Lab A - FFT Transform

Team 10 陳炫均 林芎甫 賴宥儒

Outline

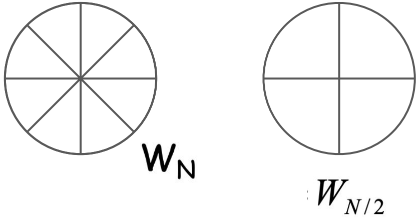
- Inroduction
    - Radix-2 FFT
- Optimization Methods
    - Address conflict
    - Sin, Cos Look-up Tables
    - Complex Multiplication
- Evaluation
    - FFT_SIZE
    - Fixed point

Introduction

Radix-2 FFT

$$X[k]=\sum_{n=0}^{N-1}x[n]W_N^{kn}, \qquad k=0,1\ldots,N-1$$

$W_N = e^{-j2\pi/N}$ is a root of the equation $W^N=1$.



$$=\sum_{n\ \text{even}}x[n]W_N^{nk}+\sum_{n\ \text{odd}}x[n]W_N^{nk}$$

$$=\sum_{r=0}^{(N/2)-1}x[2r]W_N^{2rk}+\sum_{r=0}^{(N/2)-1}x[2r+1]W_N^{(2r+1)k}$$

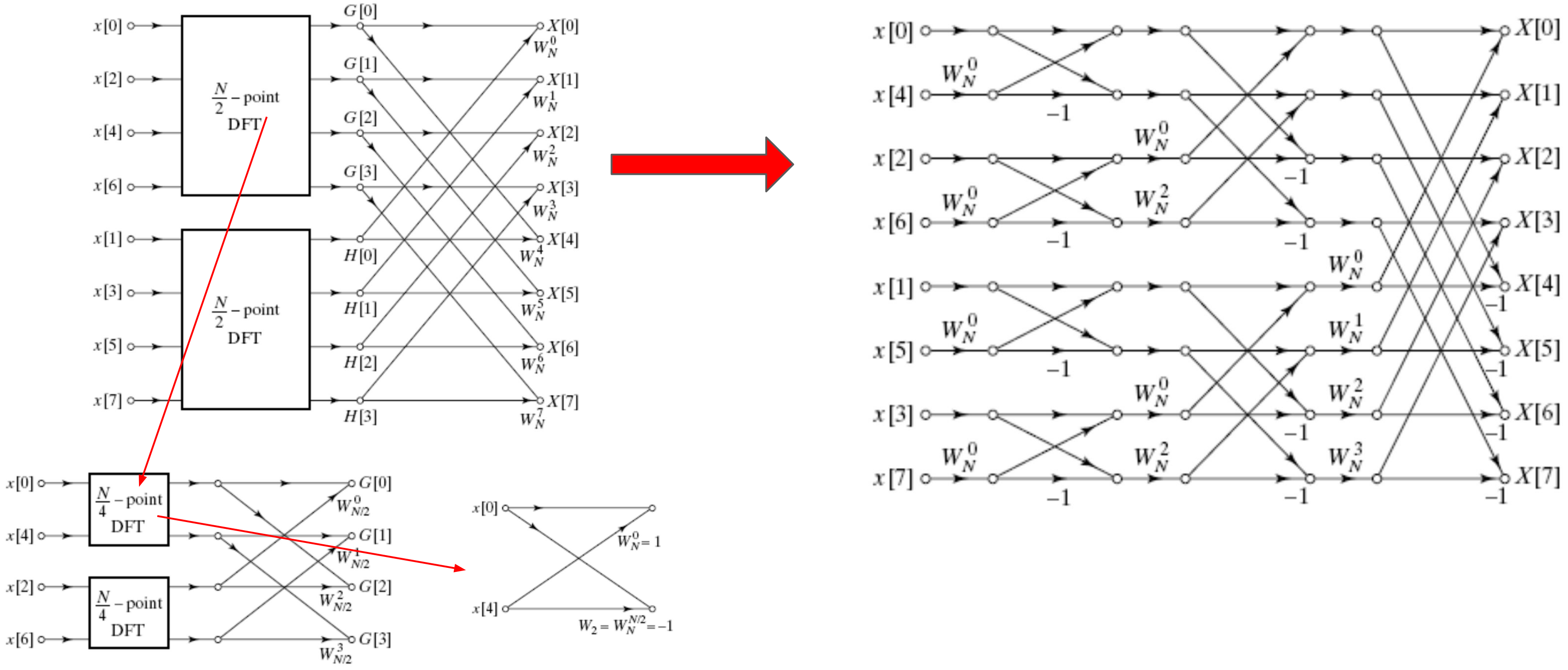
key property: $W_N^2=e^{-2j(2\pi/N)}=e^{-j2\pi/(N/2)}=W_{N/2}$

