IXPUG Annual Conference 2023

Compiler Approach To Optimize DFT Sinusoidal Computation

Wenju He, Ruqiu Cao, Cheng Chen
Intel DSE Compiler Engineering
Sep 2023

intel®

Background

DFT/IDFT (Discrete Fourier Transform and Invert Discrete Fourier Transform)

$$\text{DFT}: \; y(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-2\pi j \frac{\mathrm{kn}}{\mathrm{N}}}$$

$$e^{-2\pi j \frac{\mathrm{kn}}{\mathrm{N}}} = \cos\left(2\pi \frac{\mathrm{kn}}{\mathrm{N}}\right) - j\sin(2\pi \frac{\mathrm{kn}}{\mathrm{N}})$$

- ❑ A very important module which can reduce peak-to-average power ratio (PAPR) for uplink in 4G and 5G wireless systems. In traditional communications industry, manufacturers use hardware accelerator to implement DFT/IDFT.

```c
#include<math.h>

#define PI (3.14159265358979323846)

/*
* @brief: Discrete Fourier Transform
* @param: {double *} pInput: x(n)
* @param: {double *} pOutput: result
* @param: {int} NofDFT: size of DFT
*/
void DFT(double *pInput, double *pOutput, const int NofDFT)
{
    double *Input = pInput;
    double *Output = pOutput;
    double sine, cosine;
    double real = 0, image = 0;

    for(int i=0; i<NofDFT; i++)
    {
        real = 0;
        image = 0;
        Input = pInput;
        Output = pOutput + 2*i;

        for(int j=0; j<NofDFT; j++)
        {
            sine = sin(2 * PI * i * j / NofDFT);
            cosine = cos(2 * PI * i * j / NofDFT);
            real += *(Input+0) * cosine - *(Input+1) * sine;
            image += *(Input+0) * sine + *(Input+1) * cosine;
            Input += 2;
        }
        *(Output+0) = real;
        *(Output+1) = image;
    }
}
```

Feature Request from 5G FlexRAN

Scale as a power of 2

Size	Prime Fact
2	2^1
4	2^2
8	2^3
16	2^4
32	2^5
64	2^6
128	2^7
256	2^8
512	2^9
1024	2^10
2048	2^11
4096	2^12

Scale as irregular size

Size	Prime Fact	Size	Prime Fact	Size	Prime Fact
12	2^2*3	324	2^2*3^4	1152	2^7*3^2
24	2^3*3	360	2^3*3^2*5	1200	2^4*3*5^2
36	2^2*3^3	384	2^7*3	1296	2^4*3^4
48	2^4*3	432	2^4*3^3	1440	2^5*3^2*5
60	2*3*5	480	2^5*3*5	1500	2^2*3*5^3
72	2^3*3^3	540	2^2*3^3*5	1536	2^9*3
96	2^5*3	576	2^6*3^2	1620	2^2*3^4*5
108	2^2*3^3	600	2^3*3*5^2	1920	2^7*3*5
120	2^3*3*5	648	2^3*3^4	1944	2^3*3^5
144	2^4*3^2	720	2^4*3^2*5	2160	2^4*3^3*5
180	2^2*3^2*5	768	2^8*3	2400	2^5*3*5^2
192	2^6*3	864	2^5*3^3	2700	2^2*3^3*5^2
216	2^3*3^3	900	2^2*3^2*5^2	2916	2^2*3^6
240	2^4*3*5	960	2^6*3*5	3000	2^3*3*5^3
288	2^5*3^2	972	2^2*3^5	3240	2^3*3^3*5
300	2^2*3*5^2	1080	2^3*3^3*5		

Traditional Approach

——Manually pre-compute unique sin/cos values and put them in a fixed-size table.

Pros

- The table contains pre-computed floating-point values. DFT algorithm looks up the table.
- The best performance.

Cons

- Not scalable as DFT size could be very large.
- Not feasible if either DFT size or a parameter is not a constant value.
- Need re-calculation when a parameter is changed.

Compiler Based Approach

Length of DFT is not constant value

Identify candidate function based on DFT pattern matching.

Step 1

Compiler Based Approach——Pattern Matching

Find two-level nested loops and induction variable.

Find sin and cos function calls. Parameter of sin/cos is computed from π, the induction variables and length of DFT.

Make sure remaining instructions in the loop is DFT computation.

If application already does manually optimization of DFT, our pattern matching algorithm won't find DFT pattern, avoids repeated optimization.

