(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 108319804 B
(45)授权公告日 2023.08.08

(21)申请号 201810341207.3

(22)申请日 2018.04.17

(65)同一申请的已公布的文献号
申请公布号 CN 108319804 A

(43)申请公布日 2018.07.24

(73)专利权人 福州大学
地址 350002 福建省福州市鼓楼区工业路523号

(72)发明人 魏榕山 胡志杰 吴志强

(74)专利代理机构 福州元创专利商标代理有限公司 35100
专利代理师 蔡学俊

(51)Int.Cl.
G06F 30/3308(2020.01)
G06F 115/06(2020.01)

(56)对比文件
CN 101847137 A,2010.09.29
CN 102033852 A,2011.04.27
CN 102087640 A,2011.06.08
CN 103176949 A,2013.06.26
CN 103226543 A,2013.07.31
US 2005131976 A1,2005.06.16
WO 2018027706 A1,2018.02.15
晏敏 等.低功耗可配置FFT处理器的ASIC设计.微电子学.2010,第40卷(第6期),第787-791页.

审查员 于景

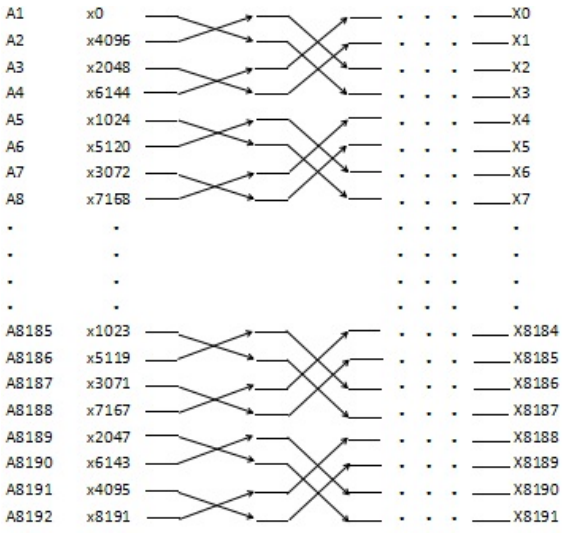
权利要求书1页 说明书6页 附图4页

(54)发明名称

低资源调用的8192点基2 DIT ASIC设计方法

(57)摘要

本发明涉及一种低资源调用的8192点基2 DIT ASIC设计方法。(1)计算主体模块:基于基2 DIT思想,利用状态机作为主体,实现对DIT蝶形图的描述,且在状态机内部采用阻塞赋值的方式,DIT蝶形图的每一级均由两组寄存器组成,在用状态机实现时,利用两组寄存器组之间的数据更新来回根据权重值变换,每组寄存器组中都有实部与虚部组成;(2)三角函数生成模块:用于生成三角函数,以便于计算主体模块进行快速傅里叶变换、欧拉变换后的三角函数调用。本发明通过蝶形图中每级主寄存器组之间的重复调用,降低计算复杂度,提高精确度,使得每级时域抽选过程中层层误差迭加的积累值大为降低,更快地实现快速傅里叶变换时域抽选过程,达到实现低资源调用。



CN 108319804 B

1.一种低资源调用的8192点基2 DIT ASIC设计方法，其特征在于，包括以下两部分模块的设计：

(1)用于DIT设计计算处理的计算主体模块：

用于DIT设计计算处理的计算主体模块基于基2 DIT思想，利用状态机作为主体，实现对DIT蝶形图的描述，且在状态机内部采用阻塞赋值的方式，DIT蝶形图的每一级均由两组寄存器组成，在用状态机实现时，利用两组寄存器组之间的数据更新来回根据权重值变换，每组寄存器组中都有实部与虚部组成；

(2)三角函数生成模块：

用于生成三角函数，以便于计算主体模块进行快速傅里叶变换、欧拉变换后的三角函数调用；

所述DIT蝶形图为8192点DIT蝶形图；

用于DIT设计计算处理的计算主体模块实现DIT设计的具体流程如下：

首先对DIT蝶形图级数、每个级数在小组中对应的计数点、总的计数的计数数目、以e为底的指数函数幅角化的三角函数的角度值初始化；当处于DIT蝶形图的第一级蝶形图时，读入计数点的计算值，并对每个数值扩大256倍，即每一个二进制值左移8位，此处读入的是数值的实部，并且在第一级蝶形图的时候需要对每一个实部对应的虚部全部取0，在读入实部与虚部之后，需要根据位置确定每一小组的计数点，并倒序处理，而后计算出对应的三角函数的数值，相加或者相减得到第一级蝶形图中以2个计数点为一组的计算结果，计算结束后需要把补码转化为原码寄存到寄存器组之中，以此完成第一级蝶形图的计算，之后更新级数、级数小组中对应的计数点、总的计算的计数数目、以e为底的指数函数幅角化过来的三角函数的角度值，开始新一轮蝶形图的计算；每一级重复蝶形图的计算、补码与原码的相互转化之后，重复更新初始值，循环计算寄存直到结束。

