# 利用Verilog将32位定点数转为16位半精度浮点数研究报告

# ——计算机组成原理第一次大作业

目录

[利用Verilog将32位定点数转为16位半精度浮点数研究报告 1](#_Toc52648833)

[——计算机组成原理第一次大作业 1](#_Toc52648834)

[一、 研究背景 2](#_Toc52648835)

[1. Verilog简介 2](#_Toc52648836)

[2. 定点数与浮点数 2](#_Toc52648837)

[（1） 定点数 2](#_Toc52648838)

[（2） 浮点数 3](#_Toc52648839)

[（3） 区别与联系 3](#_Toc52648840)

[二、 研究思路 3](#_Toc52648841)

[1. 定点数转浮点数的一般思路 3](#_Toc52648842)

[（1） 小数点移位 3](#_Toc52648843)

[（2） 配凑e并对“末位”取舍 3](#_Toc52648844)

[（3）数位对应 4](#_Toc52648845)

[三、 代码解析 4](#_Toc52648846)

[（1）声明变量 4](#_Toc52648847)

[（2）小数点移位 4](#_Toc52648848)

[（2）配凑e并对“末位”取舍 5](#_Toc52648849)

[（3）数位对应 5](#_Toc52648850)

[四、运行结果 5](#_Toc52648851)

### 研究背景

#### Verilog简介

Verilog HDL是一种硬件描述语言，用于从算法级、门级到开关级的多种抽象设计层次的数字系统建模。被建模的数字系统对象的复杂性可以介于简单的门和完整的电子数字系统之间。数字系统能够按层次描述，并可在相同描述中显式地进行时序建模。  
　　Verilog HDL 语言具有下述描述能力：设计的行为特性、设计的数据流特性、设计的结构组成以及包含响应监控和设计验证方面的时延和波形产生机制。所有这些都使用同一种建模语言。此外，Verilog HDL语言提供了编程语言接口，通过该接口可以在模拟、验证期间从设计外部访问设计，包括模拟的具体控制和运行。  
　　Verilog HDL语言不仅定义了语法，而且对每个语法结构都定义了清晰的模拟、仿真语义。因此，用这种语言编写的模型能够使用Verilog仿真器进行验证。语言从C编程语言中继承了多种操作符和结构。Verilog HDL提供了扩展的建模能力，其中许多扩展最初很难理解。但是，Verilog HDL语言的核心子集非常易于学习和使用，这对大多数建模应用来说已经足够。当然，完整的硬件描述语言足以对从最复杂的芯片到完整的电子系统进行描述。

定点数与浮点数

##### 定点数

定点小数是计算机处理的数值数据多数带有小数，小数点在计算机中通常有两种表示方法，一种是约定所有数值数据的小数点隐含在某一个固定位置上，称为定点表示法，简称

8位字长，第一位为符号位，小数点在第一位后面，后七位为具体数值，如： -0.1001原码表示为1.1001，反码为1.0110，为1.0111

所谓定点格式，即约定机器中所有数据的小数点位置是固定不变的。在计算机中通常采用两种简单的约定：将小数点的位置固定在数据的最高位之前，或者是固定在最低位之后。一般常称前者为定点小数，后者为定点整数。

##### 浮点数

浮点表示的设计者必须在尾数位宽和指数位宽之间找出折中的办法，因为字的大小是固定的，有一部分增加一位，则另一部分就要减少位。

折中是在精度和表示范围间进行权衡:增加小数部分会增加表示精度，而增加指数部分会增加数的表示范围。

浮点数通常是多个字的宽度。MIPS 中的浮点数表示如下: s为浮点数的符号，指数域为8位宽(包括指数的符号位)，尾数域为23位宽。这种表示称为符号和数值， 因为符号和数值的位置是相互分离的。

在16位半精度浮点数中，符号s占一位宽，指数域e占5个位宽，尾数域f占10个位宽。所表示的浮点数为：

(-1)^s\*(1+f)\*2^(e-15)

