Implementation and Optimization of Multi-dimensional Real FFT on ARMv8 Platform

小组成员：陆文博、张敦博、李卓倩
指导老师：龚春叶、甘新标、杨博

目录

CONTENTS

Part 1 Related Work

01. Related Work

Implemented real DFT kinds: R2HC, HC2R, DHT, DCTI-IV, DSTI-IV

Challenges:

- Diversity of real DFT brings difficulties
- Real DFT depends on complex DFT

Our work:

- Summarize and abstract real DFT optimization algorithms into an unified two reduction form.
- Implement and optimize 1D Cooley-Tukey complex FFT algorithm.
- Propose a cache-aware algorithm for ARMv8 platform for 2D real DFT

Part 2 Optimization of 1D Real DFT

02. Introduce of DFT

DFT transform this sequence into frequency domain:

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{nk}$$

DFT can be expressed as a matrix multiplication between input vector and a pre-defined DFT matrix

$$X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W_N^1 & W_N^2 & W_N^3 & W_N^4 \\ 1 & W_N^2 & W_N^4 & W_N^6 & W_N^8 \\ 1 & W_N^3 & W_N^6 & W_N^9 & W_N^{12} \\ 1 & W_N^4 & W_N^8 & W_N^{12} & W_N^{16} \end{bmatrix} \times x$$

02. Introduce of DFT

Two reduction approaches:

- Reduction from real DFT to halved complex DFT(real reduction)
- Reduction from real DFT to halved real DFT(complex reduction)

Table 1. Relation between all real DFT kinds and reduction approach

Real DFT kind	R2HC	HC2R	DHT	DCT I	DCT II	DCT III	DCT IV	DST I	DST II	DST III	DST IV
Real reduction	No	No	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Complex reduction	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

02. Complex Reduce

Reduce real DFT into complex transform with half size and split result from complex transform’ s output.

$$F_r = \sum_{l=0}^{\frac{N}{2}-1} f_l W_{\frac{N}{2}}^{rl} \quad G_r = \sum_{l=0}^{\frac{N}{2}-1} g_l W_{\frac{N}{2}}^{rl}$$

Extract Fr and Gr from result of a complex transform of only half size

① achieve a transform of only half size

$$Y_r = \sum_{l=0}^{\frac{N}{2}-1} (f_l + j g_l) W_{\frac{N}{2}}^{rl} = F_r + jG_r$$

02. Complex Reduce

Reduce real DFT into complex transform with half size and split result from complex transform' s output.

$$F_r = \sum_{l=0}^{\frac{N}{2}-1} f_l W_{\frac{N}{2}}^{rl} \quad G_r = \sum_{l=0}^{\frac{N}{2}-1} g_l W_{\frac{N}{2}}^{rl}$$

Extract Fr and Gr from result of a complex transform of only half size

② split Fr and Gr from Yr

$$F_r = \frac{1}{2}(Y_r + \overline{Y}_{\frac{N}{2}-r}) \quad G_r = \frac{j}{2}(\overline{Y}_{\frac{N}{2}-r} - Y_r)$$

02. Complex Reduce

Algorithm of Complex Reduce

ComplexReduction(Input x, Output X, Direction dir, kind k)

Complex DFT(x,Y, Direction)
Compute X[N/2], X[0];
for each $i \in [1, N/4]$ **do**
    $Y_r \leftarrow Y[i]$
    $Y_{nr} \leftarrow \overline{Y[\frac{N}{2}]}$
    $Fr \leftarrow Y_r + Y_{nr}$
    $gr \leftarrow j * (Y_{nr} - Y_r)$
    $Gr \leftarrow gr * W_N^r$
    Retrive real part and imginary part from Fr Gr
    $X[r] \leftarrow$ Reconstruct real part and imaginary part based on real DFT type(k).
end for
return;

