北京航空航天大學
BEIHANG UNIVERSITY

实验二、组合逻辑设计——定、浮点数转换

2018 年 11 月 12 日 （版本 v01）

序号	组长标记	教学班	学号	姓名	签名
1	*	162321	16231275	刘瀚骋	
2					
3					
4					
提交日期	2018-11-29		签收：	评价日期	

评阅 1	评阅 2	评阅 3	评阅 4	平均（百分制）

高等理工学院

目录

实验二、组合逻辑设计——定、浮点数转换（实验指导书部分）

1 实验指导

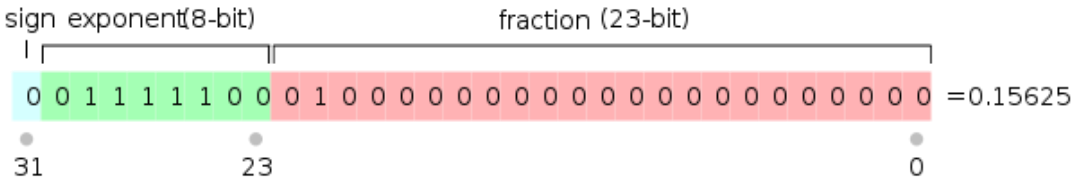
通过一个定点数转换为浮点数的案例，利用 EELAB-FPGACORE2 实验硬件实验平台（以下简称“实验板”）通过 Verilog HDL 语言实现可综合代码，加深对组合逻辑设计理念的认识。

1.1 背景知识——浮点数

1.1.1 IEEE 754 浮点数格式

浮点数是一种二进制的科学计数法（scientific notation），回顾常见的十进制科学计数法，例如：

$-103.24 = -1.0324 \times 10^{+2}$，其中模（或称为“base”）为 10，而计数法中的关键参数是：科学计数绝对值（含一位整数和小数点后的尾数，被称为“mantissa”），科学计数符号，幂指数（“exponent,”）的绝对值，幂指数的符号。二进制科学计数法只不过取模为 2，其四种参数与之相同，将该四种参数按照一定的格式以 32 位或 64 位（乃至更长）的二进制编码记录。



说明：本图来源于 wiki 百科，以$(0.15625)_{10}$，即$(0.00101)_2$的单精度浮点数为例

图 1 IEEE754 单精度浮点数的格式

常用的浮点数标准为 IEEE 754，其中单精度浮点数为 32 位，由 1-bit 符号位，8-bit 经过偏移（bias）的指数，以及 23 位尾数组成。其中（如图 1）：

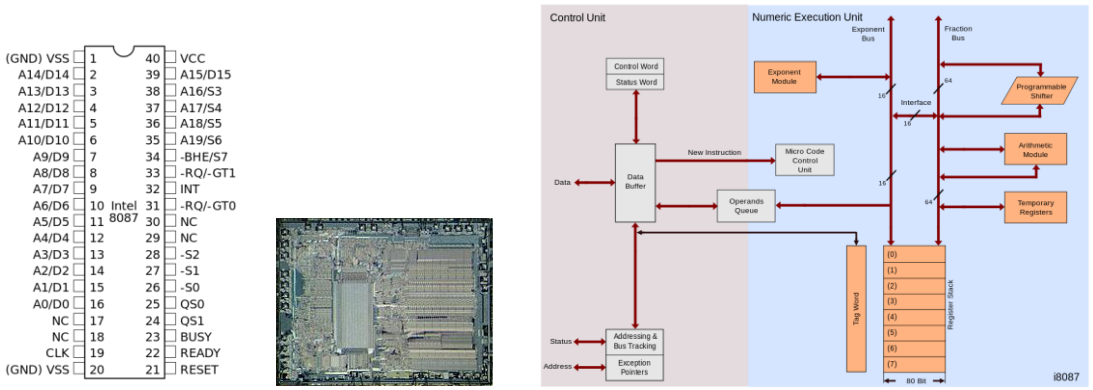
- 符号位：0 表示正数，1 表示负数；
- 指数部分：为了能够表达正指数和负指数，规定偏移量为 127，例如：对于$(0.00101)_2$，可以表达为二进制 1.01×2^{-3}，则指数为 127+(-3)=124，其自然二进制编码为 8’b0111_1100；
- 尾数部分：注意到对于二进制的科学计数值，其整数部分总是 1，所以该整数部分在 IEEE 754 标准中不被存储，只存储尾数的纯小数部分（fraction），例如：对于二进制 1.01×2^{-3}，

只用存尾数 01，后面补 0，如果是单精度浮点数，则为 23'b010_0000_0000_0000_0000_0000。

1.1.2 浮点数协处理器

定点数到浮点数，乃至浮点数到定点数的转换，可以编写程序运算，但更高效的方法是采用带有硬件实现的协处理器。在 PC 机发展之初，协处理器芯片和 CPU 芯片是分开的，这也决定了当时电脑的“档次”，例如：Intel 系列的不带协处理器的 8088 和带协处理器的 8086。