需要注意的是：当e为11111时，输出为无穷大；当e为00000时，输出为Subnorm

##### 区别与联系

定点表示法运算直观，但数的表示范围较小，不同的数运算时要考虑比例因子的选取，以防止溢出。浮点表示法运算时可以不考虑溢出，但浮点运算，编程较难。要掌握定、浮点数的转换方法及浮点数规格化方法。总而言之，定点表示法虽然简单，但比较死板，可能会浪费不必要的内存，也可能对数的表示不那么精确；而浮点数虽然灵活精确，但不易一目了然的读出，编程较难。

### 研究思路

#### 定点数转浮点数的一般思路

下面以一个32位定点数转32位浮点数为例对定点数转浮点数的步骤进行说明

##### 小数点移位

d=00011111111111111111111111111111(2)

=0.0011111111111111111111111111111(2)\*2^31

=1.1111111111111111111111111111000(2)\*2^(31-3)

##### 配凑e并对“末位”取舍

d≈1.11111111111111111111111(2)\*2^(127+31-3-127)

##### （3）数位对应

d=(-1)^s\*1.f\*2^(e-127)

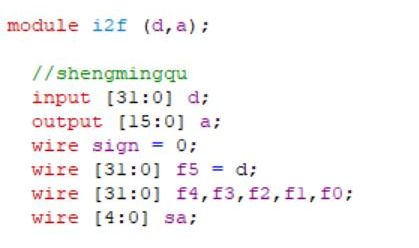
由上式及对应关系可知：s=0; e=127+31-3=10011011(2); f=11111111111111111111111(2)

a=01001101111111111111111111111111(2);

### 代码解析

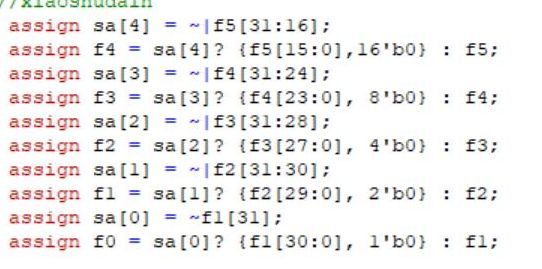
下面对实现该功能的Verilog编程语言进行解析

##### （1）声明变量



声明输入变量为32位定点数，输出变量为16位半精度浮点数，同时定义32位变量f4，f3,f2,f1,f0与5位变量sa。

##### （2）小数点移位

  
sa为偏移量，在这里我们将小数点先移到最左方，再将小数点根据需要从左往右移动

以第一步对sa[4]和f4赋值为例进行说明：

（1如果d前16位均为0，那么sa[4]为1，即小数点的偏移量为16，同时f4的前16位等于d后16位，f4的后16用0补齐。

（2 如果d前16位中有1，那么sa[4]为0，即小数点不进行偏移，f5的32位值全部对应赋给f4，进行下一轮的分析。

接下来进行8bite 4bite 2bite 1bite的判断与小数点移动，需要注意的是，最后一步的1bite判断，直接对f1[31]取反。

##### （2）配凑e并对“末位”取舍

C:\Users\mi\Documents\Tencent Files\2743941271\Image\C2C\JM]I[69DAUVM9H$2KOJV2PN.jpg

以上操作后的31-22位即为f；

在上文中提到，我们将小数点先移到最左端，再根据判断向右移，那么在偏移量为0的情况下，e=15+15=30，在有偏移量sa时，e=30-sa=1e(16)-sa。

##### （3）数位对应

C:\Users\mi\Documents\Tencent Files\2743941271\Image\C2C\VLRF]W{$AE2~ECY49BRDIO8.jpg

对d进行判断，如果d为0，那么a也为0，如果d不为零，那么a按照上文中变形整理出来的s，e，f按顺序排列转化为16位浮点数a。

### 四、运行结果

下列表格为我们测试产生的几组数据以及转换后的半精度浮点数的结果：

|  |  |
| --- | --- |
| 输入 | 输出 |
| 00000000000000000000000000000000 | 0000000000000000 |
| 10000000000000000000000000000000 | 0111100000000000 |
| 01111111111111111111111111111111 | 0111011111111111 |
| 00000000000000000000000000000100 | 0000010000000000 |
| 00000000000000000000000000000101 | 0000010100000000 |
| 00000000000000000000000000000001 | 0111110000000000 |
| 00000000000001000000000000000001 | 0100010000000000 |
| 00000000000010000000001100000001 | 0100100000000001 |