低资源调用的8192点基2 DIT ASIC设计方法

技术领域

[0001] 本发明涉及一种低资源调用的8192点基2 DIT ASIC设计方法。

背景技术

[0002] 在当今DSP技术运用越来越频繁的时代,快速傅里叶变换算法设计的重要性也开始受到关注,快速傅里叶变换的好坏往往是决定处理器性能的一个重要因素,并且随着算法运用的开发,快速傅里叶变换在图像与视频等方面的处理效果也备受青睐,不仅仅存在于日常的生活运用中。快速傅里叶变换的产生过程决定了其在军工方面卓越的应用地位,在卫星信号的处理,音频信号方面的滤波,雷达信号接收的去噪声等等国防建设中也起到了至关重要的地位。因此对于如何更稳定地实现关于快速傅里叶变换算法的硬件电路设计的同时,也对电路的资源调用程度与速度快慢提出了更高的要求。

[0003] 现有DIT设计的方案多样,但是考虑到计算过程中的寄存器的可持续复用以便减少资源的调用,以及设计过程中整个系统结构更为简单化以便分析计算各级权重的计算调用,利用蝶形图计算的结构显然是最好的选择。经过调研,在之前的绝大多数FFT设计的过程中,针对小计算点的设计大部分采用的是片外存储权重值随时待调用的状态。而大计算点的设计又往往采用的以e为底数的指数函数的计算,由于以e为底的指数函数的分布过程对于幅角的精度要求太高,在幅角接近于0时数值变换收敛于1,在幅角相对较大时,得出的计算结果又往往变化太大,在数字电路的设计上难以达到以极低资源得出计算结果的效果,而且这样做的效果使得本身设计计算后的计算数值精度不高,以及消耗大量的计算单元,当精度要求越来越高时,消耗的资源就达到一个相当可观的地步。因此在这里本设计采用软件设计的思路,提取上述两种设计方法的优点,将DIT设计中的权重计算经过欧拉公式的变换,转换成三角函数的形式,分别得出关于实部与虚部的计算结果,结合这两个计算结果相乘得出复数计算的结果。而对于三角函数的设计,在考虑了用CORDIC设计方法以及泰勒展开之后的三角函数计算结果后,本设计就三角函数调用时,内部角度在同一时间段内是一致的,考虑到三角函数本身的对称性以及角度的变化范围在0到180之间,本设计采用的是优化过后的查表法,做出关于SIN函数90度之间的变化结果,然后利用正弦函数在0到180之间的对称关系,正弦函数以及余弦函数之间的对应关系,经过简单转换过后直接得出调用的数值,计算过程相对简单方便。

[0004] 通过基2的特性,相比于基4以及基8的设计思想,基2设计本身对于计算点的限制小,通过蝶形图可以达到反复调用寄存器组的目的,达到尽量以极低的资源消耗得到告诉的8192点DIT设计的结果。经过调研,现今的大部分关于FFT大计算点的设计基本上是立足于多核处理器或者高性能处理器设计的,以流水的形式进行运算,这就导致计算过程中调用大量运算单元与存储器,多出了功耗的开销。因此,本设计在保证运行速度与精度的情况下,优化算法结构,以状态机的方式实现低资源调用。

发明内容

[0005] 本发明的目的在于提供一种低资源调用的8192点基2 DIT ASIC设计方法，通过蝶形图中每级主寄存器组之间的重复调用，降低计算复杂度，提高精确度，使得每级时域抽选过程中层层误差迭加的积累值大为降低，更快地实现快速傅里叶变换时域抽选过程，优化设计流程，达到实现低资源调用。

[0006] 为实现上述目的，本发明的技术方案是：一种低资源调用的8192点基2 DIT ASIC设计方法，包括以下两部分模块的设计：

[0007] (1)用于DIT设计计算处理的计算主体模块：

[0008] 用于DIT设计计算处理的计算主体模块基于基2 DIT思想，利用状态机作为主体，实现对DIT蝶形图的描述，且在状态机内部采用阻塞赋值的方式，DIT蝶形图的每一级均由两组寄存器组成，在用状态机实现时，利用两组寄存器组之间的数据更新来回根据权重值变换，每组寄存器组中都有实部与虚部组成；

