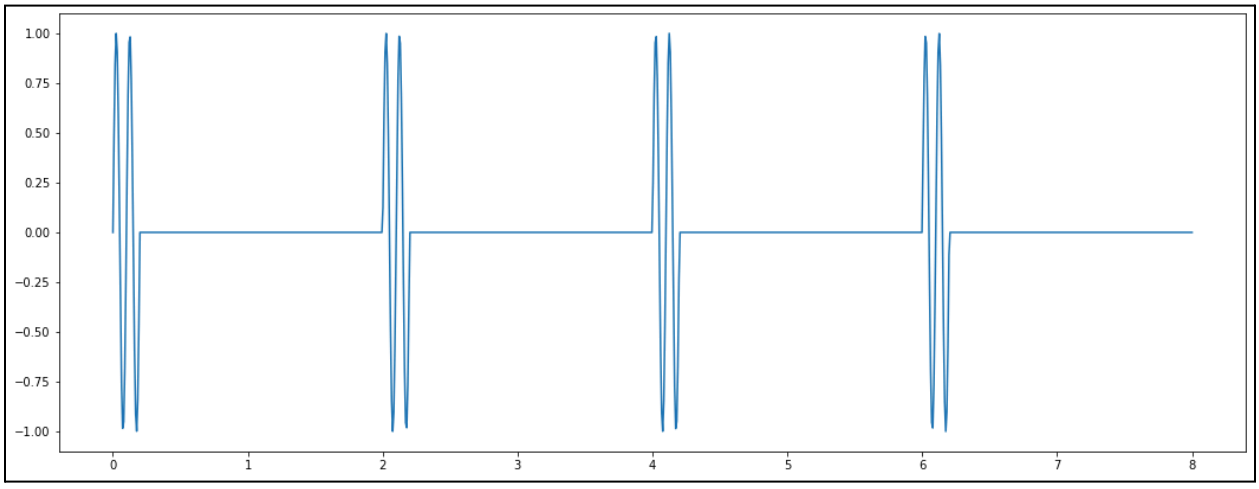
Pinger Localization using FPGA

Overview

This work is done related to navigating an AUV(Autonomous Underwater Vehicle) towards Pinger

What is pinger? How does our vehicle hear them?
Pingers are acoustic devices, which emit sinusoidal pressure waves(Sound waves). Depending upon the model number, we can emit sound waves of any frequency.

Currently assuming the following constrains:-
The pinger lies at the bottom of the waterbed, emitting a signal of 25-45kHz for 2ms, with a period of 2s.



Total hydrophones used - (4 in quantity) to detect and sense the pressure waves.

A simple example is that you(AUV) are blindfolded(unable to detect any task) and someone(pinger) is calling you, then after hearing(Hydrophones) you would approach that person(task).

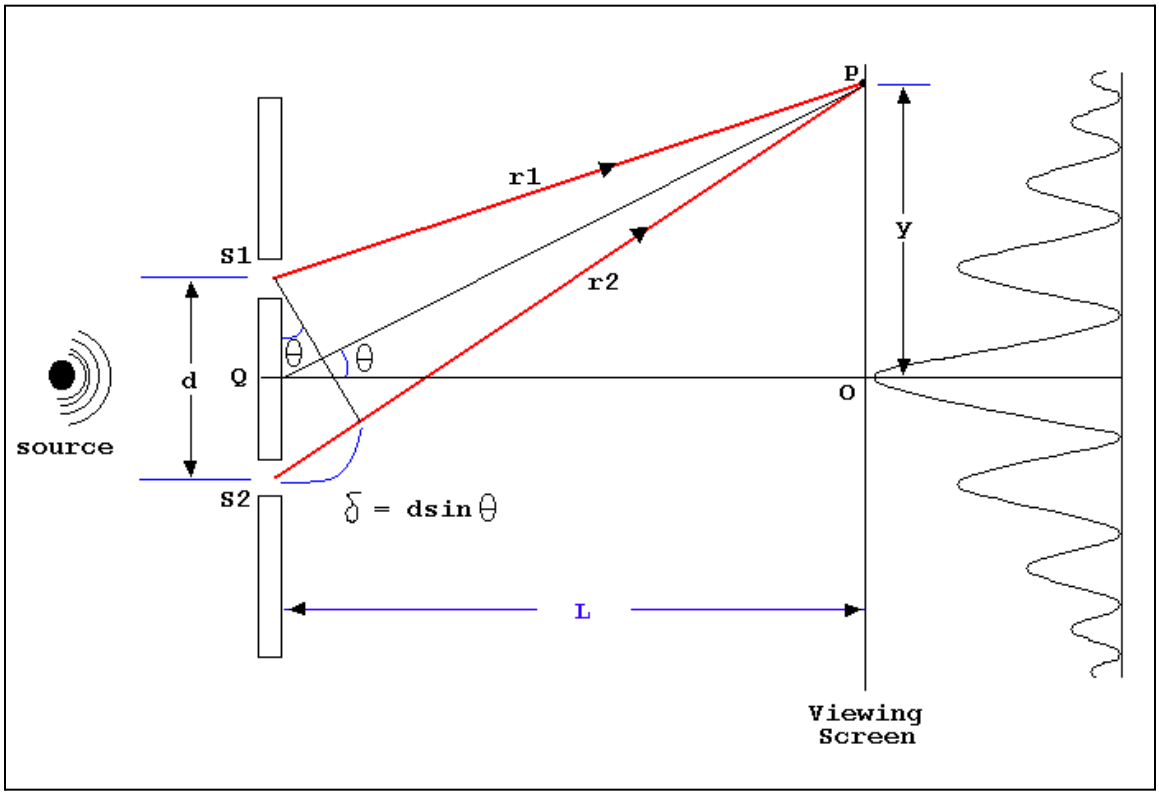
Hydrophone(Teledyne)