$$=\sum_{r=0}^{(N/2)-1}x[2r](W_{N/2})^{rk}+W_N^k\sum_{r=0}^{(N/2)-1}x[2r+1](W_{N/2})^{rk}$$

$$=G[k]+W_N^kH[k]$$

- G[k]: the (N/2)-point DFT of the even numbered points of the original sequence
- H(k): the (N/2)-point DFT of the odd-numbered points of the original sequence.

Radix-2 FFT



x[0] x[2] x[4] x[6] x[1] x[3] x[5] x[7]
N/2 – point DFT
N/2 – point DFT
G[0] G[1] G[2] G[3] H[0] H[1] H[2] H[3]
X[0] X[1] X[2] X[3] X[4] X[5] X[6] X[7]
W_N^0 W_N^1 W_N^2 W_N^3 W_N^4 W_N^5 W_N^6 W_N^7
N/4 – point DFT
N/4 – point DFT
$W_{N/2}^0$ $W_{N/2}^1$ $W_{N/2}^2$ $W_{N/2}^3$
$W_N^0 = 1$
$W_2 = W_N^{N/2} = -1$
x[0] x[4] x[2] x[6] x[1] x[5] x[3] x[7]
–1

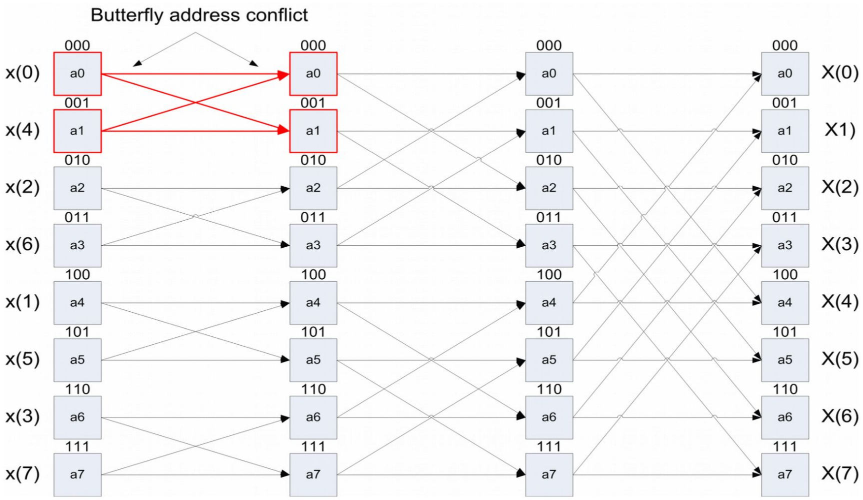
Optimization Methods

