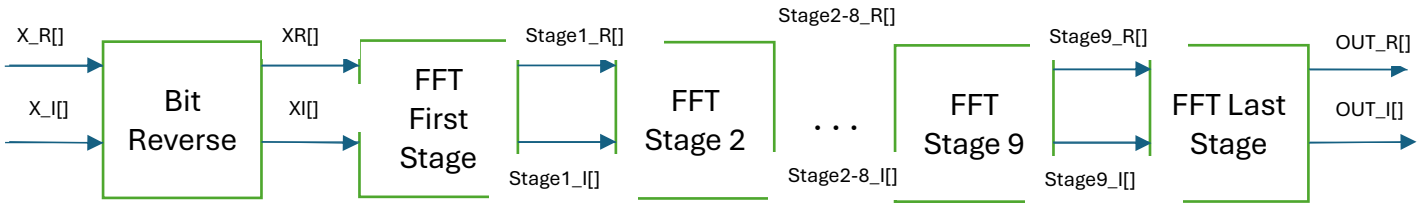
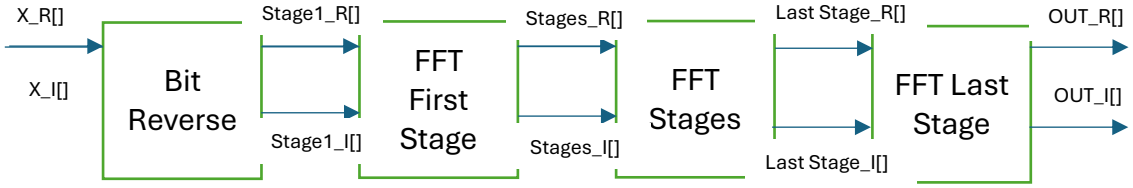
Assignment 4: Fast Fourier Transfer

My best FFT design essentially divides a 1024 point FFT into 10 separate tasks and has optimizations throughout the code. The overall architecture of the code is shown below.



To get to this design, I first started with a 4 task implementation. I used the inputs from the FFT function, X_[R] and X_[I] , as inputs to the bit reversal function. At this point, when I perform a bit reversal, I saved the values to the same input arrays. Now that X_R[] and X[I] are updated, I input them into the first stage of the FFT computation. The results of the First FFT computation is then input into the "FFT Stages" calculation. Finally the FFT Stages calculation are input to the "FFT Last Stage" and the FFT computation is complete.
At this point, my code is looking like the diagram below:



To begin my optimization, the first thing I did was make use of the dataflow directive. #pragma HLS dataflow is used to enable the task pipelining of my 10 task FFT. To do this, I made updates to the 4 task architecture as follows:

1. I declared 2 new DTYPE arrays XR[SIZE] and XI[Size] which will store results of the bit reversal call on X_[R] and X_[I]. With a new set of input arrays, I had to modify first FFT stage inputs, as well as the implementation of the bit reversal functions.
2. I replaced the "FFT Stages" task with 9 new individual functions. Essentially it is the same code as FFT Stages, but specific to the stage number (2 through 9). This also required new function declarations and updated calls in the code.

The next optimization I did was on the bit reversal function. To optimize the bit reversal, I chose to manually unroll the for loop by a factor of 2. The intent of this optimization is to improve the overall performance of the code by executing the bit reversal operations in parallel.

Next considerations:
I have not finished exploring all the possible ways to optimize my code, but my next steps would be look into the pipeline pragma and more loop unrolling.