协处理器对定、浮点数的转换采用微码的方式实现，不严格地说它也相当于一个功能比较专用的小计算机，是 8087 协处理器的引脚图、模板图和简化的微体系架构模块图。



(a)芯片引脚 (b) 芯片模板 (c) 芯片微体系架构

图 2 8087 协处理器

随着芯片集成工艺的进步，目前将浮点数协处理功能集成到 CPU 芯片中。

微码实现的浮点数协处理器功能强大而灵活，但实际上定点数到浮点数的转换可以完全通过组合逻辑电路实现（反之亦然），只不过对于字长较大的浮点数，资源非常浪费。

1.2 案例：4-bit 无符号定点数转换为浮点数

定点数到浮点数的转换首先要确定定点数中第一个不为 0 的比特的位置，根据该位置确定指数的值，并且根据该位置确定将尾数移位的位数。而位置的确定用优先权编码器就可以胜任。

下面可以通过一个 4-bit 定点数（出于演示的目的，这里暂时只考虑无符号的正整数，指数部分不加偏移量，尾数部分也不去除整数的“1”）的浮点数转换的例题进行启发式说明。

可以采用优先权编码器（priority encoder）和数据选择器（multiplexer），将二进制定点数转换为二进制浮点数；如图 3 所示，以 4-bit 的二进制无符号整数输入为例，用 74148 优先权编码器和 74153 双四选一数据选择器芯片，配合必要的反相器门电路实现。（74148 和 74153 的功能表分别如表 1 和表 2 所

示。）

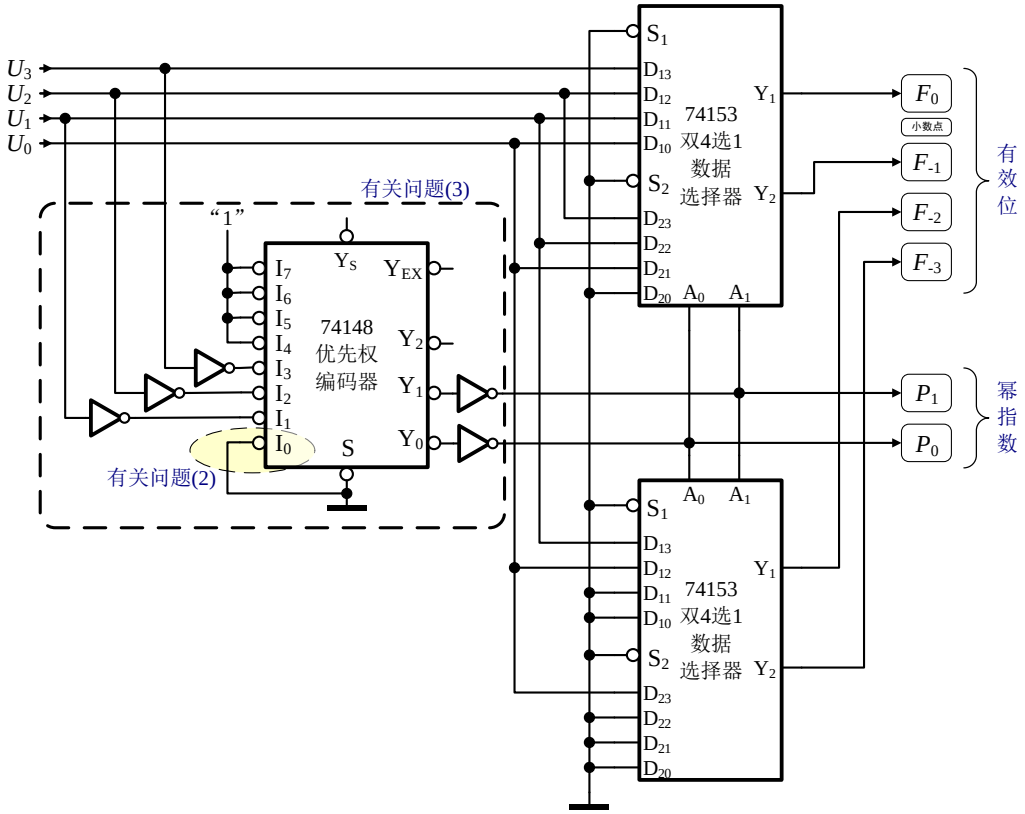


图 3 将 4-bit 的二进制无符号整数转换为二进制浮点数

表 1 优先权编码器 74148 的功能表

输入									输出				
$\overline{S}$	$\overline{I_0}$	$\overline{I_1}$	$\overline{I_2}$	$\overline{I_3}$	$\overline{I_4}$	$\overline{I_5}$	$\overline{I_6}$	$\overline{I_7}$	$\overline{Y_2}$	$\overline{Y_1}$	$\overline{Y_0}$	$\overline{Y_S}$	$\overline{Y_{EX}}$
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0	1
0	×	×	×	×	×	×	×	0	0	0	0	1	0
0	×	×	×	×	×	×	0	1	0	0	1	1	0
0	×	×	×	×	×	0	1	1	0	1	0	1	0
0	×	×	×	×	0	1	1	1	0	1	1	1	0
0	×	×	×	0	1	1	1	1	1	0	0	1	0
0	×	×	0	1	1	1	1	1	1	0	1	1	0
0	×	0	1	1	1	1	1	1	1	1	0	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0