Evaluate the effectiveness of optimizations

- address conflict
- lookup table
- complex multiplication

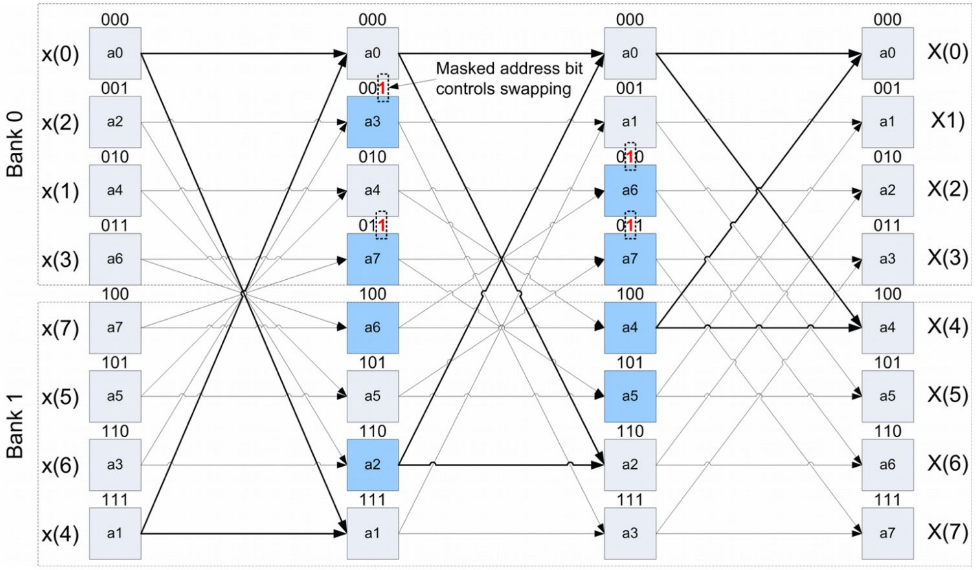
Address conflict

- In place implementation
  - single array/memory to compute each stage of FFT
- Mapped to single port memories
  - e.g., need to read and write to `a0`



Address conflict

- Solution: use two single port memories
  - requires reordering of the input data
- Read a0, a1, then write a0, a1



Force Array Store into BRAM or URAM Implementations

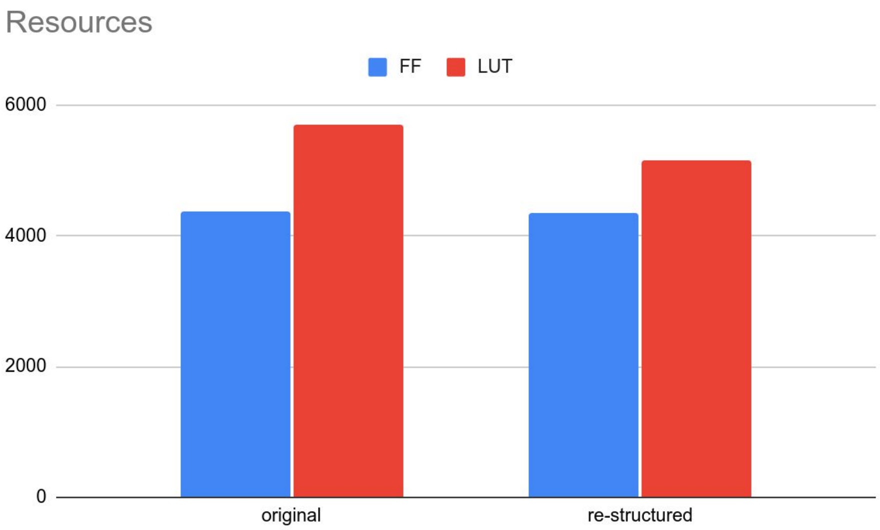
- BRAM:

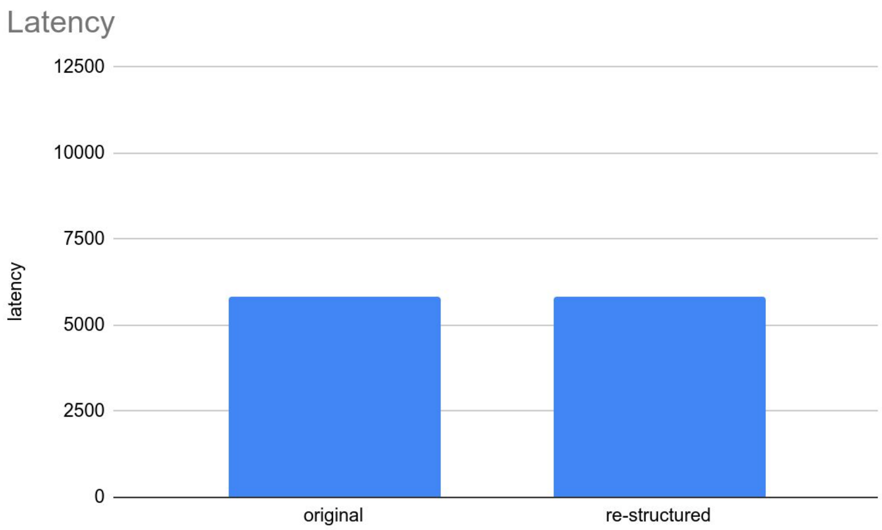
  Add m_axi interface directives for "in" array in HLS to store the array in BRAM instances.