How does our vehicle reach the pinger through Hydrophone?

We use the method of TDOA(Time difference of Arrival on the hydrophones). Extensive research is going in the Elec-Subdivision on this for finding various efficient and faster algorithms[1]

We use the principle behind **Young's Double Slit Experiment**. Here's a brief recap of what that experiment says:

Assume your light source is (infinitely) far away from the two slits. Because of this large distance between the slits and the source, the rays incoming from the source are incident in a parallel fashion on both the slits. If we assume the distance between the two slits to be ‘d’, the angle of incidence to be ‘θ’, then:



So we can see that, as the distance L, becomes very large, the two rays r1 and r2 become(almost) parallel, Hence we calculate δ as dsin(θ).

In our scenario, the pinger is located at point **P** in the diagram, We have our hydrophones as S1 and S2, We know ***d*** since we know the time difference of the arrival(calculated by the software subdivision) of the two rays at the 2 hydrophones. Hence, multiplying the time difference with the speed of sound of water, we can measure δ, and hence θ.

So we make use of the Yaw and Pitch capabilities of the vehicle to align the vehicle along the direction of the pinger.

How is TDOA calculated by the software subdivision?

An entire lecture is taken by Andrews from the software subdivision, regarding this topic.

We can get the lecture from this link. And the COLAB file used from this link.

Current problems, faced by using this method:-

1) We need to use DAQ, to sample the data from the hydrophones, which costs us around 1.5 ₹lakh.
2) Software people take around 4 seconds to process 1 million samples of data per ping of the hydrophone(2 seconds).
3) DAQ driver works on Windows, but our entire code runs on Linux OS, so we were using a Virtual machine to interface the 2, adding complexity

Aim of Project

To work on optimizing the pinger task, of ROBOSUB, by decreasing the processing time taken by the Software, and implementing it on FPGA.

Advantages

1) By decreasing the Time difference of Arrival, we can locate the pinger very quickly, hence able to complete all the tasks in a shorter duration of time.
2) Elimination of DAQ. Will reduce the cost significantly. The DAQ comes for around 1.5lakhs rupees, but the FPGA costs s around 30k. Leading to a drop of 1/5th the price.
3) Making the system scalable, would imply that if we want to increase the number of hydrophones in the system, we can easily do it. The same system can be tuned and can be used for Sonars. Various application-specific and independent modules are being created, so we can use them in any acoustic system for our vehicle in the future.

Goal achieved till now

Was able to perform the simulation on FPGA and reduced the time taken by the software from 4 seconds to 0.165 seconds. (A staggering reduction in time by 95.8%).
This shows us that, if we can reduce our dependability on Software and move more towards implementing various tasks on Hardware developing ASICs, we can reduce the time significantly.

Basics and Introduction to FPGA

What is FPGA? Which FPGA are we going to use? Why?
FPGA (Field-Programmable Gate Array) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects.
The basic building block of an FPGA is LuTs(Look Up Table).

