1. **Introduction**

In number theory and computer science, the partition problem, or number partitioning, is the task of deciding whether a given multiset ‘S’ of positive integers can be partitioned into two subsets S1 and S2 such that the sum of the numbers in S1 equals the sum of the numbers in S2. This is one of the very time-consuming problem in the realm of computer science. As the set gets larger the execution time gets extremely longer. I tried to write my code to solve this problem in Python and it took me 15 minutes to solve the array set of size 25. So, how can we make it faster? Well, there are couple of ways to make it faster. We can certainly optimize the algorithm, use multicore CPUs, use GPUs or implement this in FPGA. Thus, in this tutorial our main focus is to implement this solution in FPGA.

***Note: To make this tutorial short, the detail algorithm of the partition problem is not described in this tutorial. Video for this tutorial is found*** [***here***](https://www.youtube.com/watch?v=0_A0uHv1igU&feature=youtu.be)***.***

1. **Importing the overlay into Python**

Just like there are software libraries, there are hardware libraries which can be reconfigured, and they are called overlays. The required overlay for this project is provided inside the GitHub repository [here](https://github.com/pratik-stha/PartitionProblemUsingFPGA/blob/master/PartitionHLS.cpp).



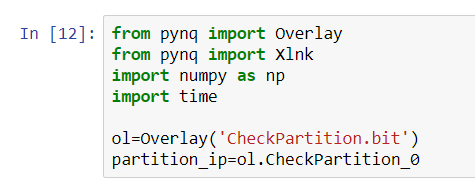
Power up the PYNQ board with USB and connect it with the machine through ethernet. Boot up the PYNQ board. When the LEDS flash blue, it indicates that the board has been successfully booted and is ready to work. Refer to [the official tutorial](https://pynq.readthedocs.io/en/v2.0/getting_started.html) if there is any problem connecting the board.

We are interested in two files. One is \*.BIT file and other is \*.HWH file. Dowbload the “**CheckPartition.hwh**” and “**CheckPartition.bit**” files from the repository and store it somewhere safe. Now navigate to [\\pynq\xilinx\jupyter\_notebooks](file:///\\pynq\xilinx\jupyter_notebooks) and copy both of these files and paste it here. Please make sure you can successfully access the PYNQ board first.

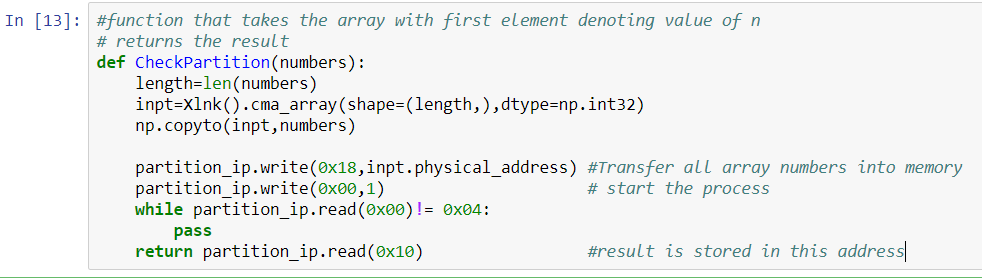
Follow the steps below to import the overlay and use it.

1. Open the Jupyter Notebook through PYNQ remote access using the IP address.
2. Create a python 3 file (click on **New** > **Python 3**) at same location where we have copied the two files.
3. Write down the following codes as shown in image below then Run it.

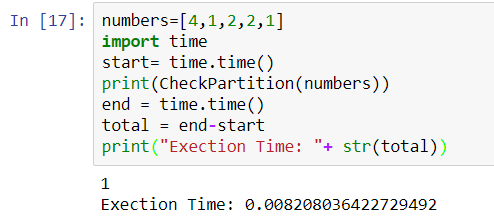
***Note: You may use the code from “PartitionProblem.ipyb” from the repository.***



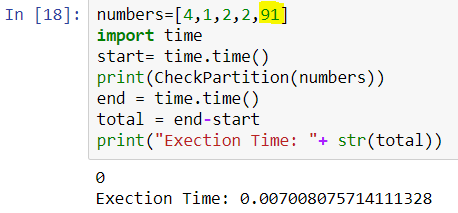
1. After successful execution, create one more cell underneath this cell and write down the following code. Then Run it.



1. Create one more cell underneath this cell and write down the following code which calls the previous function and sends the parameter. We are now testing if our overlay actually worked or not.



1. Here, the variable ‘**numbers**’ is the array of which the first element ‘**4**’ denotes there are altogether 4 numbers that needs to be partitioned. All the numbers are the first element are the actual numbers that needs to be partitioned. After execution of this cell, the function should return the value of 1 indicating that the given set can be partitioned with the total execution time. If it returns 0 then it means the given set cannot be partitioned.
2. Go ahead and try to change the number as shown in figure and see what happens.



Congratulations !! we have successfully implemented the overlay and executed this program.