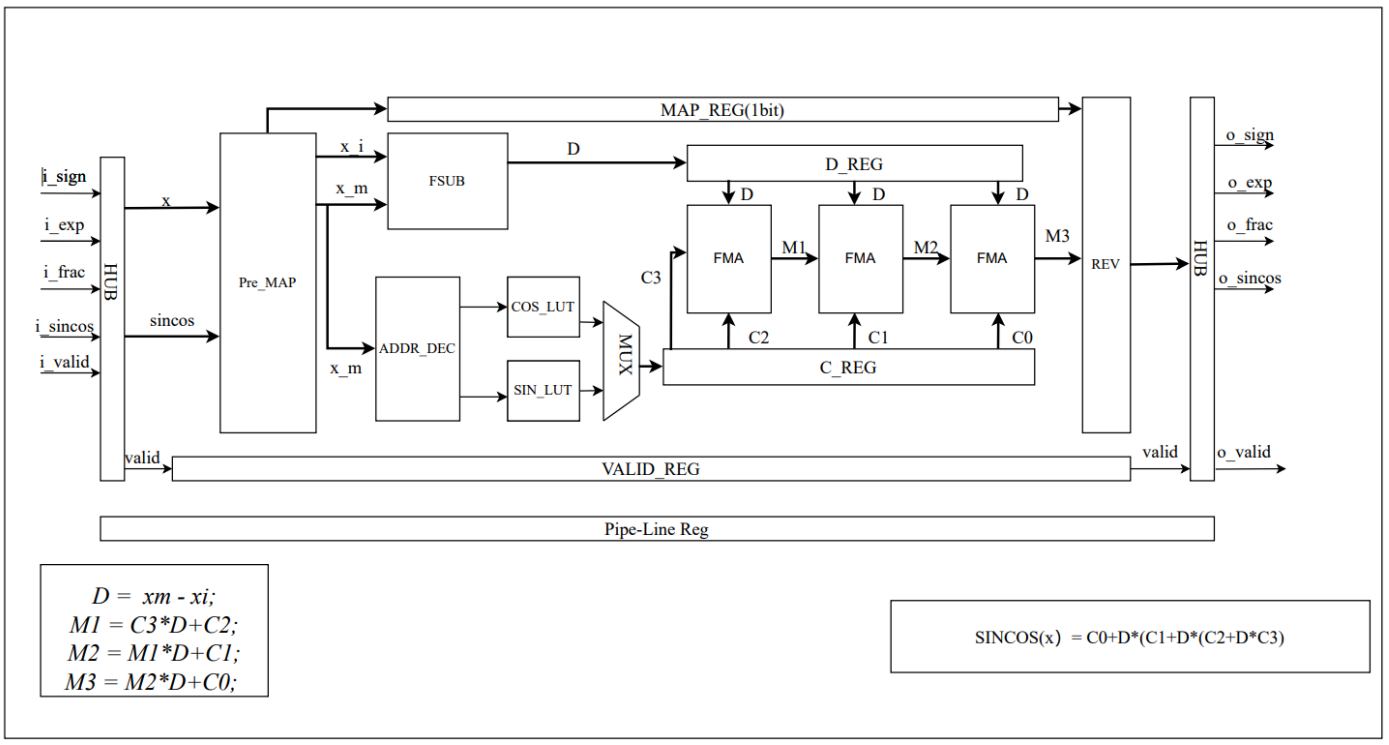
整体架构



子模块定义

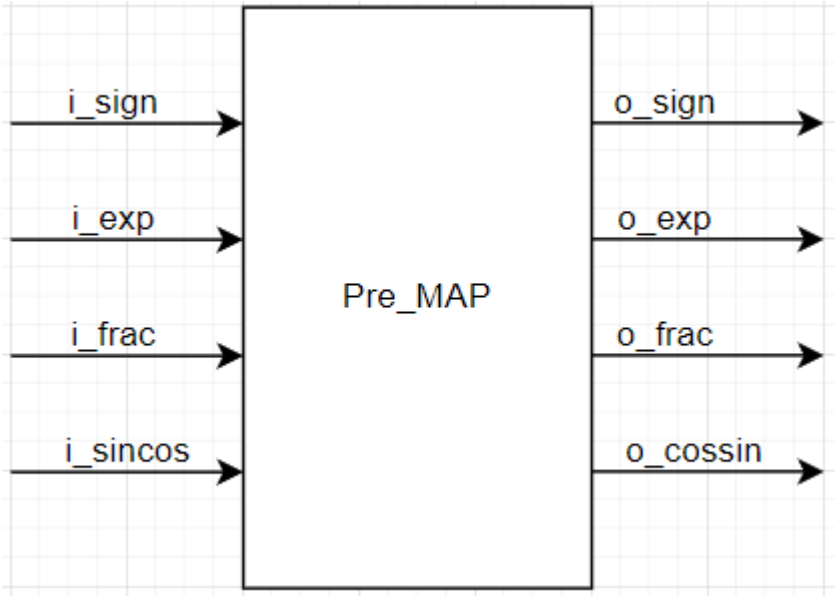
1.Pre-MAP

1.1 总体说明

预映射模块，在实现算法的过程中，前期处理主要是进行计算区域的映射，对于输入需要限定在$x \in [0,1]$，并将其压缩在$[0,0.125]$区间范围内。在进行压缩时，首先要选择模式的压缩，以及压缩的计算模式、压缩的偏置数（bias）。

