

Task : FFT Circuit Design

1. 基-2 16点FFT乘法系数表
2. 加减法溢出的问题
3. 乘法的实现方式
4. 测试向量

Task : FFT Circuit Design

1.1 基-2 16点FFT乘法系数表

系数	表达式	实部	虚部
W0	$e^{-j2\pi/16 * 0}$	1	0
W1	$e^{-j2\pi/16 * 1}$	$\cos \pi/8$	$-\sin \pi/8$
W2	$e^{-j2\pi/16 * 2}$	$\cos \pi/4$	$-\sin \pi/4$
W3	$e^{-j2\pi/16 * 3}$	$\sin \pi/8$	$-\cos \pi/8$
W4	$e^{-j2\pi/16 * 4}$	0	-1
W5	$e^{-j2\pi/16 * 5}$	$-\sin \pi/8$	$-\cos \pi/8$
W6	$e^{-j2\pi/16 * 6}$	$-\sin \pi/4$	$-\cos \pi/4$
W7	$e^{-j2\pi/16 * 7}$	$-\cos \pi/8$	$-\sin \pi/8$

这里提供了基-2 16点FFT用到的系数，如果是基-4或者其它算法可以自行推导！

Task : FFT Circuit Design

1.2 正/余弦值的8位无符号二进制浮点数表示

	十进制数值	8位二进制
$\cos \pi/8$	0.9238795325	11101100
$\sin \pi/8$	0.3826834324	01100001
$\cos \pi/4$	0.7071067812	10110101
$\sin \pi/4$	0.7071067812	10110101

无符号数，无整数位，8位均为小数位

Task : FFT Circuit Design

2 加减法溢出的问题

加减法溢出可能是FFT运算中间结果的溢出，也可能是输出结果的溢出。如果是中间结果的溢出，只要增加中间结果的位宽就能解决，但如果是输出结果溢出可能就没办法解决。

本次设计的测试向量会由助教提供。为了减少同学对FFT中加减法溢出的顾虑，测试向量会考虑这个问题，不会让加减法溢出，同学在写RTL可不考虑这个问题，正常+-即可。

Task : FFT Circuit Design

3 乘法的实现方式

本次设计的16点FFT涉及常数乘法，且系数为正/余弦值小于1。常数乘法可采用移位加减的方式实现。

例： $a * 0.75$ 可分解为 $a * 0.5 + a * 0.25 = a \gg 1 + a \gg 2$ 。

移位过程中符号位注意保留！

Task : FFT Circuit Design

4. 测试向量—输入

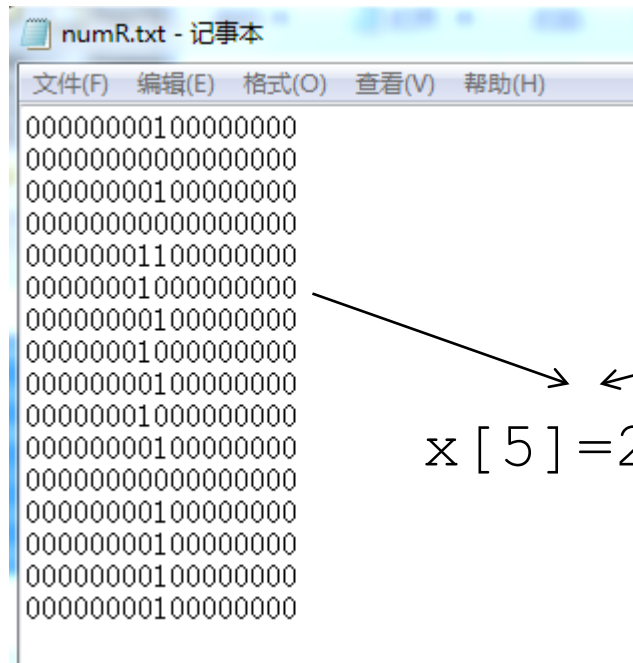
输入	十进制	输入	十进制
x[0]	1	x[8]	1
x[1]	0	x[9]	2+4i
x[2]	1	x[10]	1
x[3]	0	x[11]	0
x[4]	3	x[12]	1
x[5]	2+4i	x[13]	1
x[6]	1+8i	x[14]	1
x[7]	2+2i	x[15]	1