表 2 双四选一数据选择器 74153 的功能表

输入			输出	输入			输出
$\overline{S_1}$	A_1	A_0	Y_1	$\overline{S_2}$	A_1	A_0	Y_2
1	×	×	0	1	×	×	0
0	0	0	D_{10}	0	0	0	D_{20}
0	0	1	D_{11}	0	0	1	D_{21}
0	1	0	D_{12}	0	1	0	D_{22}
0	1	1	D_{13}	0	1	1	D_{23}

请解决如下问题：

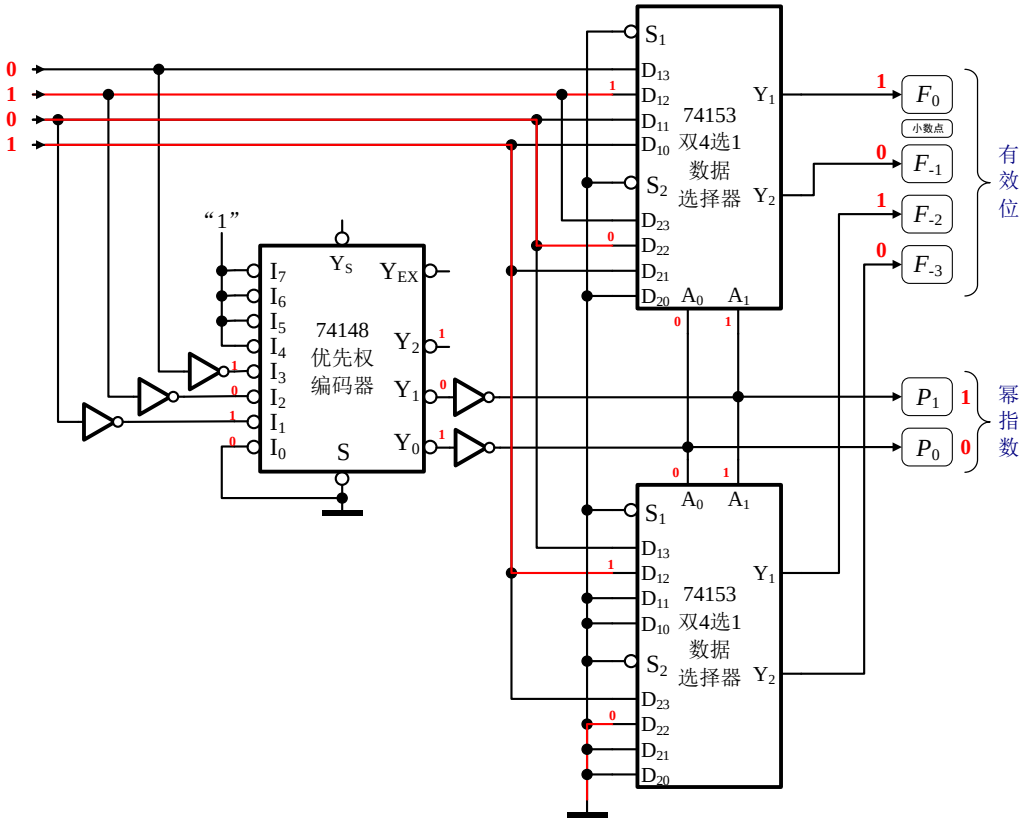
(1) 例如：输入的无符号整数 $U_3U_2U_1U_0$ 为$(0101)_2$，则输出的数字信号 $F_0F_{-1}F_{-2}F_{-3}$ 和 P_1P_0 分别为什么数值？

(2) 图 3 中的椭圆虚线框中所示的 74148 的 $\overline{I_0}$ 端为什么直接接地？（接地相当于逻辑“0”）

(3) 假设采购不到 74148 芯片，则仅依靠 74153 芯片（如必要，还可以用反相器），如何实现图 3 中方形虚线框中电路的功能？请**给出设计说明**，并**绘制**这部分功能电路的**原理图**。

解答：

(1) 如果输入的无符号整数 $U_3U_2U_1U_0$ 为$(0101)_2$，则输出的数字信号 $F_0F_{-1}F_{-2}F_{-3}$ 为$(1010)_2$，P_1P_0 为$(10)_2$；电路中的数值如下图所示。



(2) 中的 74148 芯片的 $\overline{I_0}$ 端直接接地，相当于接逻辑“0”，这是因为不论输入为$(0001)_2$还是$(0000)_2$，幂指数均应显示为$(00)_2$，所以使 $\overline{I_0}=0$，不论 U_0 是否为 1。

(3) 以 74148 优选权编码器为核心的电路功能是：

（逻辑取值，作答时可以不列出逻辑取值）：

U_3	U_2	U_1	U_0	P_1	P_0
1	×	×	×	1	1
0	1	×	×	1	0
0	0	1	×	0	1
0	0	0	1	0	0

可以分别用“四选一”实现 P_1 和 P_0 的组合逻辑函数。“四选一”的逻辑表达式为：$Y_i=[(\overline{A_1}\cdot\overline{A_0})\cdot D_{i0}+(\overline{A_1}\cdot A_0)\cdot D_{i1}+(A_1\cdot\overline{A_0})\cdot D_{i2}+(A_1\cdot A_0)\cdot D_{i3}]\cdot S_i$。