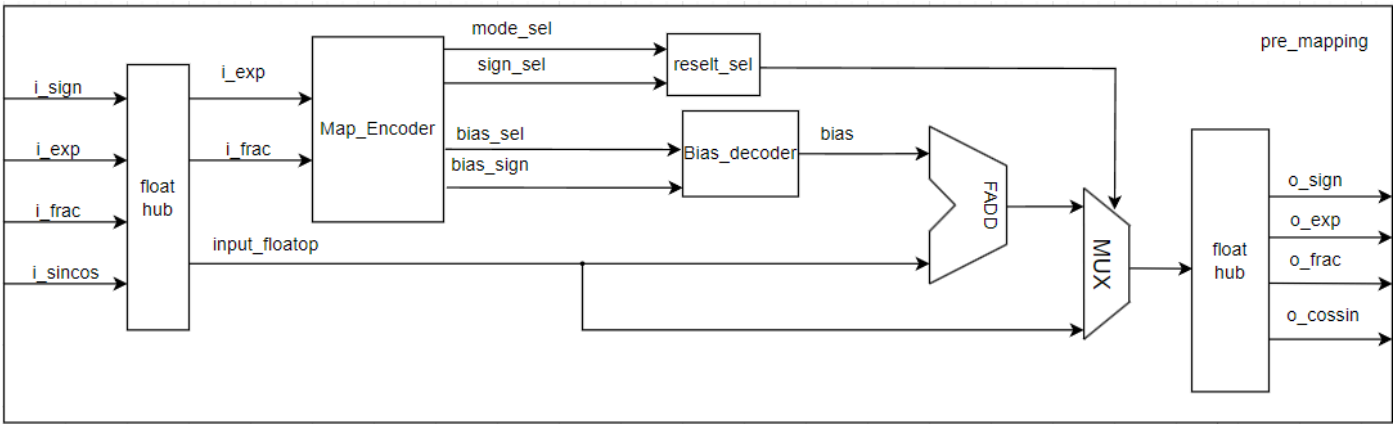
pre-mapping module.For the impelement of design, we need to compress the district of caculation. Firstly, the input number should be confined in [0,1],then being processed by this module. the input will be mapped to the [0,0.125]

