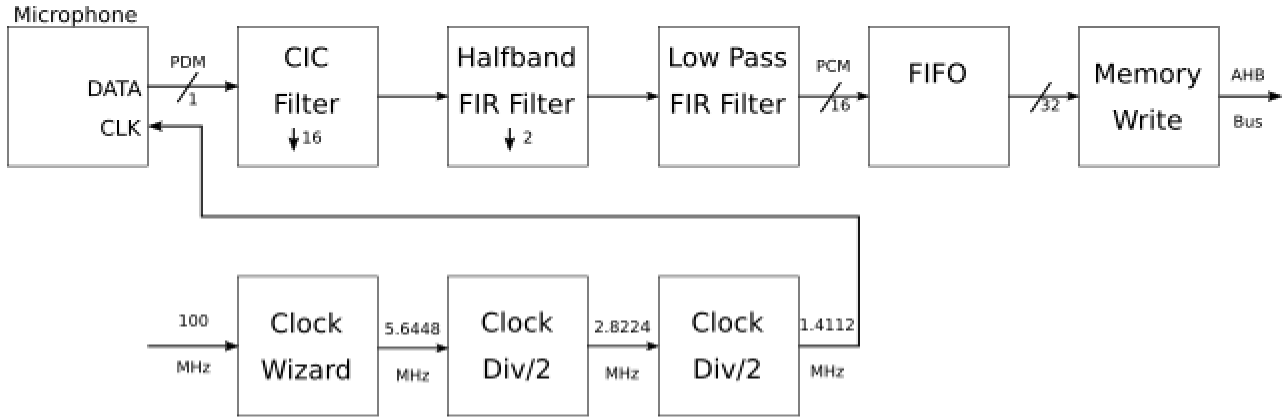
- The input from microphones is sent to FPGA board in 1-bit PDM Format
- The FPGA converts 1-bit PDM to 16-bit PCM
- The raw PCM data is sent to PC over Ethernet



Reference Design

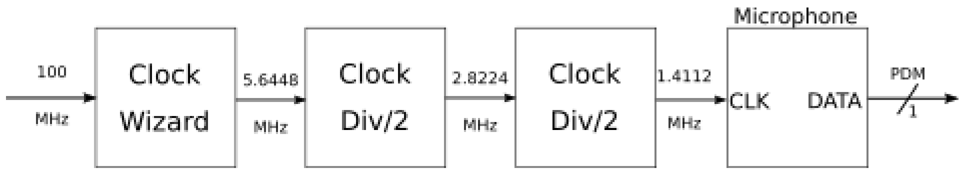
- You have been given an FPGA source code that takes input from a **single** microphone and sends to PC
- You need to expand this code to simultaneously take input from multiple microphones and implement acoustic source localization (either on FPGA itself or on PC)
- You can download the reference design `dat096_ref_design.zip` from Canvas

Reference Design Block Diagram



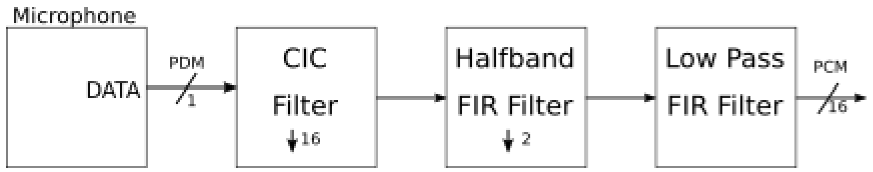
- This reference design writes PCM data to on-board Memory instead of streaming the data directly over Ethernet (streaming will be supported in future).
- A special utility on PC connects to the board, reads the memory contents and dumps them in a file.
- The PCM sampling rate in this design is 44.1 Khz and the PDM rate is 1.4112 Mhz (44.1 Khz × 32). You can change this sampling rate as per your requirements.

Reference Design Clocking



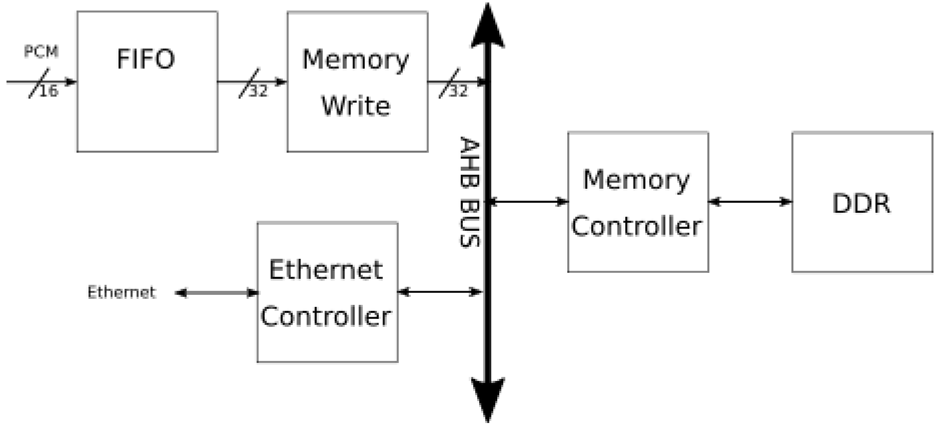
- The microphone output sampling rate depends on the input clock, so the first step is to generate appropriate clock inside FPGA.
- The design uses Clocking Wizard IP to generate clock. But because of hardware limitations, it cannot generate 1.4112 Mhz clock directly from 100 Mhz input clock. So we have to use clock dividers to bring down the clock frequency to 1.4112 Mhz.
- But you may not need clock dividers if you output clock frequency is within the clock wizard specifications. For example, you can directly generate 4.8 Mhz clock from 100 Mhz input clock (on AC701).