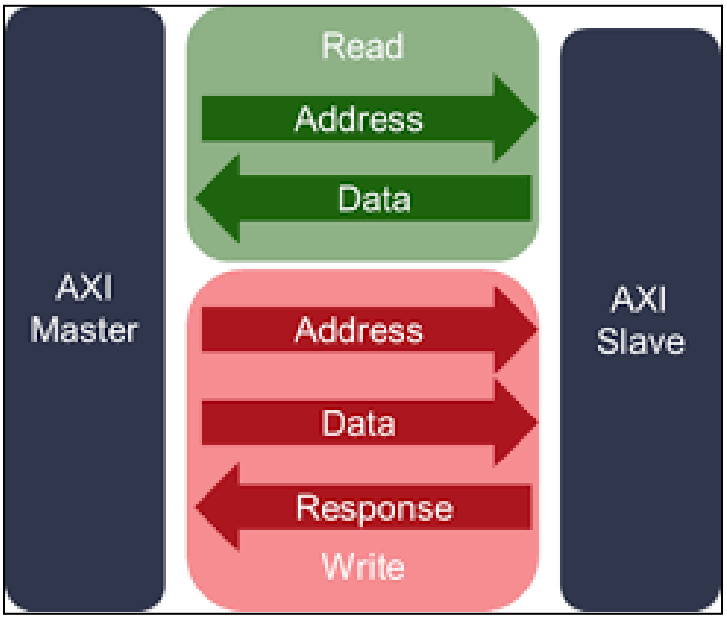
In layman's language, they are devices, which are similar to Arduino, which can be coded to perform various tasks.

We are going to use Xilinx Zybo Z7-20 FPGA in our design.

Xilinx Zynq uses AXI Protocol(Advanced eXtensible Interface) which is the part of ARM Advanced Microcontroller Bus Architecture (AMBA) specification.AXI is simply a bus Standard in which there will be a master and slave, the master generates the control signals to decide when to transfer or allow the slave to send Data, we use multiple masters and slaves by means of AXI interconnect(we will see it).

The most powerful advantage of using FPGA is to unlock the parallel processing power of it.

Tasks that are parallel in nature; FPGAs are much faster than our traditional CPUs.



We are going to use Vivado software and VHDL language for scripting and defining our hardware.

There are 5 compulsory channels:-

- 2 are used for Read transactions
    - read address
    - read data
- 3 are used for Write transactions
    - write address
    - write data
    - write response

A piece of data transmitted on a single channel is called a transfer. A transfer happens when both the VALID and READY signal is high while there is a rising edge of the clock.

Problem statement

Have to design an IP, which would take in the data from FFT and check whether the data is correct or not i.e whether any ping has been detected or not.

Here is the pseudo-code of the algo:-

Consider ‘n’ samples have been forwarded by the FFT IP:-
Find out the peak value in those n samples and the corresponding frequency “*f*”. Let the peak value be denoted as P.

Have to do signal thresholding, for ensuring SNR(Signal to Noise Ratio)<1

Calculate the number of points that have peak value>$\frac{P}{2}$ => n_H

Calculate the number of points that have peak value<$\frac{P}{2}$ => n_L

Calculate the ratio = $\frac{n_H}{n_L} = n_R$

Define a threshold Th

If n_R> Th and *f* must lie a range of desired frequency(for eg 45Khz) then we have successfully detected our ping
Else we have not detected our ping. Wait for the next n samples.

Map Successful detection as 1 and unsuccessful as 0.

Find if the given max value lies in the frequency range.
The output of the FFT IP is going to be a 64-bit vector.
Here 32 bits correspond to the Real part and 32 bits correspond to the imaginary part.

The value = magnitude of the imaginary and real part.

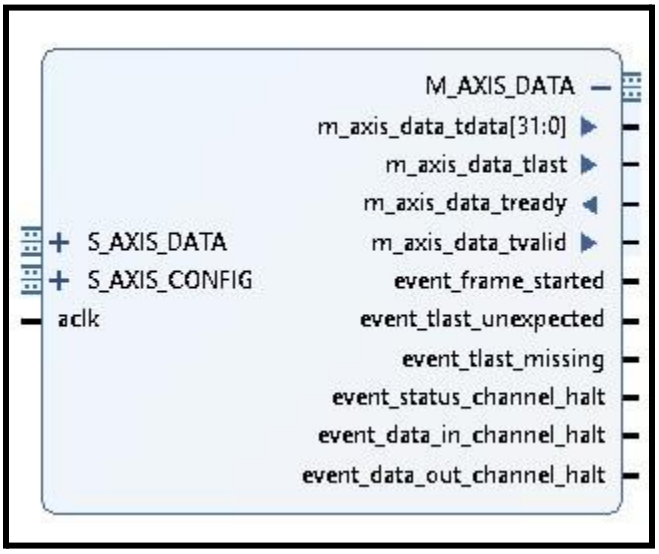
The frequency of the ith sample = $\frac{(i \times F_s)}{N}$