02. Real Reduce

- DCT/DST I are solved by Algorithm 1
- DCT/DST III is reduced to another real DFT with half size

$$DCTII: X_k = x_0 + (-1)^k x_{N-1} + 2\sum_{n=1}^{N-2} x_n cos(\frac{\pi nk}{N-1})$$

$$DCTIII: X_k = x_0 + (-1)^k x_{N-1} + 2\sum_{n=1}^{N-2} x_n cos(\frac{\pi nk}{N-1})$$

$$X_k = 2RealPart[W_{2N}^{k} \sum_{n=0}^{N/2-1} v_n W_{N/2}^{nk})]$$

$$v_k = \frac{2}{N} \sum_{n=0}^{\frac{N}{2}-1} V_n W_{N/2}^{-nk}$$

$$V_k = \frac{1}{2} W_{2N}^{-k} [X_k - jX_{N-k}]$$

02. Real Reduce

- DCT/DST IV are divided into sub-transform of DCT/DST III

$$DCTIV : X_k = 2\sum_{n=0}^{N-1} x_n cos(\frac{\pi(n+1/2)(k+1/2)}{N})$$

N here is input size of DCT/DST, which is near half of logical transform size.
if k equals DSTIV/DCTIVs **then**
$dct-input/dst-input \leftarrow x$.
Algorithm 2(dct-input, X_1, backward, dctIII);
Algorithm 2(dst-input, X_2, backward, dstIII);
for each $i \in [0, N/2]$ **do**
$X[i] \leftarrow X_1[i] * sptws[i].r + X_2[i] * sptw[i].i$
$X[i+N/2] \leftarrow X_1[N/2-1-i] * sptw[N/2-1-i].i + X_2[N/2-1-i] * sptw[N/2-1-i].r$
end for
return;
end if
if k equals DST II **then**
$xm[i] \leftarrow x[i] * (-1)^{i \bmod 2}$ i $from$ 0 to N
Algorithm2 (xm, X, forward, dctII);
end if
if k equals DST III **then**
$xm[i] \leftarrow x[N-1-i]$
Algorithm2 (xm, X, forward, dctIII);
end if
if k equals DCT II **then**
for each $i \in [0, N/2-1]$ **do**
$xm_i \leftarrow x[2i]$
$xm_{N-i} \leftarrow x[2i+1]$
end for
end if
if k equals DCT III **then**
$xm[i] \leftarrow x[N-1-i]$
end if
Algorithm1 (xm, X, dir, kind);

Part 3 Optimization of 1D Complex DFT

03. Butterfly Network Optimization

Using cooley-Tukey FFT algorithm to implement and optimize a high performance 1D complex DFT library.

Approaches in implementing cooley-Tukey FFT algorithm:

- Decimation-in-time,DIT
- Decimation-infrequency,DIF

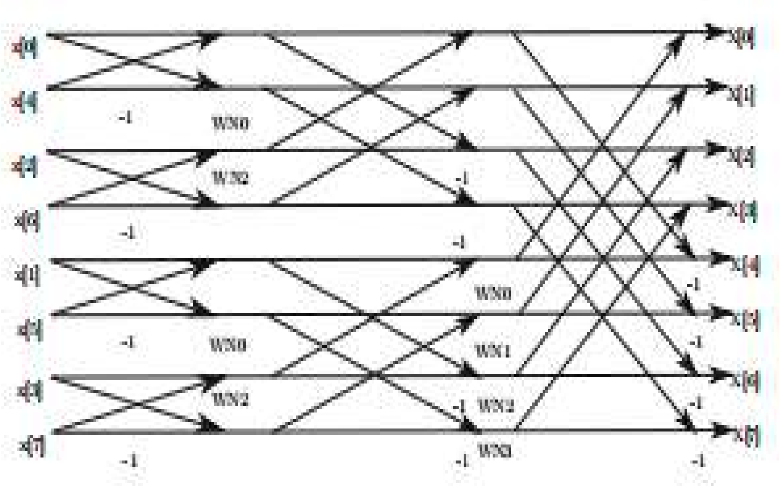


Fig. 1. DIT radix-2 butterfly network with 8 points

03. Butterfly Network Optimization

Using cooley-Tukey FFT algorithm to implement and optimize a high performance 1D complex DFT library.