逻辑表达式为：

$P_1=U_3+\overline{U_3}\cdot U_2=U_3+U_2$，

$P_0=U_3+\overline{U_3}\cdot\overline{U_2}\cdot U_1=U_3+\overline{U_2}\cdot U_1$

可以取：

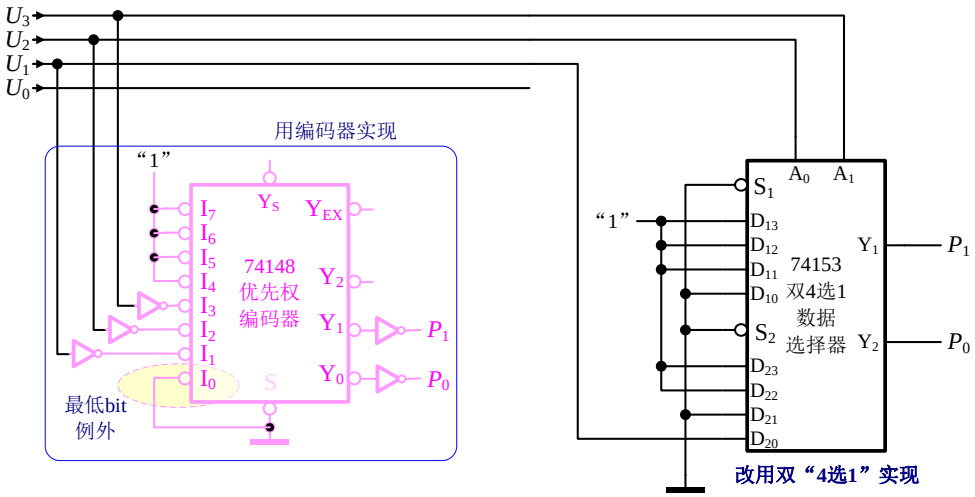
$$\begin{cases} A_1 = U_3 \\ A_0 = U_2 \\ D_{10} = 0 \\ D_{11} = D_{12} = D_{13} = 1 \\ Y_1 = P_1 \\ S_1 = 1 \end{cases}$$ ，使得：

$$P_1 = U_3 + U_2 = \overline{U_3} \cdot \overline{U_2} \cdot 0 + \overline{U_3} \cdot U_2 \cdot 1 + U_3 \cdot \overline{U_2} \cdot 1 + U_3 \cdot U_2 \cdot 1$$

$$\begin{cases} A_1 = U_3 \\ A_0 = U_2 \\ D_{20} = U_1 \\ D_{21} = 0 \\ D_{22} = D_{23} = 1 \\ Y_2 = P_0 \\ S_2 = 1 \end{cases}$$ ，使得：

$$P_0 = U_3 + \overline{U_2} \cdot U_1 = \overline{U_3} \cdot \overline{U_2} \cdot U_1 + \overline{U_3} \cdot U_2 \cdot 0 + U_3 \cdot \overline{U_2} \cdot 1 + U_3 \cdot U_2 \cdot 1$$

绘制电路图如下：



说明：恰好也不需要使用反相器。

1.3 基本实验要求

基本实验要求为：

a) 采用门级建模（严格门级，不能用 RTL 数据流方法）的方法，模仿优先级编码器 74148 和双

四选一多路复用器 74153 的功能实现单元电路模块；

b) 结合对 74148 和 74153 模块的实例化，采用门级建模（严格门级，不能用 RTL 数据流方法）方法，实现 4-bit 无符号整数到浮点数转换功能；

c) 利用逻辑函数化简方法，对 4-bit 无符号整数到浮点数转换逻辑进行求解，然后采用门级建模（严格门级，不能用 RTL 数据流方法）方法，实现 4-bit 无符号整数到浮点数转换功能；

d) 不刻意地用优先权编码器或数据选择器的概念，采用行为级建模方法，实现 4-bit 无符号整数到浮点数转换功能；

e) 分别对上述 3 种实现方式进行功能仿真；

f) 利用“实验板”对上述 3 种 4-bit 无符号整数到浮点数转换电路进行**综合和实现**，设定定点数输入和浮点数输出的人机接口，建议用 4 个 LED 灯表示输入值，操作开关或按动按钮后进行转换，用数码管显示有效位和幂指数；（任何合理的人机接口都是可以接受的）；

g) 对上述 3 种方法综合后的资源占用情况进行分析，了解或估算不同编程方法下的实现所消耗的可编程逻辑资源。

注意：本实验将采取**实验报告和现场汇报演示相结合**的形式，实验报告上交的具体**截止期限**请注意任课教师的通知，实验报告格式请参考本文档；**现场汇报演示**的时间和分组情况请注意任课老师的通知。特此广而告之。