其总体模块定义如下：



经过运算区域的压缩和变换，cos和sin的计算模式也可能发生变化

1.2 总体架构

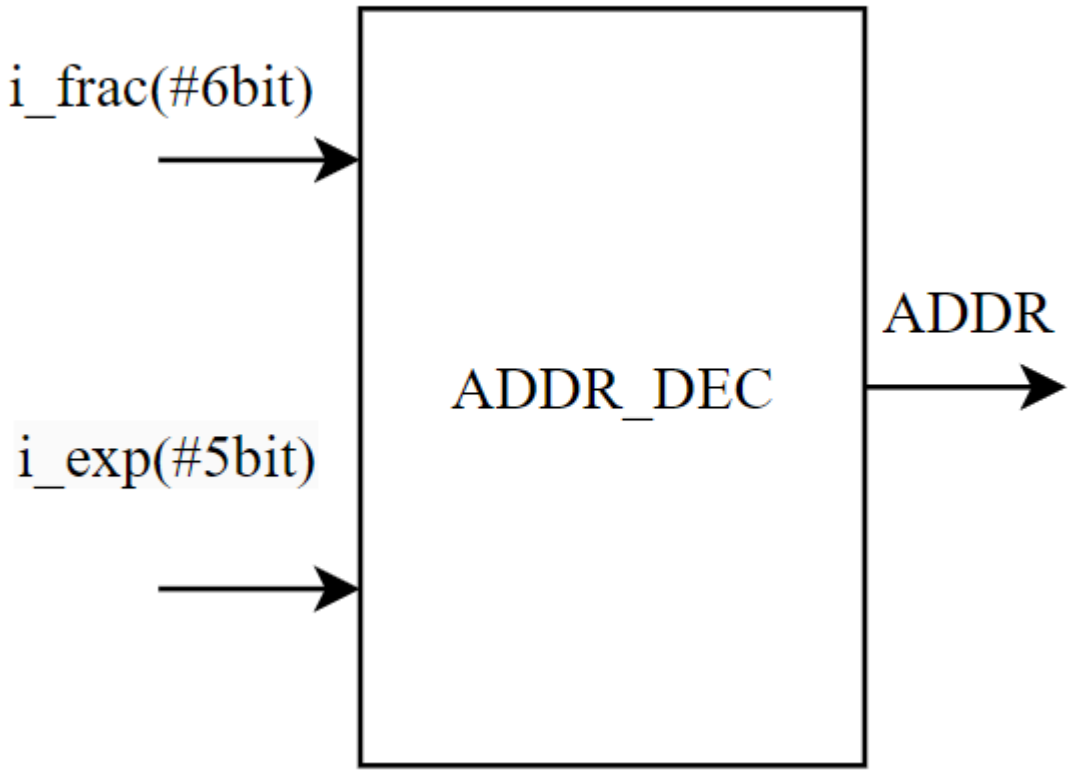


总体架构如图，注意，float hub并不是实际模块，只是连线上的操作，为了画图简单。

其中先利用exp与frac位进行映射区域的编码，映射区域总共有8个区域，4种偏置数。其种有一种区域不需要进行额外的变换与压缩，直接可以输入MUX进行结果的选择。另一条较为复杂的路径则是需要进行压缩与映射，对映射区域进行编码后，将编码后的bias相关信号送入bias解码模块，利用编码进行还原，还原出完整的bias的浮点数格式，送入浮点加法器进行区域的映射。区域如下：

cos	[0,0.125]	$\cos(x_f)$
	(0.125,0.25]	$\sin(0.25 - x_f)$
	(0.25,0.375]	$-\sin(x_f - 0.25)$
	(0.375,0.5]	$-\cos(0.5 - x_f)$
	(0.5,0.625]	$-\cos(x_f - 0.5)$
	(0.625,0.75]	$-\sin(0.75 - x_f)$
	(0.75,0.825]	$\sin(x_f - 0.75)$
	(0.825,1)	$\cos(1 - x_f)$
sin	[0,0.125]	$\sin(x_f)$
	(0.125,0.25]	$\cos(0.25 - x_f)$
	(0.25,0.375]	$\cos(x_f - 0.25)$
	(0.375,0.5]	$\sin(0.5 - x_f)$
	(0.5,0.625]	$-\sin(x_f - 0.5)$
	(0.625,0.75]	$-\cos(0.75 - x_f)$
	(0.75,0.825]	$-\cos(x_f - 0.75)$
	(0.825,1)	$-\sin(1 - x_f)$