Where i = Index of the point
Fs = Sampling frequency = No of samples we are collecting in 1 second(Currently it’s 2Msps)
N = No of points passed through the FFT block(Thought to be 4096).

In Vivado we would implement the above logic by following the steps present below:-
We would receive the output from FFT.

FFT IP:-

Computes FFT of the n samples provided as an input



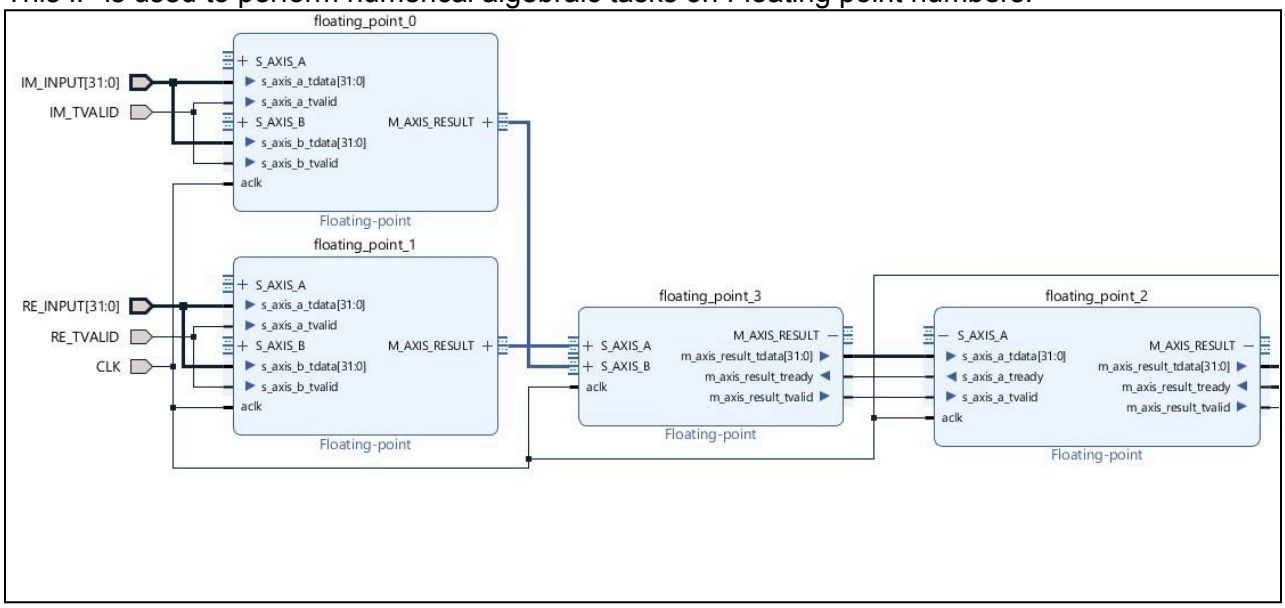
FFT IP

In this, the M_AXIS_TDATA would send the 64-bit data to the slave.
Here [0-31] would be real numbers and [32-63] would be imaginary.

M_axis_data_tlast - Tell the slave that it is sending the last data.
M_axis_data_tready - Slave tells the FFT block that it is ready to receive the data
M_axis_data_tvalid - Tells the slave that the data which it's sending is valid and not garbage value.

Floating Point IP:-

This IP is used to perform numerical algebraic tasks on Floating point numbers.