1.4 扩展要求

h) 更多位定点数转浮点数的设计，比如实现 5 位整数定点数转浮点数，并考虑浮点数的有效位只有 3 位（即 F_0,F_{-1},F_{-2}）、幂指数也扩大到 3 位（即 P_0,P_1,P_2）的情况。在这种情况下就会出现精度缺失的问题，比如：输入为$(01001)_2$时，则浮点数只能表示为$(1.00\times(10)^{11})_2$，但当输入为$(00101)_2$时，浮点数为精确数$(1.01\times(10)^{10})_2$。依然采用 74148 优先权编码器和 74153 双四选一选择器，结合基本非门电路，能否实现给定电路功能？最少需要几片芯片，给出相关设计结果。结合对 74148 和 74153 模块的实例化，采用门级建模（严格门级，不能用 RTL 数据流方法）方法，实现 5-bit 无符号整数到浮点数转换功能。第 5 位输入可以采用开关或按动按钮实现；

i) 如果尝试更多位定点数转浮点数的设计，例如：实现 8 位整数定点数转换为浮点数，同样浮点

数有效位也只有 3 位、幂指数也为 3 位的情况，如果仍采用 74148 优先权编码器和 74153 双四选一选择器，请估算所用芯片的规模与个数，最少需要几片芯片？做出理论分析即可。

j) 不刻意地用优先权编码器或数据选择器的概念，采用行为级建模方法，实现 8 位整数定点数转换为浮点数功能，在可编程器件上进行实现，并合理设计人机接口；

k) 对门级 4-bit、门级 5-bit、行为级 8-bit 的综合后资源占用情况进行分析，了解或估算不同规模下不同编程方法下的实现所消耗的可编程逻辑资源情况，给出适当定量分析结论。

1.5 其它提示

在实验中心提供的样例程序中，UART 的样例源代码具有固定的 9600 波特率、8 位数据位、偶校验、1 位停止位，缺点是一个只能每次读写一个字节的；更好的 UART（注：更稳定可靠、波特率可配置、有 Telnet 等上层协议）需要用户自行开发，或者在网上搜索更好的代码加以改造。

“实验板”的 I/O 接口和人机接口资源请参考《digiC2018 课程实验实验指导书(01)》。

实验二、组合逻辑设计——定、浮点数转换（实验报告部分）

2 实验报告

2.1 实验背景与需求分析

本实验要求将一个四位的定点数转换为四位浮点数。由于实验中采用简化后的 IEEE-754 浮点数格式，不考虑符号位，指数偏移以及省略尾数中的‘1’，因此将四位定点数转换为四位浮点数，实际上就是实现对 4 位定点数，在最高位为 0 时，进行逻辑左移，直到最高位为非零数后，则得到浮点数尾数部分，而逻辑左移的位数即是浮点数的指数部分。

根据实验要求，将采用优先编码器+多路复用器；逻辑函数化简以及行为级建模三种方式实现定点向浮点的转换。

本人数字电路实验源代码托管地址为 https://github.com/CNLHC/DigiC2018/。由于报告篇幅有限，我会在后文中可能需要引用源代码的地方直接插入指向该地址内相应文件的链接。

2.2 系统设计

2.2.1 总体设计思路

使用数码管和 LED 灯来展示数据，使用按键进行交互。通过按键可以使内部循环产生 2’b0000-2’b1111 的定点数。将定点数转换为浮点数后，使用两位数码管分别显示定点数和转换后的浮点数的尾数的十六进制字面量；将浮点数的指数通过 LED 灯的后两位显示。

采用自底向上的方法进行实现。首先根据实验要求，分别使用优先编码器+多路复用器；逻辑函数化简以及行为级建模三种方案实现三个定点转浮点的转换模块。其次，为满足人机交互需求，复用实验一中的数据生成器，按键去抖器，此外由于要在两段数码管上显示不同的数据，需要额外编写段扫描驱动模块。

2.2.2 接口设计

定点转浮点模块的输入为定点数数据（采用行为级建模实现的转换模块还需输入时钟及复位信号），输出为转换后的尾数和指数。其他模块的接口设计是平凡的，不再赘述。

2.2.3 定点-浮点转换模块

下面分别介绍基于优先编码器+多路复用器；逻辑函数化简以及行为级建模三种方式实现的定点浮点转换模块。

基于优先编码器+多路复用器的实现

根据实验指导书中的描述，基于优先编码器和多路复用器可以实现定点数和浮点数的转换。根据74HC148 优先编码器的真值表，可以写出其简化后的逻辑关系式

$$\begin{cases} Y_2' = \big((I_4 + I_5 + I_6 + I_7)S\big)' \\ Y_1' = ((I_2 I_4' I_5' + I_3 I_4' I_5' + I_6 + I_7)S)' \\ Y_0' = \big((I_1 I_2' I_4' I_6' + I_3 I_4' I_6' + I_5 I_6' + I_7)S\big)' \\ Y_S' = (I_0' I_1' I_2' I_3' I_4' I_5' I_6' I_7' S)' \\ Y_{EX}' = ((I_0' I_1' I_2' I_3' I_4' I_5' I_6' I_7' S)')' \end{cases}$$

根据 74HC153 二选一多路复用器的真值表，可以写出其简化后的逻辑关系式

$$\begin{cases} Y_1 = [D_{10}(A_1' A_0') + D_{11}(A_1' A_0) + D_{12}(A_1 A_0') + D_{13}(A_1 A_0)]S_1 \\ Y_1 = [D_{20}(A_1' A_0') + D_{21}(A_1' A_0) + D_{22}(A_1 A_0') + D_{23}(A_1 A_0)]S_2 \end{cases}$$