Task : FFT Circuit Design

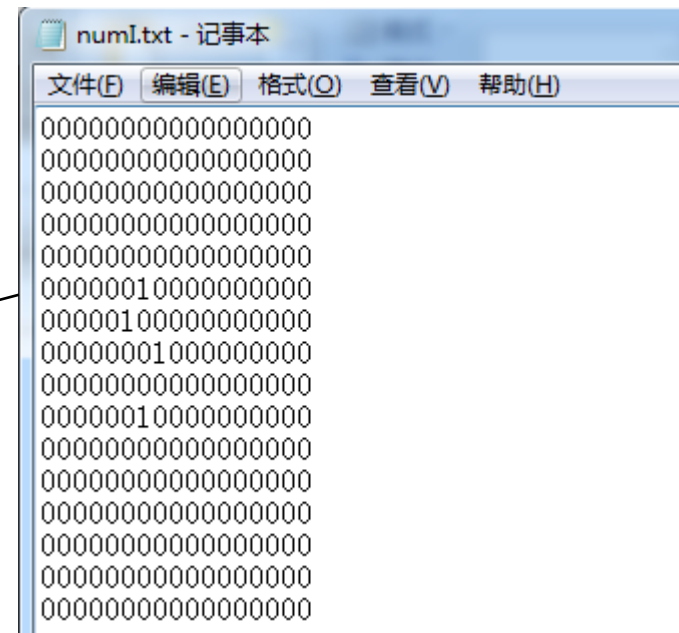
4. 测试向量—输入

numR.txt: 存放16个输入数据的实部

numI.txt: 存放16个输入数据的虚部



A screenshot of a Windows Notepad window titled "numR.txt - 记事本". The window contains 16 lines of text, each consisting of a binary string. The strings are: 00000000100000000, 00000000000000000, 00000000010000000, 00000000000000000, 00000000110000000, 00000000100000000, 00000000010000000, 00000000010000000, 00000000010000000, 00000000010000000, 00000000010000000, 00000000010000000, 00000000010000000, 00000000010000000, 00000000010000000, 00000000010000000.



A screenshot of a Windows Notepad window titled "numI.txt - 记事本". The window contains 16 lines of text, each consisting of a binary string. The strings are: 00000000000000000, 00000000000000000, 00000000000000000, 00000000000000000, 00000000000000000, 00000000000000000, 00000010000000000, 00000100000000000, 00000001000000000, 00000001000000000, 00000000000000000, 00000000000000000, 00000000000000000, 00000000000000000, 00000000000000000, 00000000000000000.

$x[5] = 2 + 4i$

Task : FFT Circuit Design

4. 测试向量—输出

输出	十进制	输出	十进制
X[0]	18+18i	X[8]	2-2i
X[1]	5.4327-15.2721i	X[9]	5.881-0.0416i
X[2]	-10+4.2426i	X[10]	-10-4.2426i
X[3]	2.0542+10.3628i	X[11]	9.2595+4.9509i
X[4]	8-10i	X[12]	-4-6i
X[5]	-8.8112+3.564i	X[13]	-2.5025+3.7497i
X[6]	3.1716+1.4142i	X[14]	8.8284-1.4142i
X[7]	0.4276+2.8760i	X[15]	-11.7413-10.1897i

Task : FFT Circuit Design

4. 测试向量—输出的实部

输出	实部(二进制补码)	输出	实部(二进制补码)
X[0]	0_00010010_00000000	X[8]	0_00000010_00000000
X[1]	0_00000101_01101110	X[9]	0_00000101_11100001
X[2]	1_11110110_00000000	X[10]	1_11110110_00000000
X[3]	0_00000010_00001101	X[11]	0_00001001_01000010
X[4]	0_00001000_00000000	X[12]	1_11111100_00000000
X[5]	1_11110111_00110000	X[13]	1_11111101_01111111
X[6]	0_00000011_00101011	X[14]	0_00001000_11010100
X[7]	0_00000000_01101101	X[15]	1_11110100_01000010