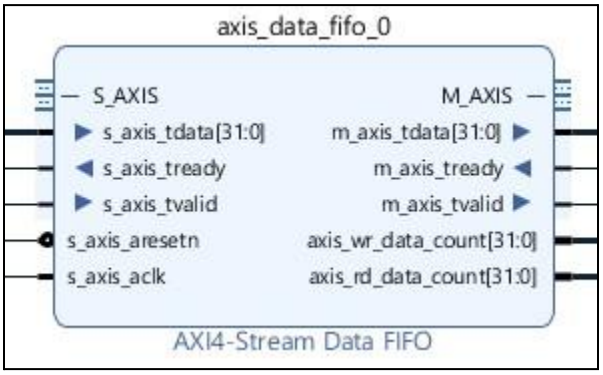


The floating_point_0 and floating_point_1 receive the imaginary(*Im*) and real parts(*Re*) respectively, and it calculated the square of it. Floating_point_3 calculated the sum of the squares, i.e $Im^2 + Re^2$ and then the floating_point_2 will convert the floating-point back to integers by truncating them.

This entire thing would hence send the integral part of $Im^2 + Re^2$.

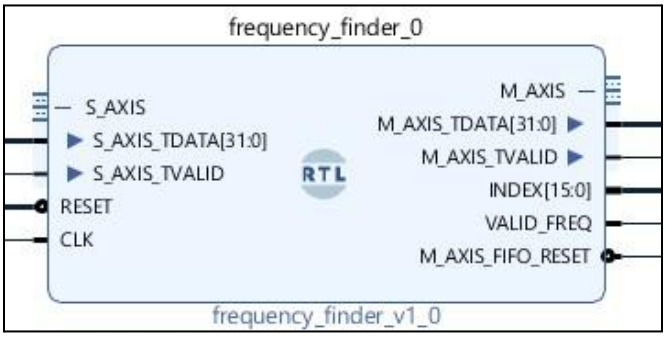
FIFO Storage IP

This IP is used to store the data, and then send the data outwards when required. This behaves exactly like Queue Data structure, where the first data is the first one to go out.



It receives the data from the floating_point_2(integral part of $Im^2 + Re^2$) and saves it for future use. Till then, the frequency_finder IP would find if our desired frequency is present in the data or not. If the frequency is not present, then it would Reset the values of the FIFO, and it would again start storing the data. If the frequency is present it would send the data to fifo_output IP.

Frequency_finder IP



This custom IP is used to detect if the maximum amplitude in the given data by the FFT corresponds to the desired frequency or not.

We find the index here of the maximum amplitude, and then detect the frequency from the formula:-

The frequency of the ith sample = $\frac{(i \times F_s)}{N}$

Where i = Index of the point

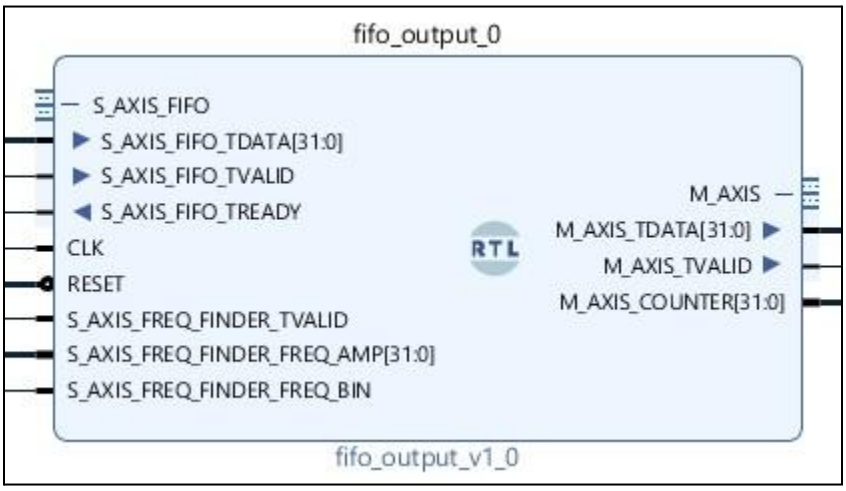
Fs = Sampling frequency = No of samples we are collecting in 1 second(Currently it's 2Msps)

N = No of points passed through the FFT block(Thought to be 4096).

If the frequency is present, we give a pulse to the fifo_output IP to start checking the threshold, else we flush the FIFO_data IP and directly give the output=0 that our bin is not detected.

FIFO_OUTPUT IP

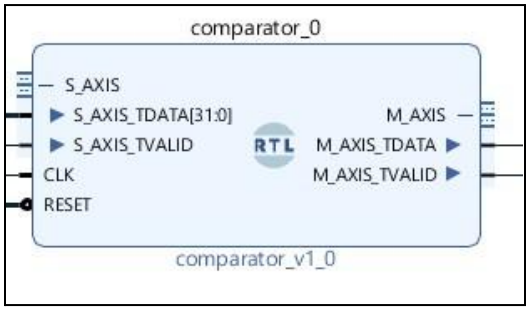
This IP is used to take input from the FIFO storage IP, and then perform the signal thresholding.