[0009] (2)三角函数生成模块：

[0010] 用于生成三角函数，以便于计算主体模块进行快速傅里叶变换、欧拉变换后的三角函数调用。

[0011] 在本发明一实施例中，所述DIT蝶形图为8192点DIT蝶形图。

[0012] 在本发明一实施例中，用于DIT设计计算处理的计算主体模块实现DIT设计的具体流程如下：

[0013] 首先对DIT蝶形图级数、每个级数在小组中对应的计数点、总的计数的计数数目、以e为底的指数函数幅角化的三角函数的角度值初始化；当处于DIT蝶形图的第一级蝶形图时，读入计数点的计算值，并对每个数值扩大256倍，即每一个二进制值左移8位，此处读入的是数值的实部，并且在第一级蝶形图的时候需要对每一个实部对应的虚部全部取0，在读入实部与虚部之后，需要根据位置确定每一小组的计数点，并倒序处理，而后计算出对应的三角函数的数值，相加或者相减得到第一级蝶形图中以2个计数点为一组的计算结果，计算结束后需要把补码转化为原码寄存到寄存器组之中，以此完成第一级蝶形图的计算，之后更新级数、级数小组中对应的计数点、总的计算的计数数目、以e为底的指数函数幅角化过来的三角函数的角度值，开始新一轮蝶形图的计算；每一级重复蝶形图的计算、补码与原码的相互转化之后，重复更新初始值，循环计算寄存直到结束。

[0014] 相较于现有技术，本发明具有以下有益效果：本发明通过蝶形图中每级主寄存器组之间的重复调用，降低计算复杂度，提高精确度，使得每级时域抽选过程中层层误差迭加的积累值大为降低，更快地实现快速傅里叶变换时域抽选过程，优化设计流程，达到实现低资源调用。

附图说明

[0015] 图1是本发明采用的8192点fft蝶形图。

[0016] 图2为资源调用图。

[0017] 图3为部分寄存器结果。

[0018] 图4为初始化数据图。

[0019] 图5为结果波形图。

[0020] 图6为拟合指数函数图。

[0021] 图7为设计流程图。

具体实施方式

[0022] 下面结合附图,对本发明的技术方案进行具体说明。

[0023] 本发明的一种低资源调用的8192点基2 DIT ASIC设计方法,包括以下两部分模块的设计:

[0024] (1)用于DIT设计计算处理的计算主体模块:

[0025] 用于DIT设计计算处理的计算主体模块基于基2 DIT思想,利用状态机作为主体,实现对DIT蝶形图的描述,且在状态机内部采用阻塞赋值的方式,DIT蝶形图的每一级均由两组寄存器组成,在用状态机实现时,利用两组寄存器组之间的数据更新来回根据权重值变换,每组寄存器组中都由实部与虚部组成;

[0026] (2)三角函数生成模块:

[0027] 用于生成三角函数,以便于计算主体模块进行快速傅里叶变换、欧拉变换后的三角函数调用。

[0028] 所述DIT蝶形图为8192点DIT蝶形图(如图1所示)。

[0029] 用于DIT设计计算处理的计算主体模块实现DIT设计的具体流程如下:

[0030] 首先对DIT蝶形图级数、每个级数在小组中对应的计数点、总的计数的计数数目、以e为底的指数函数幅角化的三角函数的角度值初始化;当处于DIT蝶形图的第一级蝶形图时,读入计数点的计算值,并对每个数值扩大256倍,即每一个二进制值左移8位,此处读入的是数值的实部,并且在第一级蝶形图的时候需要对每一个实部对应的虚部全部取0,在读入实部与虚部之后,需要根据位置确定每一小组的计数点,并倒序处理,而后计算出对应的三角函数的数值,相加或者相减得到第一级蝶形图中以2个计数点为一组的计算结果,计算结束后需要把补码转化为原码寄存到寄存器组之中,以此完成第一级蝶形图的计算,之后更新级数、级数小组中对应的计数点、总的计算的计数数目、以e为底的指数函数幅角化过来的三角函数的角度值,开始新一轮蝶形图的计算;每一级重复蝶形图的计算、补码与原码的相互转化之后,重复更新初始值,循环计算寄存直到结束。