根据上述的逻辑关系式，可以用门电路分别实现具有和 74HC148，74HC153 功能相同的模块，进而可以使用实验指导书中给出的方式实现定点数向浮点数的转换。

代码参考:

1. 74HC148 实现 Gate148.v
2. 门级 74HC153 实现 Gate153.v
3. 基于优先编码器和多路复用器的定点-浮点转换器实现 ToFloat.v

基于门电路的实现

由于本实验中，4 位定点数向浮点数的转换完全是组合逻辑，因此可以写出其真值表，并由真值表计算出简化后的逻辑关系式，最后根据逻辑关系式，使用门电路实现。

列出真值表的操作是平凡的，在此不再赘述，如需检查真值表，请参考本次实验的源代码仓库中truthTable。

为了简化操作，使用 Python 编写了基于 ESPRESSO-II[1]算法的真值表化简工具。请参考源码仓库目录下的 mini.py 文件获取更多信息。化简后的结果如下

$$\begin{cases} F_3 = D_0 + D_2 + D_3 + D_1 D_2' \\ F_3 = D_2 D_2 + D_0 D_1 D_3' + D_1 D_2 \\ F_1 = D_1 D_3 + D_0 D_2 D_3' \\ F_0 = D_0 D_3 \\ P_1 = D_2 + D_3 \\ P_0 = D_3 + D_1 D_2' \end{cases}$$

根据上述逻辑关系式，可以更具门电路实现定点数向浮点数的转换。

代码参考:

1. 门级定点-浮点转换器实现 PureGateToFloat.v

基于行为级建模的实现

使用行为级建模实现定点到浮点的转换，基本思路是将转换的过程看作是一个逻辑左移的过程。最后的输出是连续左移后最高位不为 0 时的数据以及左移的相对位移量，分别作为转换后浮点数的尾数和指数。

该模块内部实际上实现了一个小型状态机,其转换过程可用图 4 简单描述

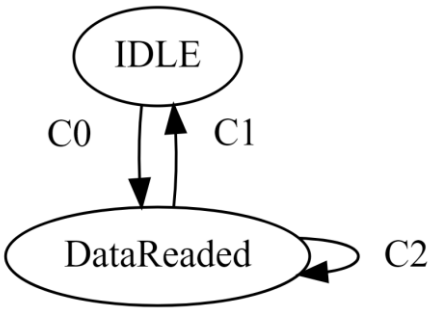


图 4 状态转换示意图

其内部定义了两个状态:IDLE 和 DataReaded，当状态为 IDLE 时，内部一个 Buffer 的数据会和输入数据作比，如果二者不同，则判定为新数据输入，将输入数据存储到合适的寄存器中，并触发 C0，状态切换到 DataReaded；当状态为 DataReaded 时，如果位移寄存器内的数据首位为 0，则将数据左移一位，并增加位移计数器，此时触发 C2，内部状态不变；当状态为 DataReaded 时，如果位移寄存器内的数据首位为非 0 项，则将位移计器和位移计数器中的值更新到输出缓冲区内，并触发 C1，状态切换到 IDLE。该模块会响应全局异步复位信号，将状态置为 IDLE。

代码参考: 行为级定点-浮点转换器实现 RTLToFloat.v

2.2.4 数码管扫描驱动模块

由于实验板的两个数码管共用段选信号，因此为了能使两个数码管显示不同的信息，故编写此模块。

其工作原理是在很短的时间内快速切换数码管的位选信号，并驱动数码管显示相应的数据。尽管理论上同一时刻只有一个数码管被点亮，但由于切换的速度极快以至于人眼无法捕捉到数码管的变化，因此数码管看上去同时显示了两个数据。

代码参考：行为级定点-浮点转换器实现 boundedScanComDriver.v

2.3 功能仿真测试

2.3.1 测试程序设计

分别为每一个模块编写 TestBench 进行测试。本人习惯于将 TestBench 和普通代码编写在同一个文件内，因此可以在模块对应的源代码中（重要模块的源代码已于 2.2.3 **定点-浮点转换模块**给出）检阅相应测试程序的具体实现。

2.3.2 功能仿真过程

在此列出定点转浮点模块的仿真过程。

2.3.2.1 基于优先编码器+多路复用器方案的仿真

使用 TestBench 激励模块并观察相应的输出。测试了输入分别为 4’b0101,4’b0111,4’b0011,4’b0001 时的结果，相应的尾数输出分别是 4’b1010, 4’b1110, 4’b1100,4’b1000，相应的指数输出分别是 2’b10,2’b10,2’b01,2’b00。输出符合预期。测试结果如图 5 所示。