Approaches in implementing cooley-Tukey FFT algorithm:

- Decimation-in-time,DIT
- Decimation-infrequency,DIF

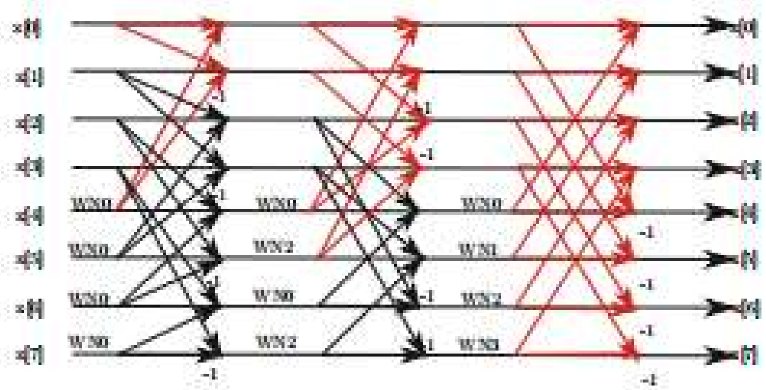


Fig. 2. Unified network with 8 points. (Color figure online)

03. Butterfly Network Optimization

Advantages of DIF

- No need to be bit-reversed
- Simd-friendly
- Mix-radix friendly

Spliting the first stage out of the general computation network

- Twiddles used in first stage is constant 1
- The layout of first stage output is different from other stags

03. Butterfly Computation Optimization

- ✓ Given x_0,x_1, x_2, x_3, x_4 as kernel input
- ✓ Given X_0, X_1, X_2, X_3, X_4 as kernel output

$$X_0 = x_0 + x_1 + x_2 + x_3 + x_4$$

$$X_1 = x_0 + W_5^1 x_1 + W^2 x_2 + W^{-2} x_3 + W_5^{-1} x_4$$

$$X_2 = x_0 + W_5^2 x_1 + W_5^{-1} x_2 + W_5^1 x_3 + W_5^{-2} x_4$$

$$X_3 = x_0 + W_5^{-2} x_1 + W_5^1 x_2 + W_5^{-1} x_3 + W_5^2 x_4$$

$$X_4 = x_0 + W_5^{-1} x_1 + W_5^{-2} x_2 + W_5^2 x_3 + W_5^1 x_4$$

The original computation steps

Merge the same terms

$$X_0 = x_0 + (x_1 + x_4) + (x_2 + x_3)$$

$$X_1 = x_0 + (A - B) \; X_2 = x_0 + (C + D)$$

$$X_3 = x_0 + (C - D) \; X_4 = x_0 + (A + B)$$

$$A = (x_1 + x_4) * W_5^1.r + (x_2 + x3) * W_5^2.r$$

$$B = [(x_1 - x_4) * W_5^1.i + (x_2 - x3) * W_5^2.i] * (-j)$$

$$C = (x_1 + x_4) * W_5^2.r + (x_2 + x3) * W_5^1.r$$

$$D = [(x_1 - x_4) * W_5^2.i - (x_2 - x3) * W_5^1.i] * j$$

03. Butterfly SIMD Optimization

ARMv8 supports both 32-bit execution status and 64-bit status

① Inter-Butterfly Parallelization

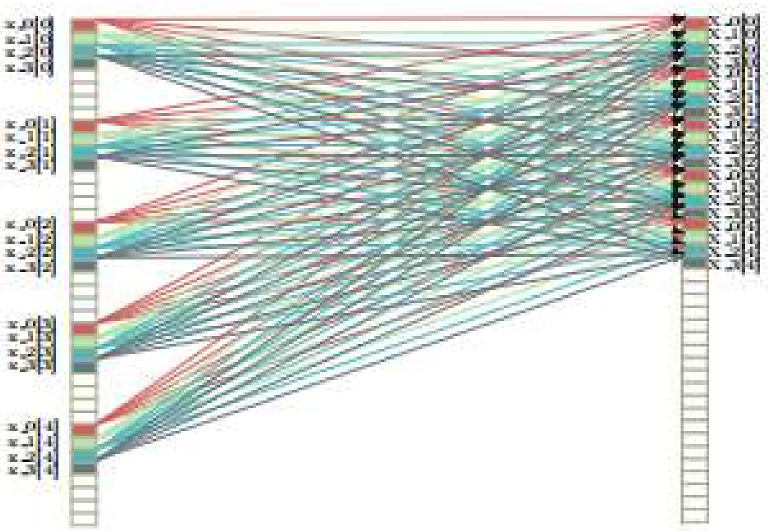


Fig. 3. Parallelization of 4 butterfly computations when radix is 5 (Color figure online)

03. Butterfly SIMD Optimization

ARMv8 supports both 32-bit execution status and 64-bit status

② Assembly Instruction Selection

- Through zip1 instruction to rearrange output elements' order within the first stage.
- Use ld2, faddq, st2 and other instructions to efficiently do complex number arithmetic operation
- Apply fmla/fmls properly to gain better computational performance.