[0031] 以下为本发明的具体实现过程。

[0032] 本发明方法主要涉及三个模块组成,调用随机函数生成关于8192测试点的模块、用于DIT设计计算处理的主体模块、三角函数生成模块。生成关于8192测试点的模块是调用的SYSTEM VERILOG 中的随机函数生成,不算在整个ASIC电路设计的过程中,只是拿来验证算法的结果的精确性以及为电路提供一个关于数据处理接口的接入。在之前的诸多关于FFT设计的工程中,FFT设计的分级蝶形图计算使得流水处理在于FFT设计中比较不适用。一般采用的是并行处理每一个分级的部分结果,处理完之后再综合计算,这样做的弊端在于这种做法一般是分模块执行并行过程的,而这个并行处理之前,会有一个大量数据输入的时钟消耗等待时间,以及数据接收后的数据传输时间。对于小的计算点,没必要进行并行处理过程,而对于大的计算点来说,后期所要为并行模块接入数据输出后的计算结果又显得逻辑结构过于错综复杂。出于对分级计算过程中逻辑结构简单化的考虑,本设计这里采用的状态机实现串行处理的效果,由于在状态机中对DIT蝶形图的描述呈现的结构过程较为

清晰明了，降低设计过程中的复杂度。在设计的过程中，采用状态机设计时，为了使得电路结果能够及时得到后续步骤的补充以及衔接，在状态机内部采用阻塞赋值的方式，大量节约状态机内部计算消耗的时钟，使得对接的寄存器组得以及时传递数值以及擦除原始数据，达到寄存器组之间数据传递平衡的效果，最为重要的是，在传递数据之后，原始寄存器组所拥有的数据便处于无用的状态，我们可以及时采用这部分寄存器组达到寄存器组之间数据互相更新的方式，而每一级中每一小组中的两个计算点之间是可以在一个状态机时钟内达到这种效果的。

[0033] 快速傅里叶变换是把原始信号所拥有的时域部分的信号到频域部分转变的一个过程：

[0034] $$F(\omega)=\mathcal{F}[f(t)]=\int_{-\infty}^{\infty}f(t)e^{-iwt}dt$$

[0035] 在快速傅里叶变换的过程中，对于以e为底的指数函数乘数部分，本设计通过欧拉公式的转换，将指数函数的复数形式转变为三角函数形式的复数形式，得以分离出实数部分以及隐藏在以e为底的指数函数的虚数部分的数据，具体的欧拉公式转换的一个形式如下所示：

[0036] $$e^{ix}=cosx+isinx$$

[0037] 在计算以e为底的指数函数数据时，如果要计算的话，在数字电路中如果将之泰勒展开的话，经过matlab软件模拟，需要展开到第10项时，泰勒展开式才能达到与原始函数拟合的效果，在展开到第5项时，数值随着指数提高而导致误差逐渐增大。函数走向如下图6所示。在用matlab对以e为底的指数函数的泰勒展开到第5项时与第10项时的公式如下所示：

[0038]
```
y1 =
exp(-x)
>> y2=taylor(y1,x,'order',5)
y2 =
x^4/24 - x^3/6 + x^2/2 - x + 1
```

[0039]
```
y1 =
exp(-x)
>> y3=taylor(y1,x,'order',10)
y3 =
- x^9/362880 + x^8/40320 - x^7/5040 + x^6/720 - x^5/120 + x^4/24 - x^3/6 + x^2/2 - x + 1
```

[0040] 所以在使用泰勒展开的时候，对于计算数值的小数部分的精度需求很大，而且计算的复杂度以及原始数值所带的正负数变换计算在数字电路中需要转换成补码形式，大大增大了设计的难度，降低了数据的精确度。所以在这里就体现出了利用欧拉公式转换成三角函数过程的重要性，大大提高了数据的精确度，降低了迭代过程中的误差积累，降低了电路设计的复杂度。

[0041] 在设计DIT(快速傅里叶变换的时域抽选)时，本设计这里用到了在DSP设计FFT时域抽选中的蝶形图，更为直观地更为有层次地进行DIT设计，并且随着层次级数的提高，可以方便快捷地得出权重中级数相应的数值计算。

[0042] 在整体设计中调用随机函数生成关于8192测试点的模块主要是调用随机数据生成的函数来设置8192点DIT设计的数据，然后之后可以有两种调用方式。一种是直接通过IO口输入8192点数据输入，一种是在原始初始化输入数据生成的时候直接写入到本工程中的txt文本中，在主体设计模块里调用此txt文本读入数据，存储到寄存器组中。在这里为了使其最后数字电路可综合，在生成txt文档的同时，在主体模块暂时不做关于文本调用的函数读入，使用IO口读入8192点数据。如附图4中的初始化数据生成所示。

[0043] 三角函数调用数据的模块主要体现在经过欧拉公式变换之后，利用三角函数的复数形式划分出实部与虚部，直接将以e为底的指数函数的复数形式计算，变为更为简单的实部与实部、虚部与虚部之间、实部与虚部之间的计算，最后的数据整理成实部与虚部两部分。这部分三角函数的正弦函数与余弦函数可以通过在简单的变换调取正弦函数值的0到90度之间的数据，判别正负数之后调取数值，这里用查表法，结合三角函数公式，将正弦函数0到90之间按照正弦函数的结果平均划分，存储三角函数值。