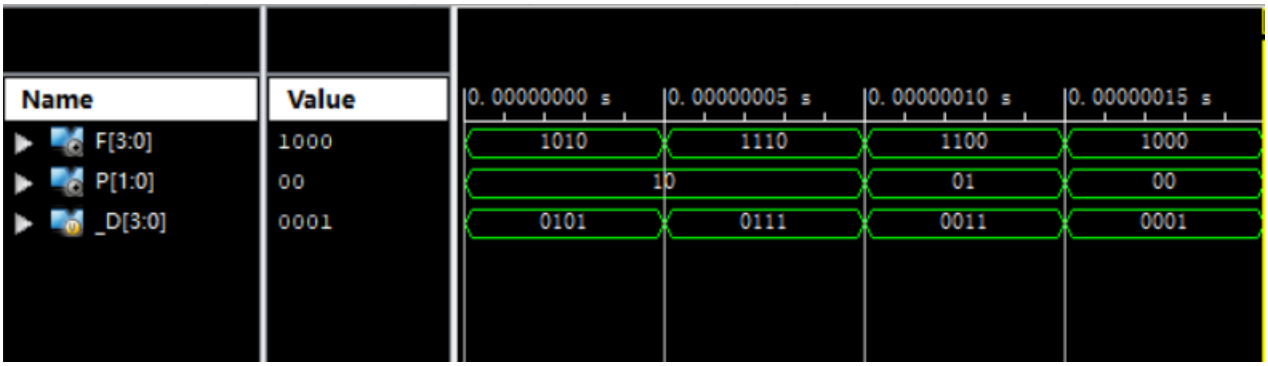


图 5 基于优先编码器+多路复用器方案的仿真结果

2.3.2.2 基于门电路方案的仿真

由于本模块外部接口相对于采用优先编码器+多路复用器方案的模块完全一样，故仿真时采用和2.3.2.1 基于优先编码器+多路复用器方案的仿真时相同的测试集。图 6 为基于门电路方案的仿真结果。

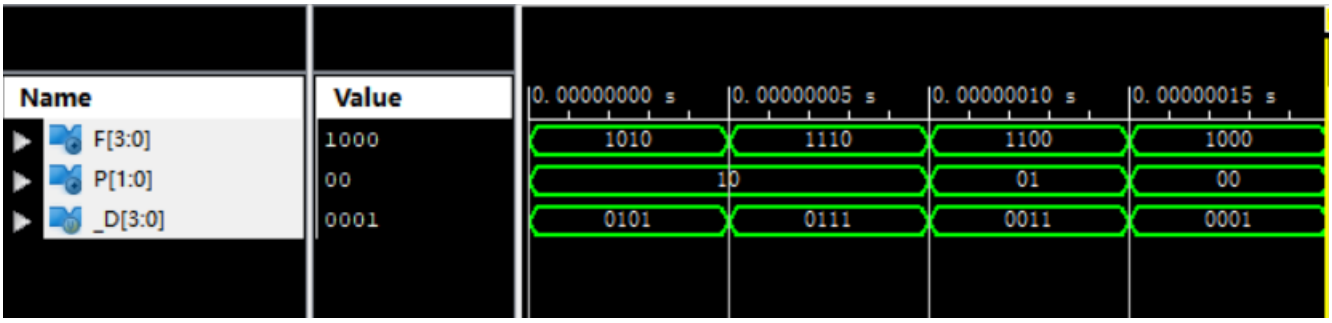


图 6 基于门电路方案的仿真结果

2.3.2.2 基于行为级建模方案的仿真

使用与前文所述相同的测试集测试模块。首先模块在异步复位信号的作用下进行复位，之后将输入数据(_D)转换为相应的浮点数和尾数。由测试结果图 7 可以看出，从模块中获得了理想的输入。

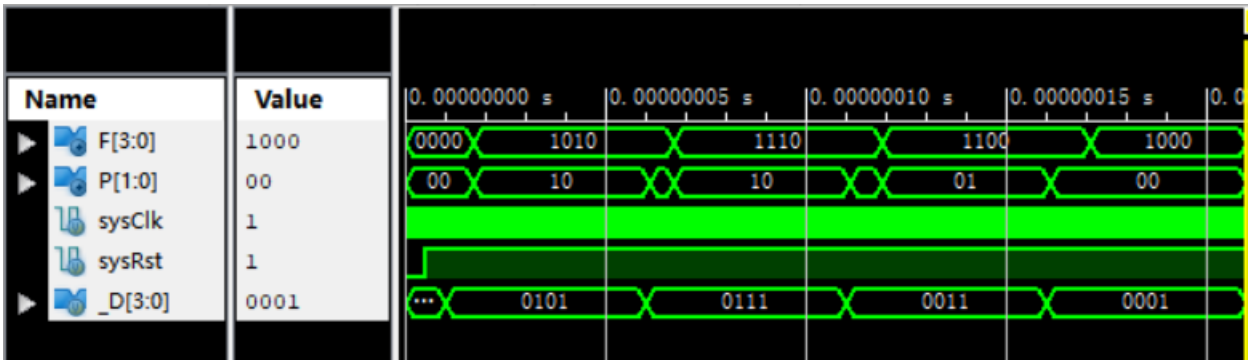


图 7 基于行为级建模方案的仿真

使用行为级建模，由于需要逐位进行逻辑左移操作，因此转换并不是在一个时钟周期内完成。这将带来一些问题，例如图 8 所示，由于在第一个游标处数据已经发生了改变，为了能够统计正确的位移数，所以位移计数器会被清零。但此时转换尚未完成，输出仍然保持上个状态未改变。这样就造成了一个短暂的错误输出窗口。