- URAM:

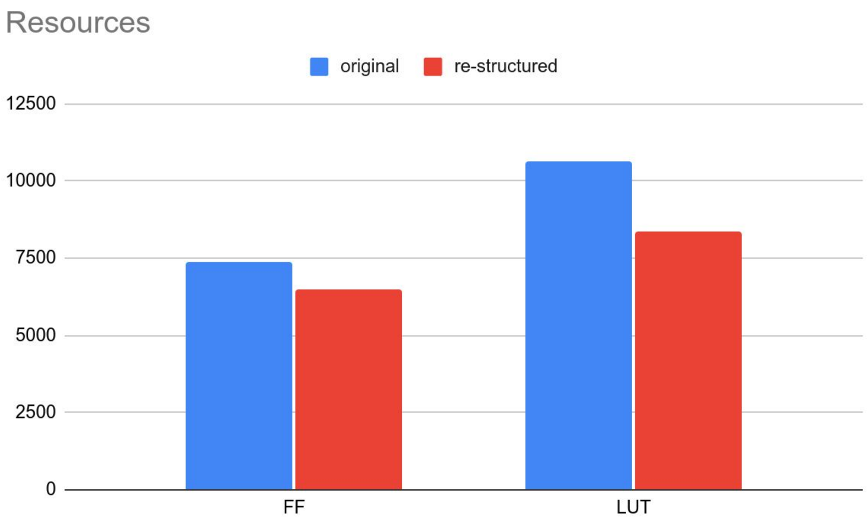
  There is no directive to force URAM implementation through HLS for the interface. Change the MEM_STYLE attribute in RTL and re-run RTL synthesis.

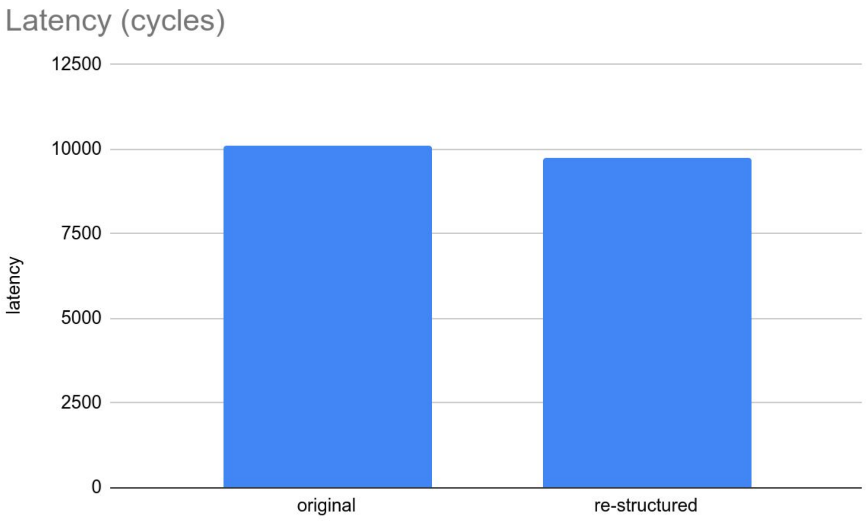
Optimizations - Address Conflict (without force mapping)





Optimizations - Address Conflict (force mapping to BRAM)