[0044] 由于DIT设计计算处理的主体模块主体是利用状态机实现的，在状态机的实现工程中，采用的阻塞赋值，大量节约了由于非租塞赋值的计算与同步时钟赋值问题而造成的时钟等待以及逻辑设计过程中大量数据的调取在非阻塞赋值中计算存储同步问题的寄存器组浪费。在用状态机实现时，主要利用两组寄存器组之间的数据更新来回根据权重值变换，每组寄存器组中都有实部与虚部组成。实部与虚部分开计算，互相结合，导致在计算中两者需要同步进行更新，擦除原来存在的数据。之所以可以用两组寄存器组来回更新数据而不造成大的浪费，主要是我们采用的基2 DIT设计，所以每级之间都是以两个为一组同步更新的，不用考虑其他多余的情况，每一级之中的每个寄存器组只需要更新一次，通过层层迭加的关系达到最后每个数据与其他所有数据的计算的需求。

[0045] 整体DIT的设计如附图7设计流程图所示，首先对级数、每个级数在小组中对应的计数点、总的计算的计数数目、以e为底的指数函数幅角化过来的三角函数的角度值初始化，当处于蝶形图的第一级时，读入计算点的计算值，这里对每个数值扩大256倍，也就是每一个二进制值左移8位，这里读入的是数值的实部，并且在第一级蝶形图的时候需要对每一个实部对应的虚部全部取0，这是因为在开始计算的时候，为了便于计算与写设计电路的过程中的实时对寄存器组的简单验证判断设计结果，所以即使在这一级的虚部置入虚数去计算，把这整个8192点当成某一个n*8192点的DIT设计的某一部分的话，也是可行的。在读入实部与虚部之后，需要根据位置确定每一小组的计算点，这里假设已经事先经过了倒序，计算出对应的三角函数的数值，相加或者相减得到第一级蝶形图中以2个计算点为一组的计算结果，在计算相加或者相减的过程中，要注意带符号数的存在，由于有符号数的计算相对注意的事项比较多，容易造成计算混乱，所以为了设计的方便与算法的可实现化，手动在每一个数的最左边的二进制数置符号数，所以不仅相乘时要考虑结果的符号问题，对于不能确定计算符号的数值，需要先把原码转化为补码计算，避免计算混乱。计算结束后需要把补码转化为原码寄存到寄存器组之中，以此完成第一级的计算，之后更新级数、级数小组中对应的计数点、总的计算的计数数目、以e为底的指数函数幅角化过来的三角函数的角度值，开始新一轮蝶形图的计算。由于每一级蝶形图的计算都要经过大量的计算与数值转化、保存，所以每一级的计算都要耗费大量的计算资源与寄存器组。这里也就是我们提出低资源设计大计算点DIT电路的意义所在。每一级重复蝶形图的计算、补码与原码的相互转化之

后,重复更新初始值,循环计算寄存直到结束。

[0046] 对于整个设计的仿真结果正如图5的波形图所示,此波形图展示的是设计过程中所调用到的部分信号在整个电路仿真过程中的波形。在大计算点的情况下,设计运行的复杂度是与2的指数大小成正比的,本设计在高复杂度的情况下,整体设计仿真时间为1.8毫秒,减去开始时候对内部寄存器的赋值初始化所开销的时间0.2毫秒,整个电路运行的时间为1.6毫秒,电路接入的基本时钟频率为100MHz。电路仿真结果表明,此优化的算法电路在低资源调用的前提下,还达到了高速运行的效果。电路的仿真结果包括寄存器部分的计算值大小,此部分的仿真结果如图3所示,图3显示的是部分寄存器的计算数值,在经过整个电路的运行之后,将结果分为实部与虚部两部分寄存。经过优化过后的算法对于电路精确度有较大的提升,此设计中图三所示是实部数据,每一个在寄存器中的数值是按照256倍扩大其大小以保证在优化的算法之后,小数的数值不被截取。保证多次迭加计算之后的误差减小。

[0047] 此设计以低资源调用为主要设计方向。经过对算法的改进以及对电路结构的优化,来达到低资源调用的目标。如图2所示,本设计在VIVADO软件中综合后跑出的资源使用情况。整个设计所使用的是芯片型号为xc7vx485tffg1157-1 ,其中VIVADO显示Slice LUTs使用了131,个单位,占总数的0.04%。Slice Registers 使用了0.03%,使用了195个单位。整个设计占用了600个Nets,419个Leaf Cells。此设计中,电路资源重复利用率高,节省资源硬件开销,运行周期短。