③ Reuse of Vector Register

④ Optimization for Small Scale

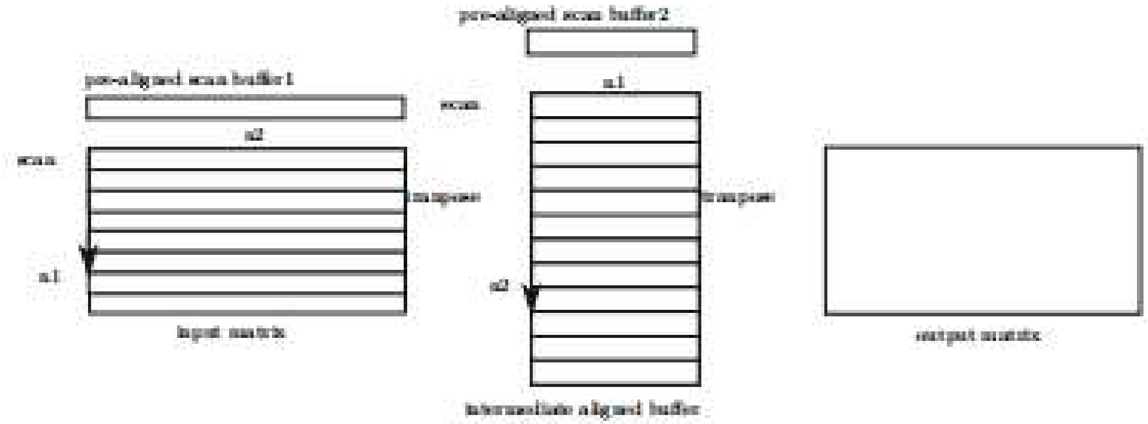
Part 4 Optimization of 2D Real DFT

04. Consideration of 2D Real DFT Optimization

Following techniques to improve cache performance

① Cache block Method
② Memory alignment

Procedure of 2D Real DFT Optimization



Part 5 Experimental Results and Analysis

05. Test Platform and Comparison Baseline

- CPU: ARM Cortex A57, 2.1 GHZ.
- Operation System: Ubuntu 15.04 with main memory size is 64 GB
- Comparison Baseline: FFTw 3.3.7
- Performance metric: Gflops = Floats Operations/Wall time.

05. Experimental Results

Our 1D float transforms outperform FFTw3.3.7 7 significantly with even greater advantage when input size is becoming large.

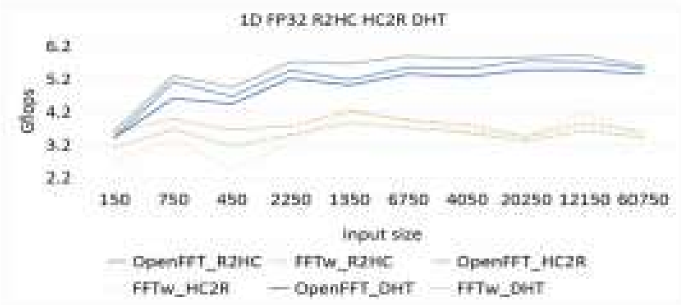


Fig. 5. 1DFP32 R2HC/HC2R/DHT (Color figure online)

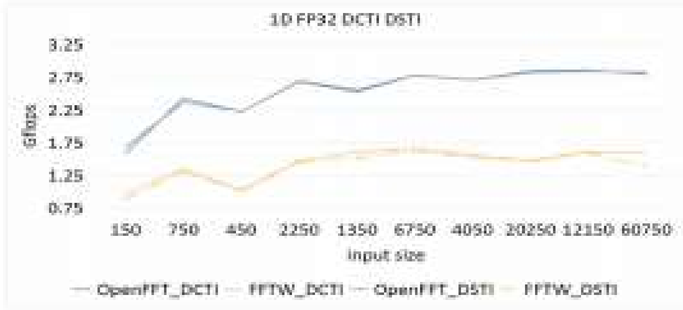


Fig. 6. 1DFP32 DCT/DST I (Color figure online)