This IP would do measure the number of points, which have amplitude > Max amplitude/2, which we had found in frequency_finder IP, and pass it on to the comparator IP

Comparator IP

This IP is used for comparing our number of points>max_amplitude/2, with the threshold, which we have defined earlier. If the condition is satisfied then we have successfully detected the bin.



This IP was tested in behavioral simulation, and now needs to be tested with actual Hardware

Rising Edge IP

This IP was used to detect the start of our ping, after performing cross-correlation with a 45kHz wave.

Sliding window protocol was used, and the ratio of the sum of the 2 windows was taken.

The index where we would achieve the maxima of the ratio was the final output.

FFT IP Block:-

Output ordering:- After doing FFT, we don't get in the natural order of indices, first we get X[0], X[4], X[2]
To get back in normal order just select the natural ordering
https://www.allaboutcircuits.com/technical-articles/xilinx-fft-ip-core-fast-fourier-transform-software-walkthrough/

Data throughput = At what rate is your data incoming
Takes too long to simulate
To check the latency = click on IP block and see the latency
It was around 165us
Scaling factor = 2 bits for each Level. Just shifts the bits to left(multiplied by 2)
https://support.xilinx.com/s/question/0D52E00006iHuXQSA0/fft-scaling-schedule?language=en_US
https://support.xilinx.com/s/article/34171?language=en_US

Update
The data throughput has direct relation resources and inversely with Latency.
Radix 2 burst occupies minimum resources but has max latency
Radix 4 burst a tremendous amount of resources

Flowchart for better understanding:-
https://lucid.app/lucidchart/0c6e8c74-a7b1-4cd7-a35b-3fbfd425bb64/edit?shared=true&page=0_0#

Links used:-
FFT theory
https://forums.xilinx.com/t5/Design-Entry/How-to-get-the-magnitude-of-FFT-v9-0/td-p/795383

https://www.realdigital.org/doc/40151bc8902625f0bf2af6f333126084#:~:text=Recall%20the%20Axi%20handshake%20procedure,clock%20edge%20completes%20the%20handshake.

https://forums.xilinx.com/t5/High-Level-Synthesis-HLS/How-do-I-split-my-incoming-complex-axi-stream-so-I-can-do-real/td-p/663183

Slice IP
https://forums.xilinx.com/t5/AI-Engine-DSP-IP-and-Tools/Basic-AXI-Stream-Slicing-and-Dicing/td-p/390223

Things learned:-
There are many different types of RAM
Sram, Dram. We also have distributed Ram and Block Ram.
Distributed Ram is used in LUT(Lookup tables).
BRAM are small chunks of memory, that we have on the PL side, which is different from the logical block(slice) in PL where other elements (Multiplexers/LUTs, FlipFlops) are present. It is made so that we don't waste a Logical Block for storing Data.
Functionally a BRAM is used for storing Blocks of Data, It can also be used as a FIFO between PL and PS blocks and others only when we use it with help of a Controller which is responsible for synchronizing reading and writing data in BRAM.
So a BRAM is just some bunch of Raw space for Bits and the BRAM controller is the Brain which uses the Space for our Needs.

AXI Interconnect - Bus which connects various masters and slaves. Consist of a memory map for each slave.

AXI DMA(Direct Memory Access) - It is used as a Data Mover. Used to read/write from AXI stream block to Memory. Has Scatter gather feature which is used to decrease the load on the CPU. Increases the efficiency of the PS to perform more instructions.
AXI Central DMA - Only transfers data from one memory to another.

We generally make design the block according to our own needs. The collection of blocks is known as IP(Intellectual Property). The final system is the collection of IPs.
For designing an IP, we generally do give access to the CLKs and interconnects, by which exterior IP can access the blocks present inside it.

System Description

	Arty A7-35T	Zybo Z7020
LUTs	5200	53200
BRAMs	225KB	630KB
FFs	29200	106400

Bin detection
Utilization;-

- LUTs - 648
- Slice registers - 1106
- BRAM - 2
- DSP - 6

Correlation
Utilization:-

- BRAM - 8