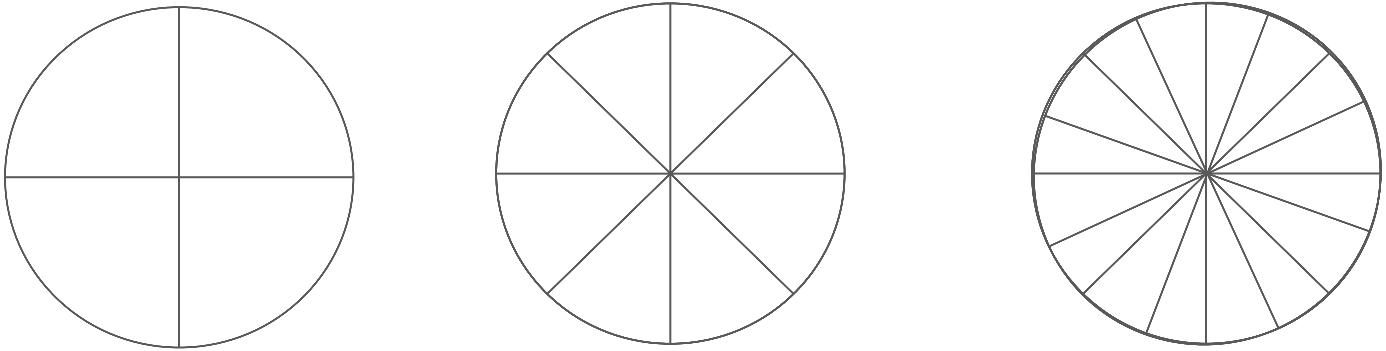




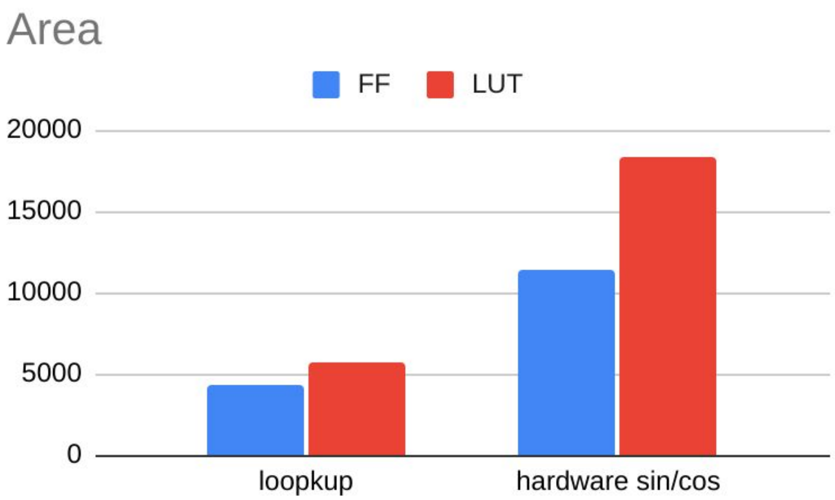
post-P&R clock period (ns)	**9.416**	**9.326**

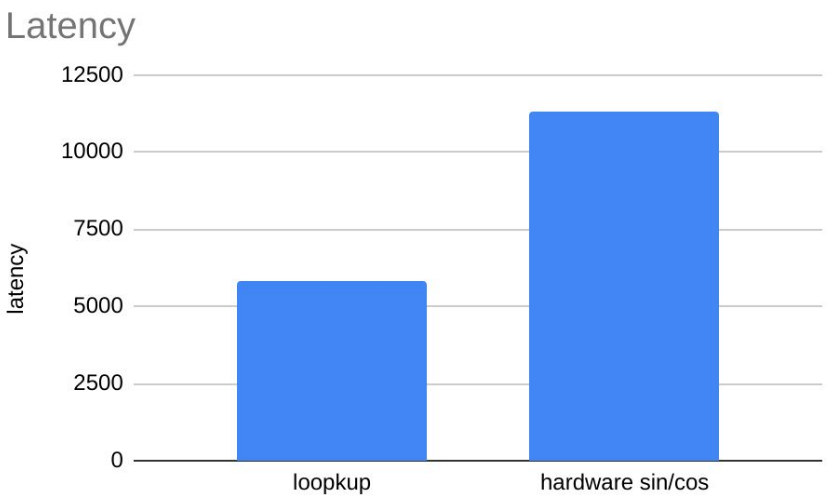
Sin, Cos Look-up Tables

- sin, cos functions are hard to synthesize
- the input was decided when configured
  - simply precompute and read them from a look-up tables