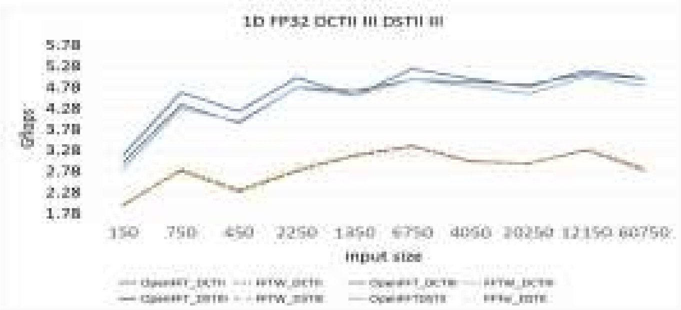


Fig. 7. 1DFP32 DCT/DST II/III (Color

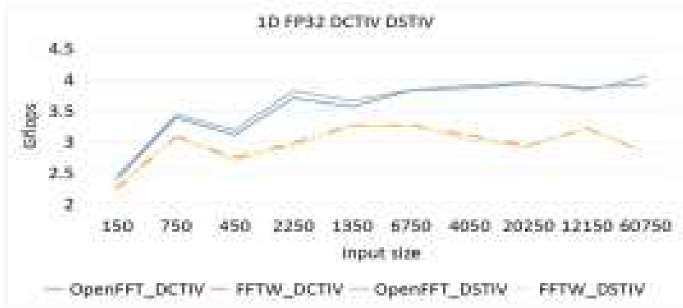


Fig. 8. 1DFP32 DCT/DST IV (Color

05. Experimental Results

Our 1D double transforms outperform FFTw3.3.7 a lot, except for DCT/DST IV when input size is small,

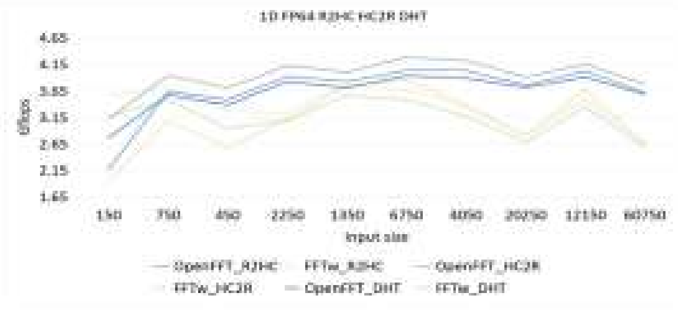


Fig. 9. 1DFP64 R2HC/HC2R/DHT (Color figure online)

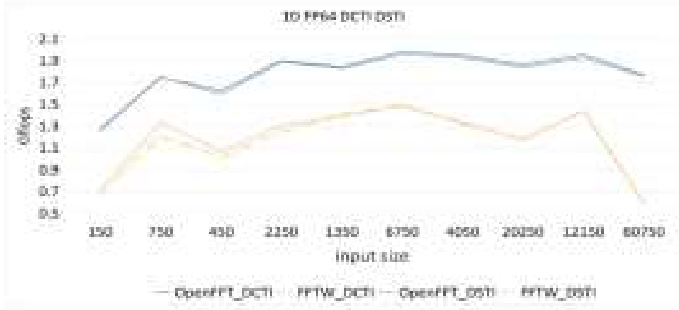


Fig. 10. 1DFP64 DCT/DST I (Color figure online)