[0048] 以上是本发明的较佳实施例,凡依本发明技术方案所作的改变,所产生的功能作用未超出本发明技术方案的范围时,均属于本发明的保护范围。

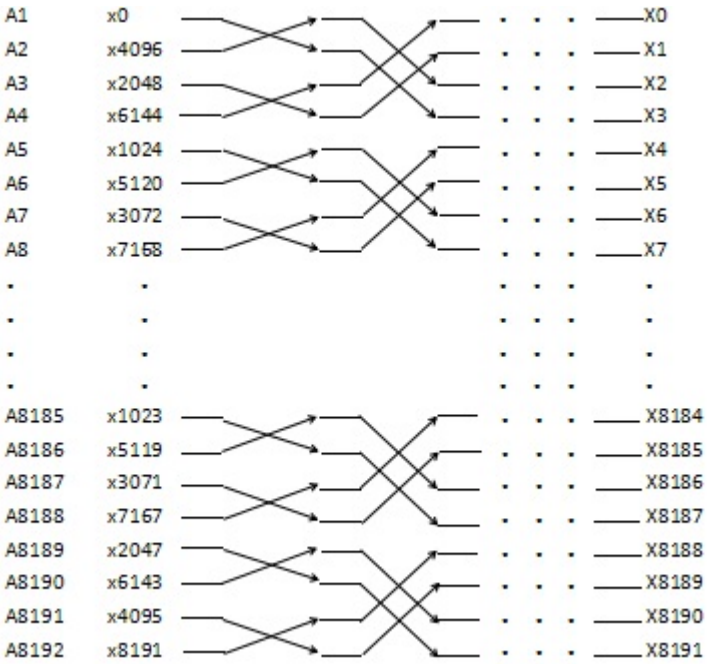


图1

Hierarchy

Name	Slice LUTs (303600)	Slice Registers (607200)	Slice (75900)	LUT as Logic (303600)	LUT Flip Flop Pairs (303600)	Bonded IOB (600)
fft_1	0.04%	0.03%	0.11%	0.04%	0.02%	0.50%

图2

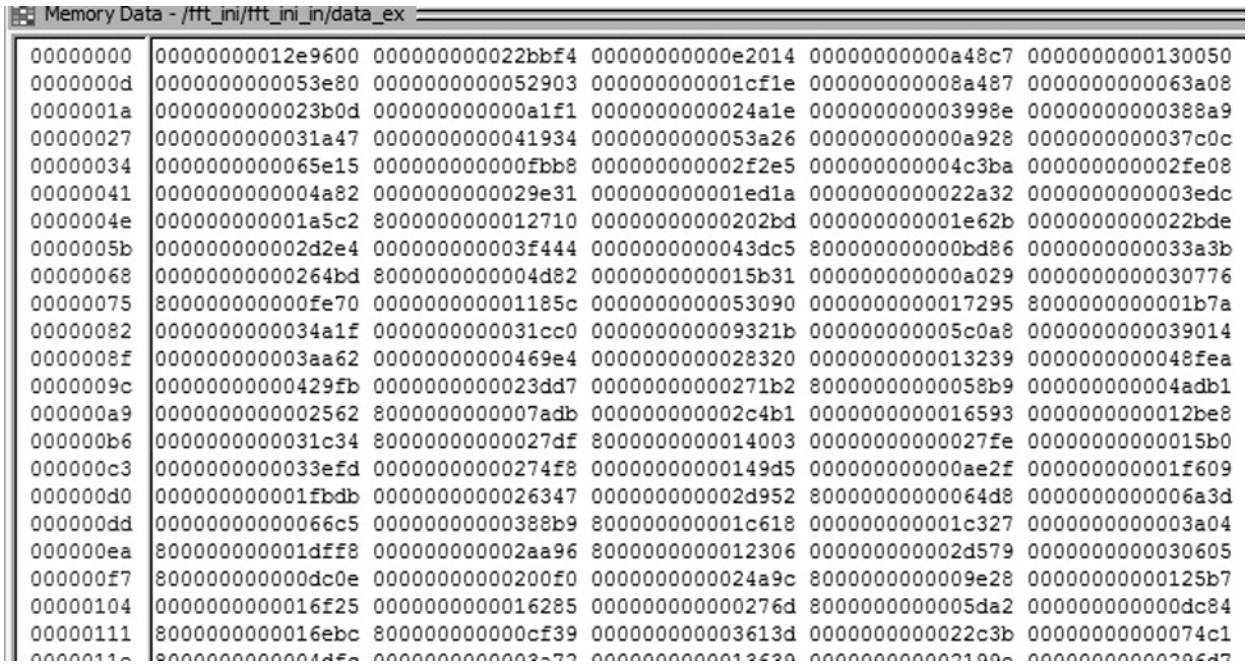


图3

```
initial
begin
out_en=0;
clk=0;
count=0;
handle = $fopen("data.txt");//????
  for(i=0;i<8192;i=i+1)
  begin
  data[i]={$random}%20;
  //data_out=data[i];
  $fdisplay(handle,"%h",data[i]);
  end

$fclose(handle);
end
```