PDM to PCM Conversion



- The output from Microphone is in 1-bit PDM format, so it needs to be converted to PCM before it can be processed as shown in the figure above.
- The design uses CIC Filter IP to decimate the PDM input by a factor of 16, Halfband FIR Filter to downsample by CIC output by 2 and another instance of FIR filter configured as low pass single rate filter to remove the high frequency noise.
- The least significant bits of FIR filter are truncated to get 16-bit PCM output.
- You can read more about the PDM to PCM conversion filter design in `PCM to PCM Conversion.pdf` on Canvas.
- This design is optimized for good SNR. If you don’t care about SNR, you can optimize this filter design in later stages of the project.

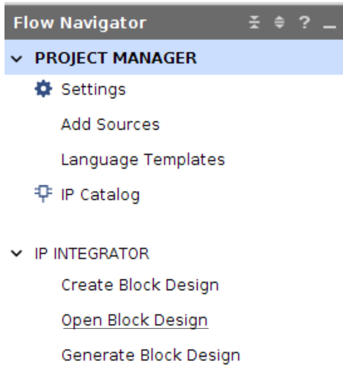
Memory and Etherent Interface



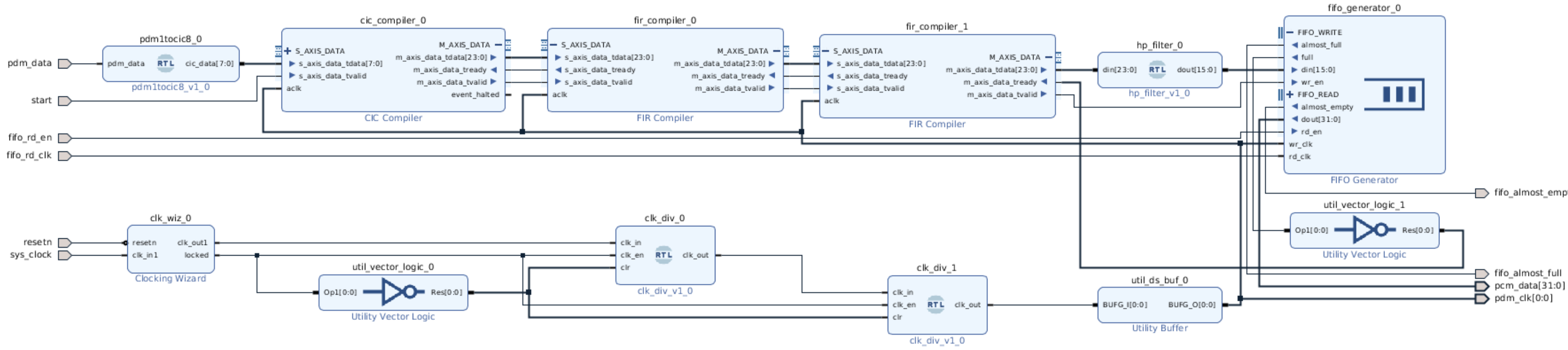
- In this design, we write the PCM samples to on-board DDR and then read them from PC over ethernet
- For this, we need memory controller, ethernet controller and a bus implementation in FPGA to connect these blocks
- We use the readymade blocks from GRLIB library get this interface up and running
- Hopefully, you won’t need to modify this part of the design at all

Vivado Project

- You have been given a zip of vivado project. Start with extracting the file locally.
- Now open vivado, `File→Project→Open…`, navigate to extracted reference project folder and open *`dat096_ref_design.xpr`*.
- Once the project is open, click on *`Open Block Design`* under `IP Integrator`.