2. ADDR_DEC



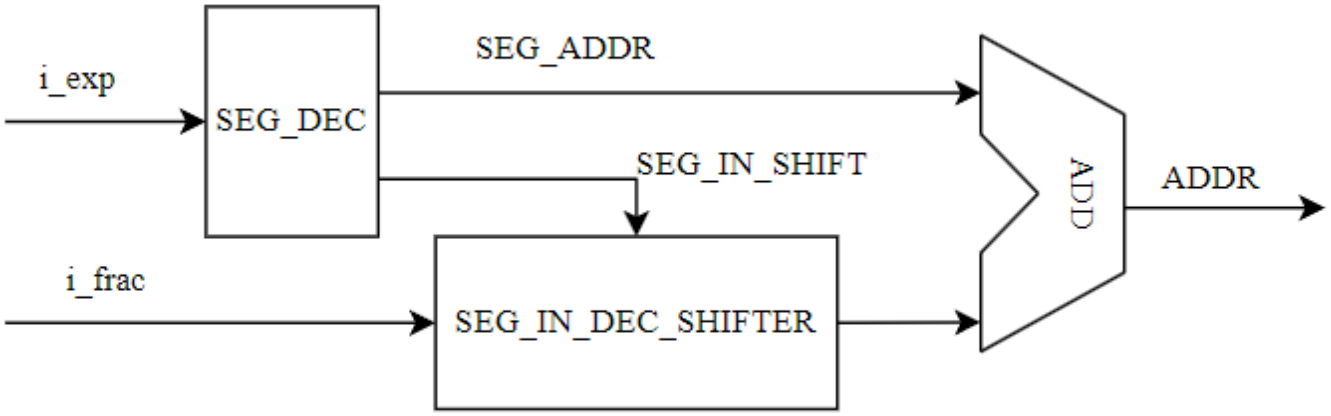
2.1 总体说明

译码模块需要将输入的浮点数数据映射到其对应的系数表中，根据系数个数84，总共需要7位地址码，可以分为四段，如果直接根据小数对其进行译码，复杂程度很高，电路规模会很大。为了将其进一步划分，减少复杂程度，可以对其进行段首地址与段偏移地址的译码。各段如下：

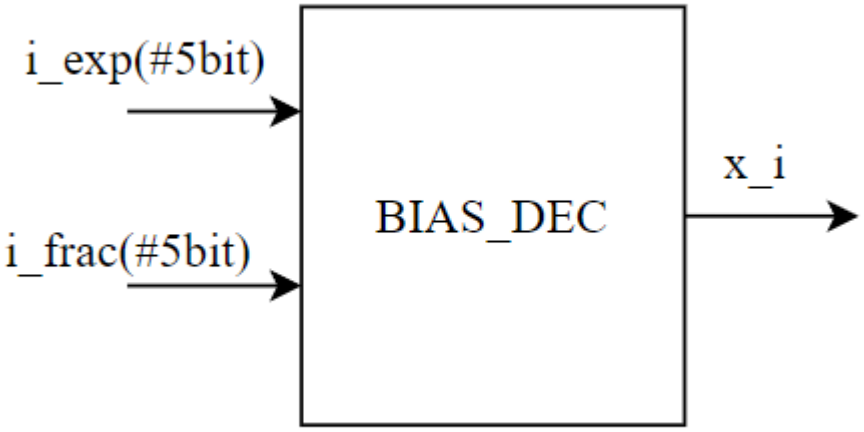
Lower Bound	Upper Bound	Segment	D(distance)	n
0	2^{-20}	$in : x; out : 2\pi x$		1
2^{-20}	2^{-15}	$[2^{-20}, 2^{-15}]$	Upper-Lower	1
2^{-15}	2^{-13}	$[2^{-15}, 2^{-13}]$	Upper-Lower	1
2^{-13}	2^{-11}	$[2^{-13}, 2^{-11}]$	Upper-Lower	1
2^{-11}	2^{-9}	$[2^{-11} + (n-1)D, 2^{-11} + nD]$	2^{-11}	1~3
2^{-9}	2^{-5}	$[2^{-9} + (n-1)D, 2^{-9} + nD]$	2^{-10}	1~30
2^{-5}	2^{-3}	$[2^{-5} + (n-1)D, 2^{-5} + nD]$	2^{-9}	1~48