当时钟频率远远大于输入数据更新的频率时，该问题是可被忽略的。

通过设置多级输出流水线可以完全消除上述错误输出窗口，但是在本次实验中，由于数据更新的频率远远小于时钟频率，因此正如上文所言，该问题是可被忽略的。

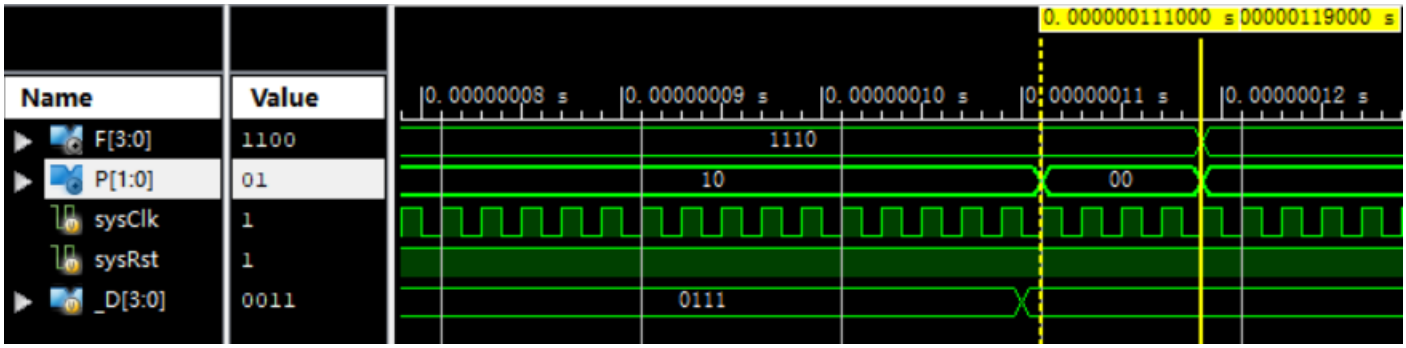


图 8 输出延迟带来的问题

2.3.2 实验关键结果及其解释

已在仿真**功能仿真过程**一节中，配合仿真结果进行了说明。

2.4 设计实现

2.4.1 综合和下载过程

代码编写完成后，点击 ISE 程序主界面左侧的 Synthesis 按钮进行综合与测试。测试完成后，点击 Generating Programming File 按钮生成下载所需的配置文件。

下载时，首先连接实验板，通过 ISE iMPACT 工具配置 Daisy Chain。在测试阶段通过.bit 文件直接编程 FPGA；测试完成后，通过 bit 文件生成 MCS 文件，编程实验板上的非易失性存储器。

此外，根据要求，对使用三种不同的实现方案时系统资源的占用进行对比。

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	55	9,312	1%	
Number of 4 input LUTs	63	9,312	1%	
Number of occupied Slices	61	4,656	1%	
Number of Slices containing only related logic	61	61	100%	
Number of Slices containing unrelated logic	0	61	0%	
Total Number of 4 input LUTs	111	9,312	1%	
Number used as logic	63			
Number used as a route-thru	48			
Number of bonded IOBs	18	66	27%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	2.44			

图 9 使用方案一的系统资源占用

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	55	9,312	1%	
Number of 4 input LUTs	63	9,312	1%	
Number of occupied Slices	61	4,656	1%	
Number of Slices containing only related logic	61	61	100%	
Number of Slices containing unrelated logic	0	61	0%	
Total Number of 4 input LUTs	111	9,312	1%	
Number used as logic	63			
Number used as a route-thru	48			
Number of bonded IOBs	18	66	27%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	2.44			

图 10 使用方案二的系统资源占用

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	73	9,312	1%	
Number of 4 input LUTs	76	9,312	1%	
Number of occupied Slices	73	4,656	1%	
Number of Slices containing only related logic	73	73	100%	
Number of Slices containing unrelated logic	0	73	0%	
Total Number of 4 input LUTs	122	9,312	1%	
Number used as logic	76			
Number used as a route-thru	46			
Number of bonded IOBs	18	66	27%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	2.58			

图 11 使用方案三的系统资源占用

比对图 9 图 10 图 11 中的数据，可以看出，使用门级实现的两种方案消耗的系统资源一样多；使用行为级建模消耗的系统资源最多，相对于使用门级的方案，多使用了 13 个 4 输入查找表，以及 18 个 Flip Flops。使用行为级建模实现需要耗费更多的系统资源，而且运行速度较慢。

2.4.2 实验关键结果及其解释

程序下载后，点击开发板上的按钮，数码管 1 会循环显示‘0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F’，表示待转换定点数的十六进制字面量；数码管 2 会循环显示‘0-8-8-C-8-A-C-E-8-9-A-B-C-D-E-F’表示转换后的尾数；LED 灯的最后两位状态按以下顺序循环‘00-00-01-01-10-10-10-10-11-11-11-11-11-11-11-11’表示转换后的指数。

2.4 小结

本次实验使用不同的方法实现了 4 位定点数向浮点数的转换，并对不同方案最终占用系统的资源进行了分析。

参考文献

[1] R. Brayton, G. Hatchel, C. McMullen, and A. Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Publishers, Boston, MA, 1984.