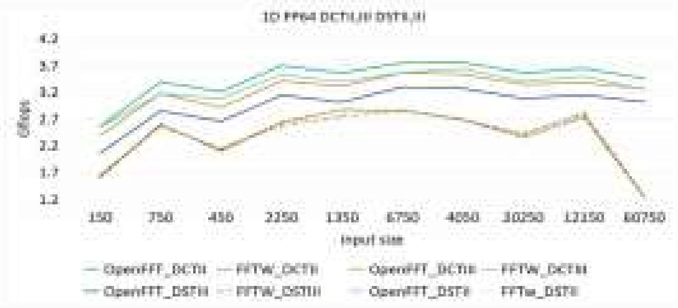


Fig. 11. 1DFP64 DCT/DST II/III (Color

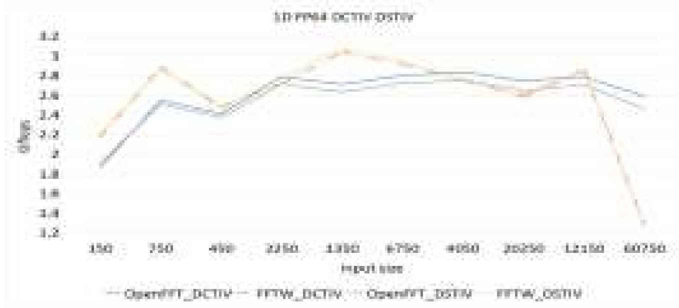


Fig. 12. 1DFP64 DCT/DST IV (Color

05. Experimental Results

- Our 2D float transforms outperform FFTw3.3.7 a lot

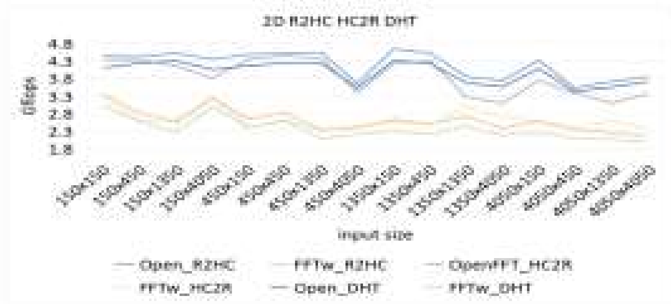


Fig. 14. 2DFP32 R2HC/HC2R/DHT (Color figure online)

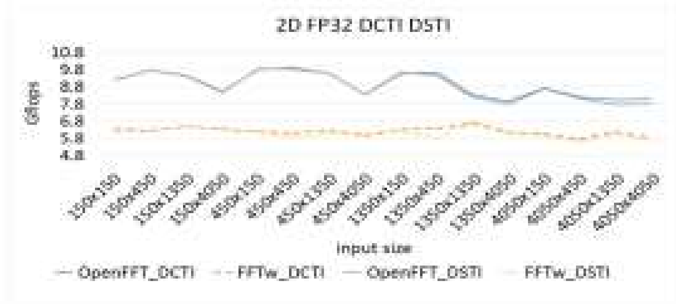


Fig. 15. 2DFP32 DCT/DST I (Color figure online)

Fig. 16. 2DFP32 DCT/DST II/III (Color

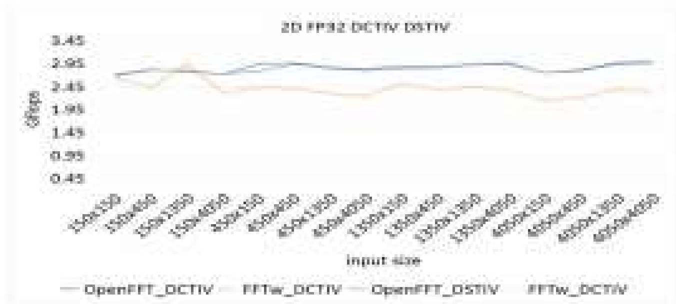


Fig. 17. 2DFP32 DCT/DST IV (Color

05. Experimental Results

- 2D double transforms’ speedup across all kind is from 0.99x to 1.25x

Fig. 18. 2DFP64 R2HC/HC2R/DHT (Color figure online)

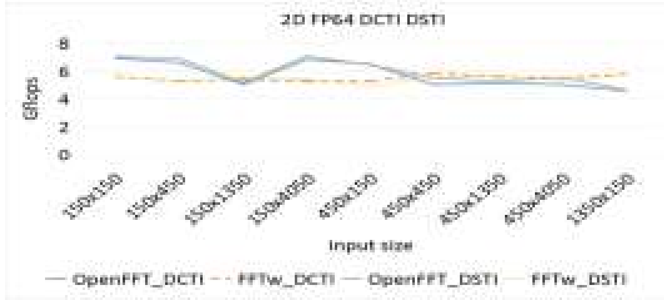


Fig. 19. 2DFP64 DCT/DST I (Color figure online)