Task : FFT Circuit Design

4. 测试向量—输出的虚部

输出	虚部(二进制补码)	输出	虚部(二进制补码)
X[0]	0_00010010_00000000	X[8]	1_11111110_00000000
X[1]	1_11110000_10111010	X[9]	1_11111111_11110101
X[2]	0_00000100_00111110	X[10]	1_11111011_11000001
X[3]	0_00001010_01011100	X[11]	0_00000100_11110011
X[4]	1_11110110_00000000	X[12]	1_11111010_00000000
X[5]	0_00000011_10010000	X[13]	0_00000011_10111111
X[6]	0_00000001_01101010	X[14]	1_11111110_10010101
X[7]	0_00000010_11100000	X[15]	1_11110101_11001111

Task : FFT Circuit Design

4. 测试向量—输出误差分析和流水线验证

电路的输出可以直接以补码形式输出，输出值可能与提供的标准值存在误差。比较误差的方法可以以补码形式直接比较，也可以将补码转换成10进制再做比较，实际输出和标准输出应当是大致相等的。

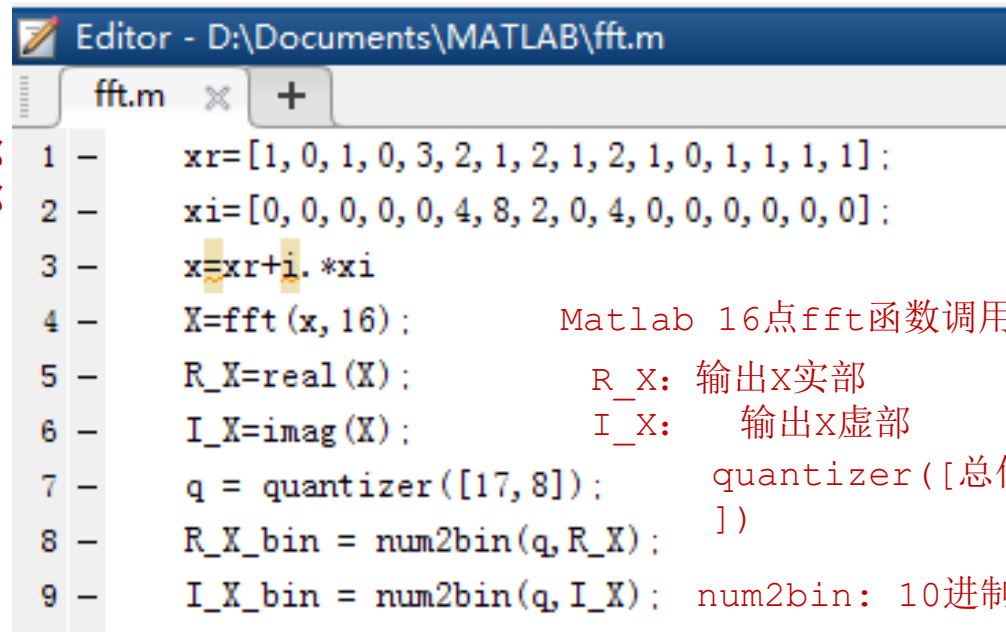
提供的测试向量只是为了验证FFT的功能，只进行一次16点FFT运算，实际电路应当是能够流水线式地执行若干次16点FFT。流水线的功能可以自行额外增加测试向量来进行验证。

Task : FFT Circuit Design

4. 测试向量生成方法—matlab

fft.m代码功能：自定义输入x，执行16点FFT，输出X

xr: 输入x实部
xi: 输入x虚部



```
1 - xr=[1, 0, 1, 0, 3, 2, 1, 2, 1, 2, 1, 0, 1, 1, 1, 1];
2 - xi=[0, 0, 0, 0, 0, 4, 8, 2, 0, 4, 0, 0, 0, 0, 0, 0];
3 - x=xr+i.*xi
4 - X=fft(x, 16);
5 - R_X=real(X);
6 - I_X=imag(X);
7 - q = quantizer([17, 8]);
8 - R_X_bin = num2bin(q, R_X);
9 - I_X_bin = num2bin(q, I_X);
```

Matlab 16点fft函数调用

R_X: 输出x实部
I_X: 输出x虚部

quantizer([总位数, 小数位数])

num2bin: 10进制转2进制补码