图4

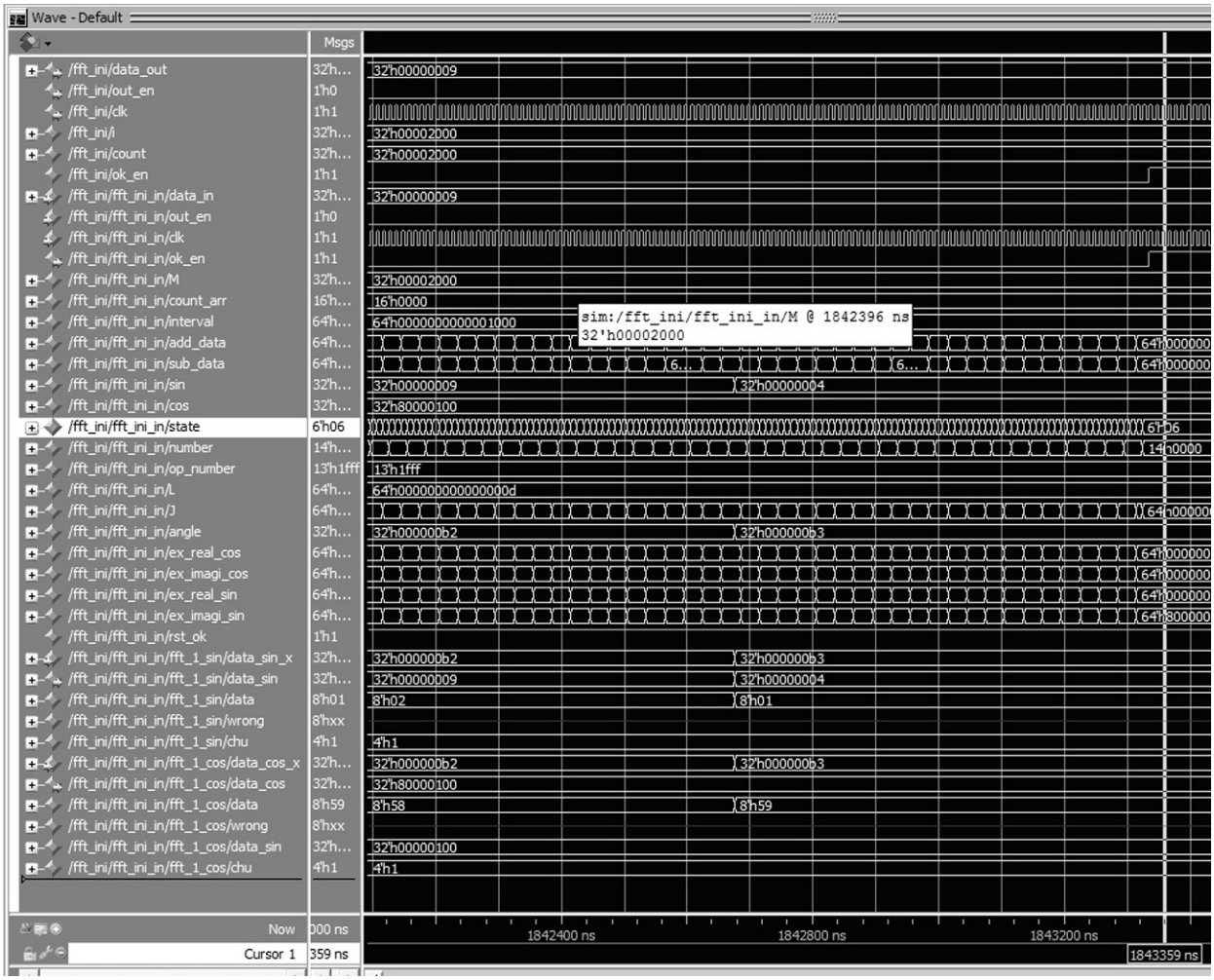

Wave - Default
Msgs
/fft_ini/data_out
/fft_ini/out_en
/fft_ini/clk
/fft_ini/i
/fft_ini/count
/fft_ini/ok_en
/fft_ini/fft_ini_in/data_in
/fft_ini/fft_ini_in/out_en
/fft_ini/fft_ini_in/clk
/fft_ini/fft_ini_in/ok_en
/fft_ini/fft_ini_in/M
/fft_ini/fft_ini_in/count_arr
/fft_ini/fft_ini_in/interval
/fft_ini/fft_ini_in/add_data
/fft_ini/fft_ini_in/sub_data
/fft_ini/fft_ini_in/sin
/fft_ini/fft_ini_in/cos
/fft_ini/fft_ini_in/state
/fft_ini/fft_ini_in/number
/fft_ini/fft_ini_in/op_number
/fft_ini/fft_ini_in/L
/fft_ini/fft_ini_in/J
/fft_ini/fft_ini_in/angle
/fft_ini/fft_ini_in/ex_real_cos
/fft_ini/fft_ini_in/ex_imagi_cos
/fft_ini/fft_ini_in/ex_real_sin
/fft_ini/fft_ini_in/ex_imagi_sin
/fft_ini/fft_ini_in/rst_ok