Compiler Based Approach **Length of DFT is not constant value**

Identify candidate function based on DFT pattern matching.

Step 1

Insert new instructions to compute the number of unique sin/cos values.

Step 2

Allocate global variable to store the table of unique values.

Step 3

Insert new instructions to pre-compute unique sin/cos values and put them in a table.

Step 4

Insert new instructions to replace original sin/cos function call with look-up from the table.

Step 5

All above is based on LLVM IR level. **Table generation is done in runtime.**

- ❑ Scalable even DFT size could be very large.
- ❑ Feasible when DFT size or a parameter is not a constant value.

Compiler Based Approach

Length of DFT is constant value

Step 1: Identify candidate function based on DFT pattern matching.

Step 2: Compute the number of unique sin/cos values.

Step 3: Allocate global variable to store the table of unique values.

Step 4: Pre-compute unique sin/cos values and put them in a table.

Step 5: Insert new instructions to replace original sin/cos function call with look-up from the table.

This is done in compile time. (Steps 2 and 4)

Experiment Result

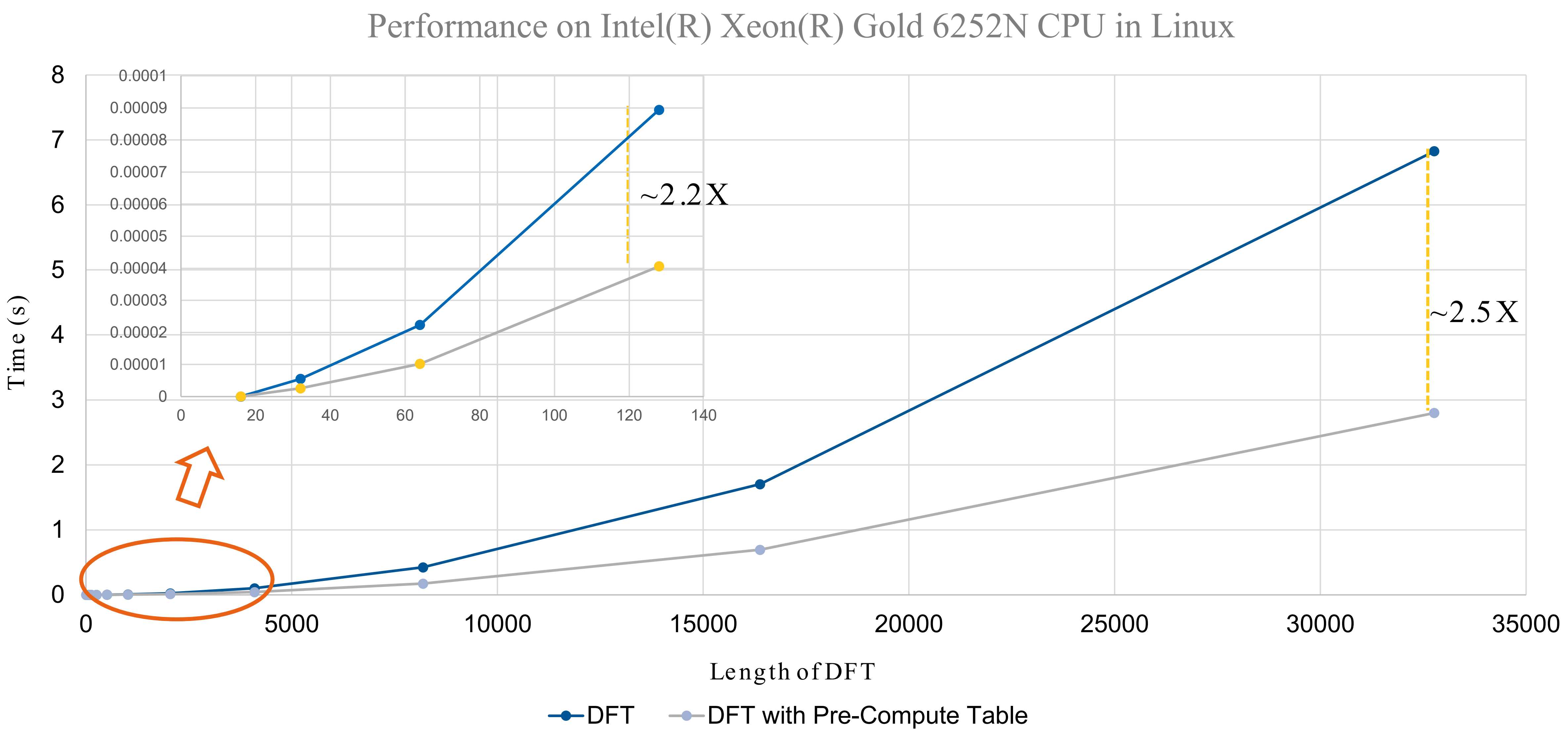
Table Size

Length of DFT (N)	Times of Call Sin Function (N^2)	Num of non -repetitive sin (N/2)
2	4	1
4	16	2
12	144	6
16	256	8
32	1,024	16
64	4,096	32
108	11,664	54
256	65,536	128
512	262,144	256
1,024	1,048,576	512
3,240	10,497,600	1,620
4,096	16,777,216	2,048

Example : Sin Values for DFT8

	0	1	2	3	4	5	6	7
0	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$
1	$\sin(0)$	$\sin\left(\frac{\pi}{4}\right)$	$\sin\left(\frac{\pi}{2}\right)$	$\sin\left(\frac{3\pi}{4}\right)$	$\sin(0)$	$-\sin\left(\frac{\pi}{4}\right)$	$-\sin\left(\frac{\pi}{2}\right)$	$-\sin\left(\frac{3\pi}{4}\right)$