可以根据段数分析，其最大偏移地址为48，则需要6个位宽的偏移地址。

2.1 总体架构

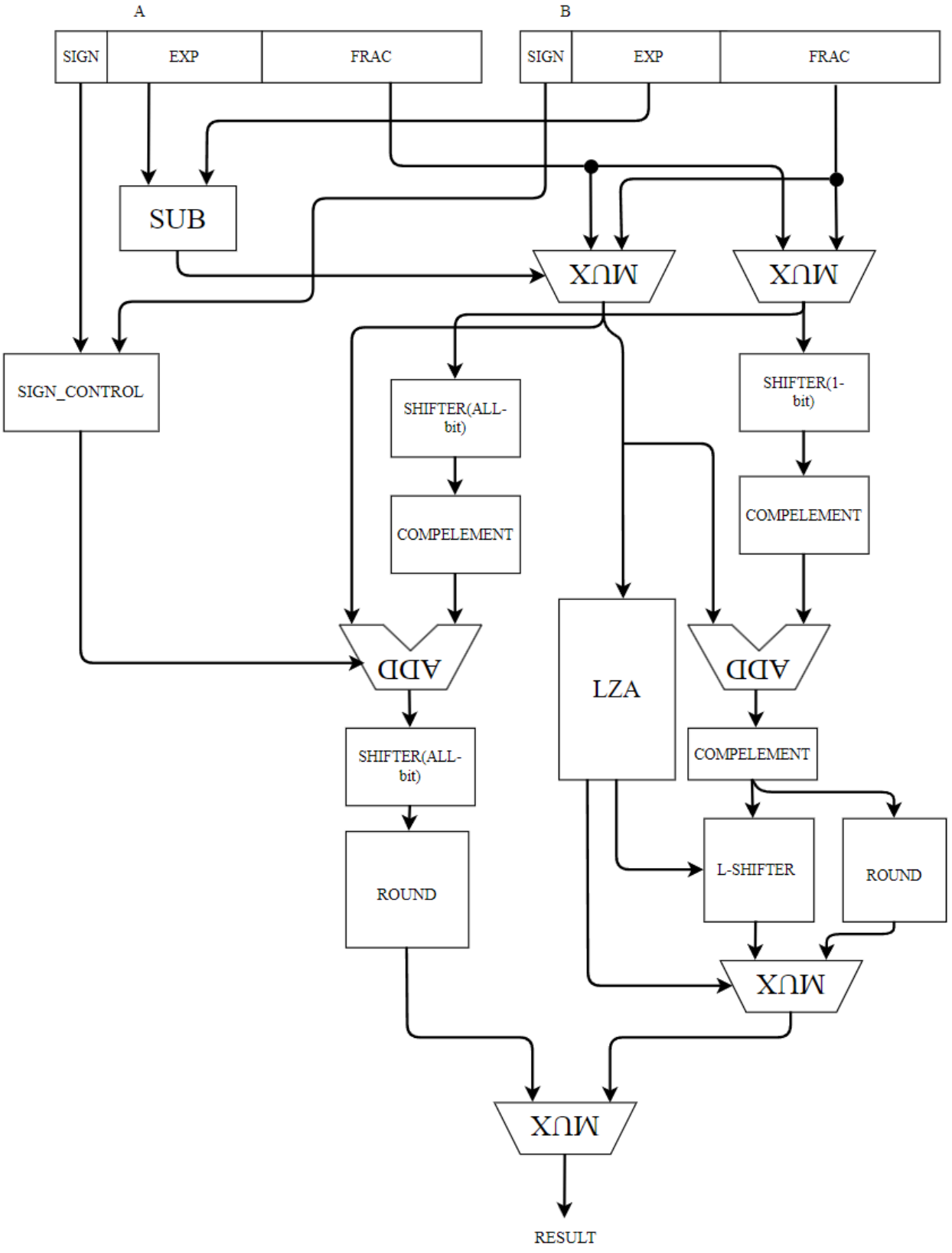


3.BIAS_DEC



主要恢复偏移数x_i，其输入和ADDR_DEC以及中间的段地址信号可以复用。

4. FADD