Optimizations - Lookup Table

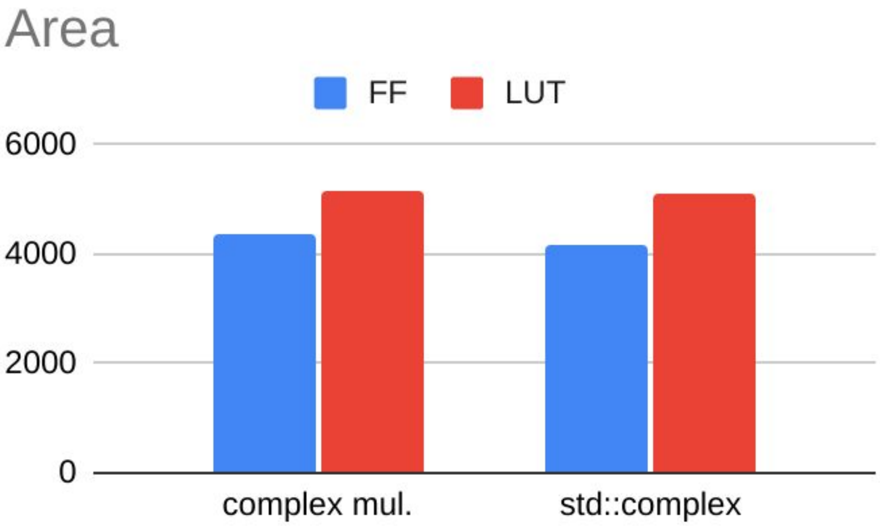


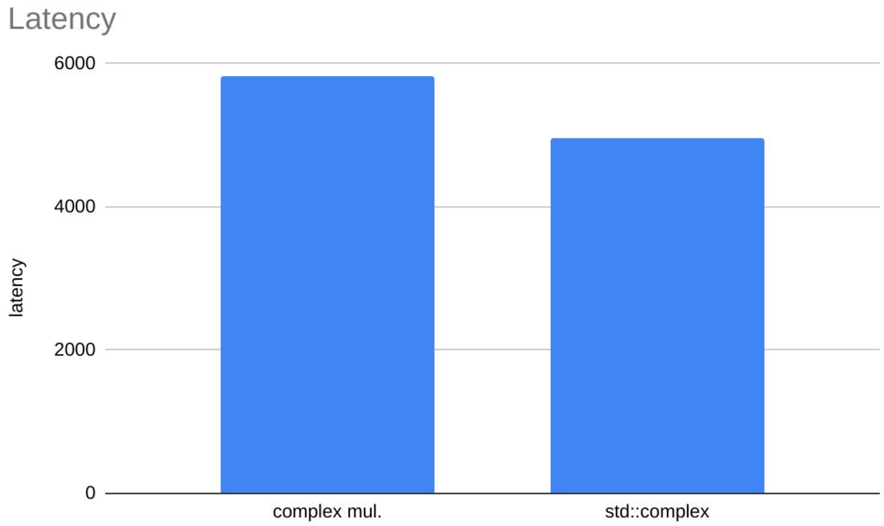


Complex Multiply

- `(a+bi)*(c+di) = (ac-bd)+(ad+bc)i`
    - 4 complex number multiplications
- `e = a*(c-d)`
    - `real part: d*(a-b)+e`
      ```
                      = ad-bd + ac-ad
                      = ac-bd
      ```
    - `imaginary part: c*(a+b)-e`
      ```
                      = ac+bc - (ac-ad)
                      = bc+ad
      ```
    - 3 complex number multiplications