2	$\sin(0)$	$\sin\left(\frac{\pi}{2}\right)$	$\sin(0)$	$-\sin\left(\frac{\pi}{2}\right)$	$\sin(0)$	$\sin\left(\frac{\pi}{2}\right)$	$\sin(0)$	$-\sin\left(\frac{\pi}{2}\right)$
3	$\sin(0)$	$\sin\left(\frac{3\pi}{4}\right)$	$-\sin\left(\frac{\pi}{2}\right)$	$\sin\left(\frac{\pi}{4}\right)$	$\sin(0)$	$-\sin\left(\frac{3\pi}{4}\right)$	$\sin\left(\frac{\pi}{2}\right)$	$-\sin\left(\frac{\pi}{4}\right)$
4	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$	$\sin(0)$
5	$\sin(0)$	$-\sin\left(\frac{\pi}{4}\right)$	$\sin\left(\frac{\pi}{2}\right)$	$-\sin\left(\frac{3\pi}{4}\right)$	$\sin(0)$	$\sin\left(\frac{\pi}{4}\right)$	$-\sin\left(\frac{\pi}{2}\right)$	$\sin\left(\frac{3\pi}{4}\right)$
6	$\sin(0)$	$-\sin\left(\frac{\pi}{2}\right)$	$\sin(0)$	$\sin\left(\frac{\pi}{2}\right)$	$\sin(0)$	$-\sin\left(\frac{\pi}{2}\right)$	$\sin(0)$	$\sin\left(\frac{\pi}{2}\right)$
7	$\sin(0)$	$-\sin\left(\frac{3\pi}{4}\right)$	$-\sin\left(\frac{\pi}{2}\right)$	$-\sin\left(\frac{\pi}{4}\right)$	$\sin(0)$	$\sin\left(\frac{3\pi}{4}\right)$	$\sin\left(\frac{\pi}{2}\right)$	$\sin\left(\frac{\pi}{4}\right)$

Experiment Result



More Application-- Mixed Radix DFT/iDFT Algorithm

$$y(k) = y(p\cdot k_1 + k_0) = y(k_1, k_0) = \sum_{n=0}^{N-1} x(n)\cdot W_N^{nk}$$

$$= \sum_{n_0=0}^{p-1}\sum_{n_1=0}^{q-1} x(q\cdot n_0 + n_1)\cdot W_N^{(q\cdot n_0+n_1)(pk_1+k_0)}$$

$$= \sum_{n_0=0}^{p-1}\sum_{n_1=0}^{q-1} x(n_0, n_1)\cdot W_N^{qn_0k_0} W_N^{pn_1k_1} W_N^{n_1k_0} W_N^{pqn_0k_1}$$

$$= \sum_{n_1=0}^{q-1}\left\{\left[\sum_{n_0=0}^{p-1} x(n_0, n_1)\cdot W_p^{n_0k_0}\right] W_N^{n_1k_0}\right\} W_q^{n_1k_1}$$

$$= \sum_{n_1=0}^{q-1}\left[u(k_0, n_1) W_N^{n_1k_0}\right] W_q^{n_1k_1}$$

$$= \sum_{n_1=0}^{q-1} v(k_0, n_1) W_q^{n_1k_1}$$

$$W_N = e^{j2\pi/N}$$

intel®

Contact: cheng2.chen@intel.com