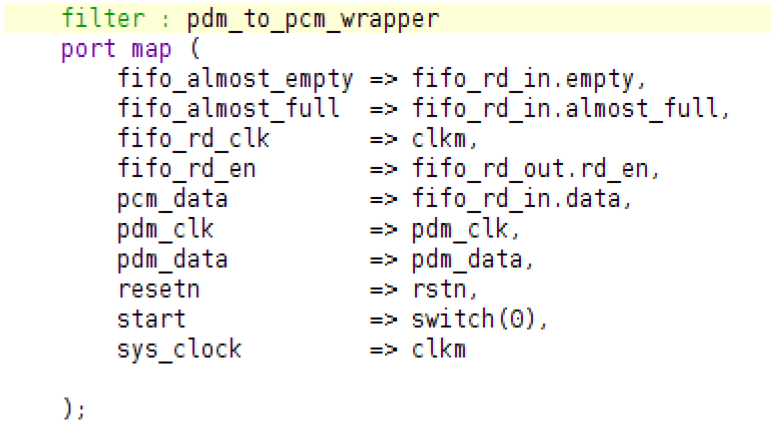


Vivado Project contd.



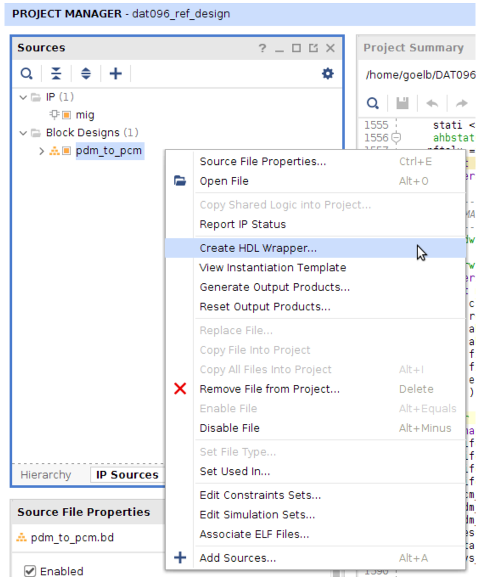
- The entire PDM to PCM conversion design described before is implemented as block design in this project.
- Feel free to explore different IP blocks
- **Note**: Using Block design feature in Vivado is good as a beginner, but moving forward, you should instantiate the IPs in your VHDL code directly instead of using block design feature since it is not scalable.

Vivado Project contd.



```
    filter : pdm_to_pcm_wrapper
    port map (
        fifo_almost_empty => fifo_rd_in.empty,
        fifo_almost_full  => fifo_rd_in.almost_full,
        fifo_rd_clk       => clkm,
        fifo_rd_en        => fifo_rd_out.rd_en,
        pcm_data          => fifo_rd_in.data,
        pdm_clk           => pdm_clk,
        pdm_data          => pdm_data,
        resetn            => rstn,
        start             => switch(0),
        sys_clock         => clkm

    );
```



- Take a look at how the pdm to pcm block has been instantiated in the top file `leon3mp.vhd`
- The `pdm_to_pcm_wrapper.vhd` file is generated by vivado from the block design by right clicking on `pdm_to_pcm` block design in IP Sources and then selecting `Create HDL Wrapper`.

Hardware Setup

- Generate the bitstream by clicking on `Program and Debug` → `Generate Bitstream`. This will take around 5-7 minutes.
- Before powering up the AC701 board, make sure that DIP Switch SW1 is on ON-OFF-ON position (JTAG boot mode) and SW2 has all switches set to OFF.
- Connect the FMC breakout board directly to FMC connector (J30) of AC701 and then connect microphone array board to connector ”A” of breakout board using the ribbon cable.
- You have been provided a USB-to-Ethernet adapter. Plug it in USB port of PC, situated on the front panel, at the very bottom (if using your own laptop, it doesn’t matter). Connect the etherent port of AC701 board with adapter using ethernet cable.
- Now connect the AC701 board to a power source using power cable and connect the USB-JTAG port to PC’s USB port using a USB cable. Refer to AC701 user guide if required.
- Power up the board and open Hardware Manager in Vivado. Click on `Open Target` → `Auto connect`. Vivado should now be connected to the board. Program the newly generated bitfile on the board.

Connecting to the board

- We will use GRMON Utility from Gaisler to connect to the board. It is already installed on lab computers, but if you can also install it on your own laptop
- Open windows command prompt. Enter command

```
”C:\Program Files\grmon-eval-3.2.0\windows\bin64\grmon.exe” –eth 192.168.0.51
```

- If everything went right, you should now be connected to the board.
- You can look at the contents of the memory by using command

```
grmon> mem 0x40000000
```

- Now to start recording the microphone input, set dip switch SW2(0) to ON for few seconds, then set it back to OFF. Now check memory contents at `0x40000000` to see if the memory contents have changed.
- To dump 5 seconds of data from memory to file, enter command

```
grmon> dump –binary 0x40000000 441000 pcm_5s.raw
```

- You can listen to the recorded data by importing the binary file `pcm_5s.raw` in a free software like Audacity or convert raw pcm to wav file using Matlab/Python.