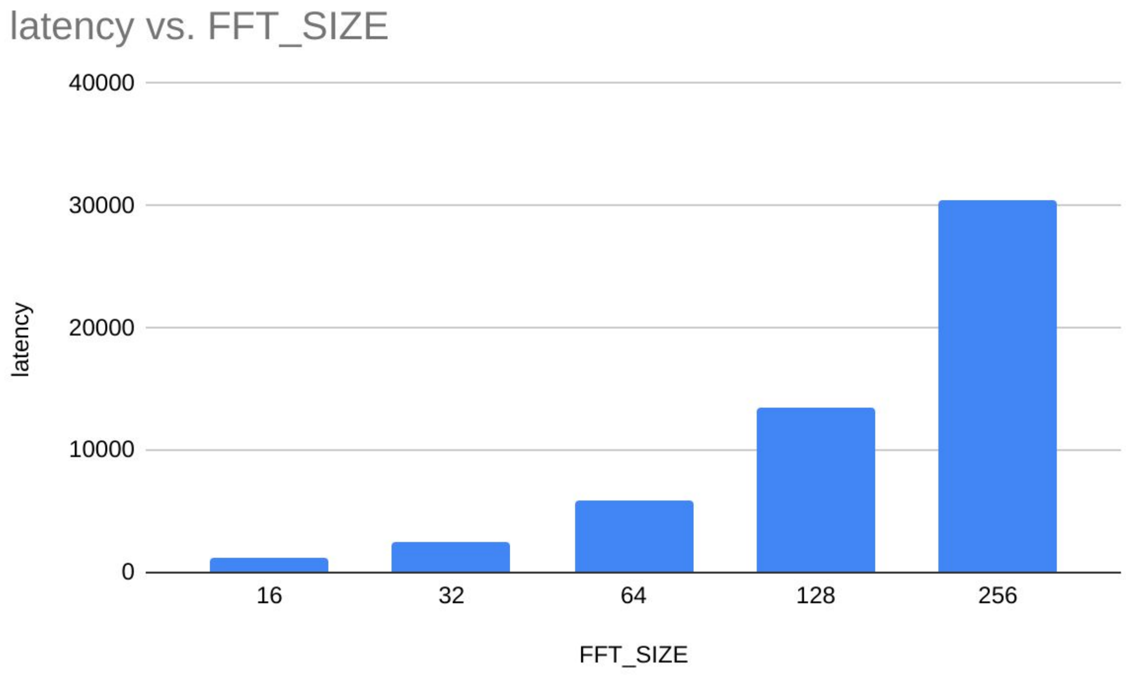
Optimizations - Complex Multiplication





Evaluation

FFT_SIZE lantency

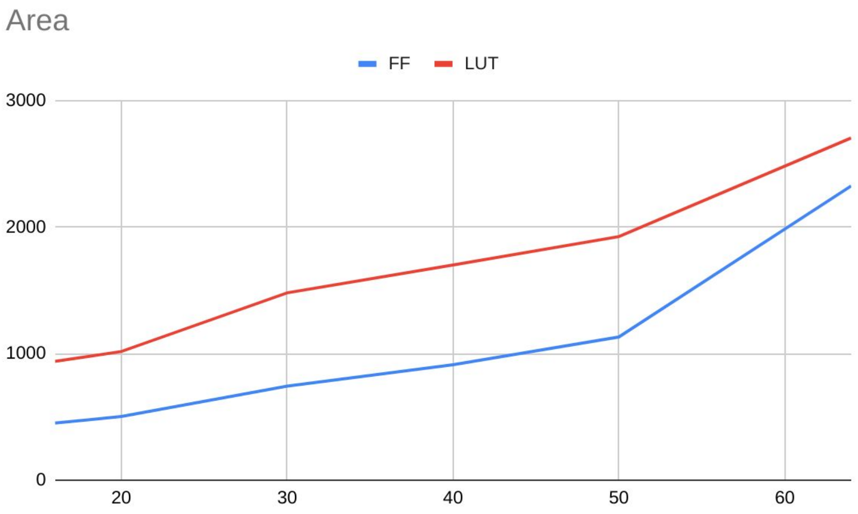


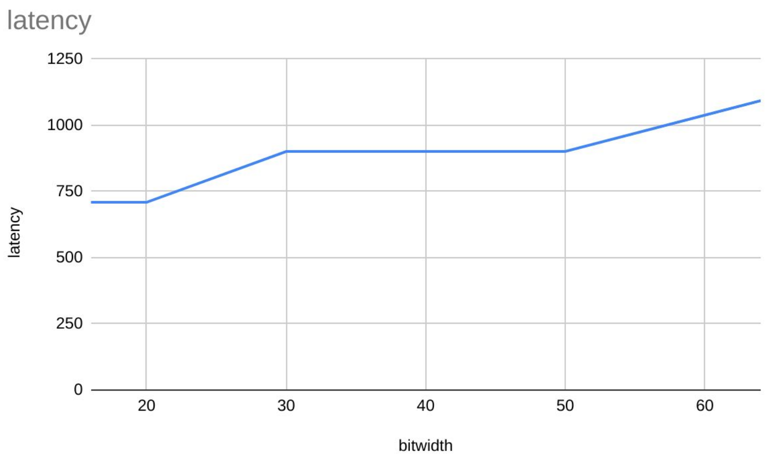
FFT_SIZE lantency

- We pick FFT_SIZE = 64 for later experiment
  - It is time-consuming to simulate large RTL design

Using different bit width fixed floating point

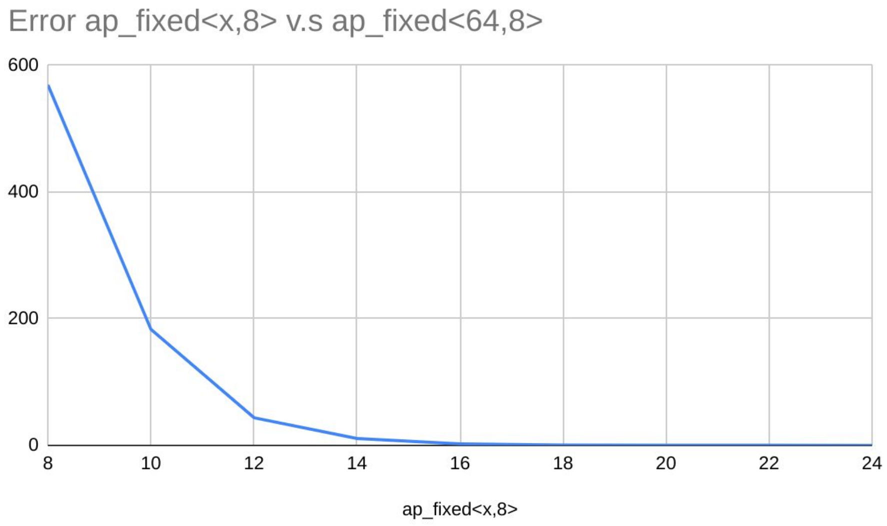
- Area and Lantency increase when using higher precision.