/fft_ini/fft_ini_in/fft_1_sin/data_sin_x
/fft_ini/fft_ini_in/fft_1_sin/data_sin
/fft_ini/fft_ini_in/fft_1_sin/data
/fft_ini/fft_ini_in/fft_1_sin/wrong
/fft_ini/fft_ini_in/fft_1_sin/chu
/fft_ini/fft_ini_in/fft_1_cos/data_cos_x
/fft_ini/fft_ini_in/fft_1_cos/data_cos
/fft_ini/fft_ini_in/fft_1_cos/data
/fft_ini/fft_ini_in/fft_1_cos/wrong
/fft_ini/fft_ini_in/fft_1_cos/data_sin
/fft_ini/fft_ini_in/fft_1_cos/chu
sim:/fft_ini/fft_ini_in/M @ 1842396 ns
32'h00002000
Now
Cursor 1
1842400 ns
1842800 ns
1843200 ns
1843359 ns

图5

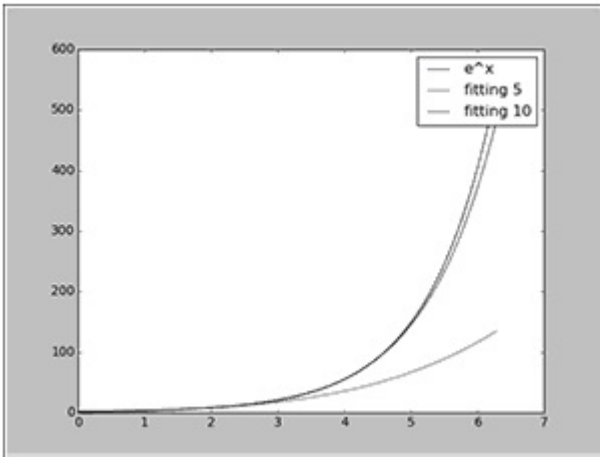

e^x
fitting 5
fitting 10
600
500
400
300
200
100
0
0
1
2
3
4
5
6
7

图6

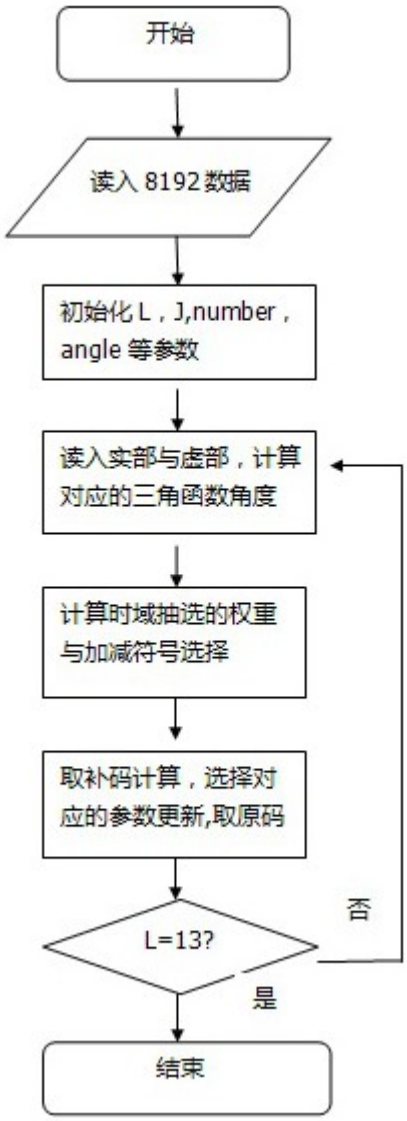

开始
读入 8192 数据
初始化 L，J,number，angle 等参数
读入实部与虚部，计算对应的三角函数角度
计算时域抽选的权重与加减符号选择
取补码计算，选择对应的参数更新,取原码
L=13?
否
是
结束

图7