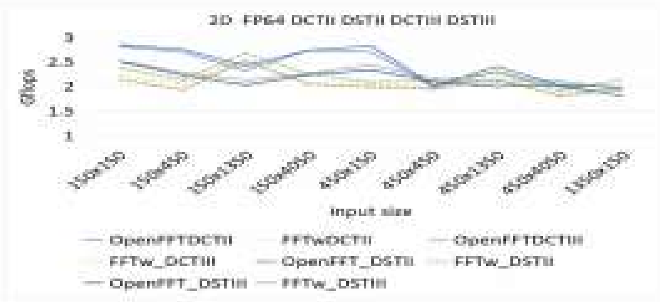


Fig. 20. 2DFP64 DCT/DST II/III (Color

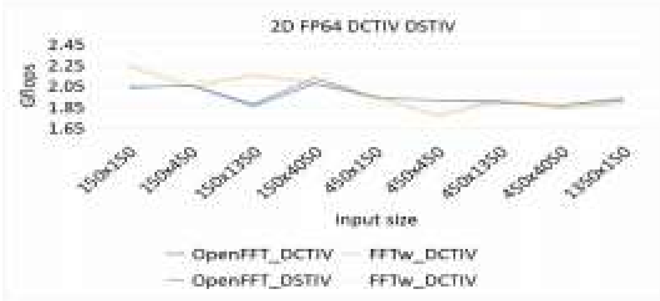


Fig. 21. 2DFP64 DCT/DST IV (Color

05. Performation Analysis

As a whole, our transforms outperform FFTw3.37’s a lot except for some transform cases. Moreover, speedup of double data type is not as significant as float as shown in Fig. 13

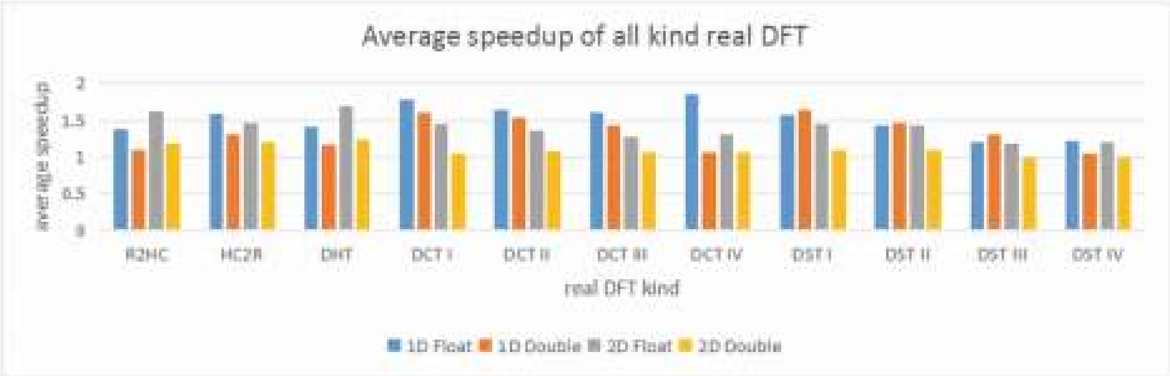


Fig. 13. Speedup across all transform kinds and types (Color figure online)

05. Performance Analysis

Causes:

- 1D Double DCT/DST IV
- General Analysis of performance Degeneration Between 2D and 1D Transforms
- Analysis of Performance of 2D DCT/DST I/IV
- Abnormal Performance Peak Point of Fig. 17
- Influence of Double Data Type

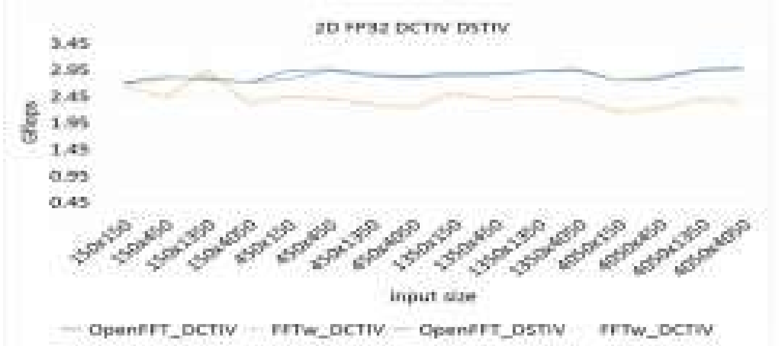


Fig. 17. 2DFP32 DCT/DST IV (Color figure online)

Part 6

Conclusion and Fulture Work

06. Conclusion And Fulture Work

Conclusion:

The paper’s outperform FFTw3.3.7 in most cases.

Fulture work:

- Further optimize double DCT/DST IV
- Design a strategy to optimize radix
- Research optimization for 2^n input size

THANK YOU！