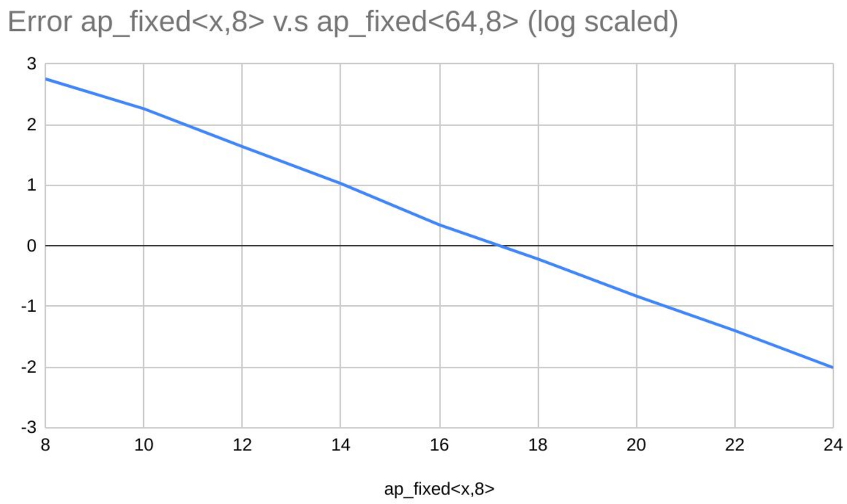




Using different bit width fixed floating point

- Fixed software FFT solution using high-precision floating point `ap_fixed<64, 8>`
- RTL hradware FFT solution using low-precision floating point `ap_fixed<x, 8>`





Implmentation Details

- vivado innate fixed types are not as useful as mentor's implementation
    - ap fixed types are failed to support hls_x_complex types

Github repo

- https://github.com/eee4017/HLS_Lab_A