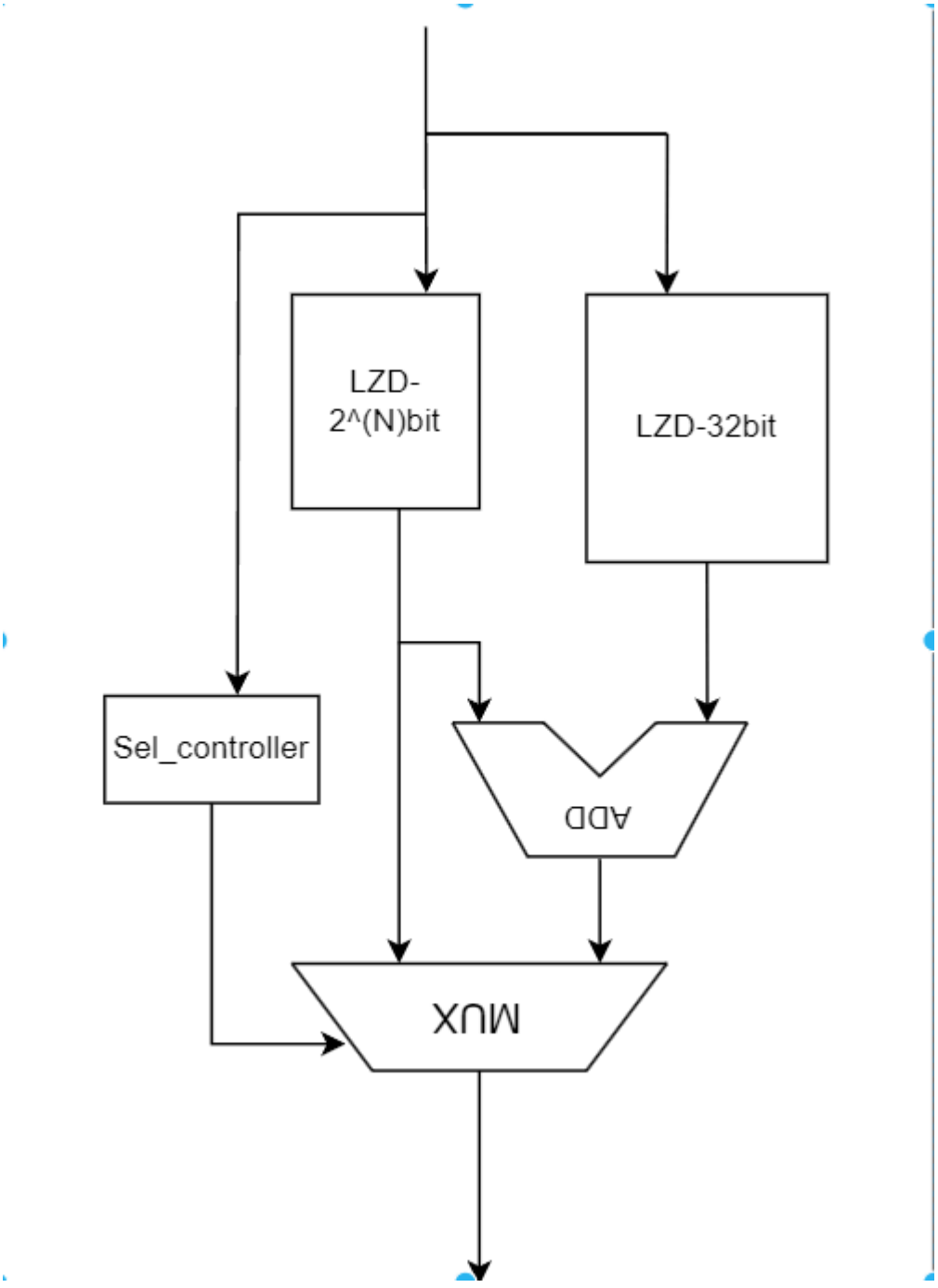


4.1 总体结构及说明

采用双路径算法，其中对于双路径算法进行进一步优化，通过分析LZA的结果，可以判断出那些不需要舍入，进而进一步优化逻辑。

5. LZD(leading zero detector)

5.1 总体架构及说明

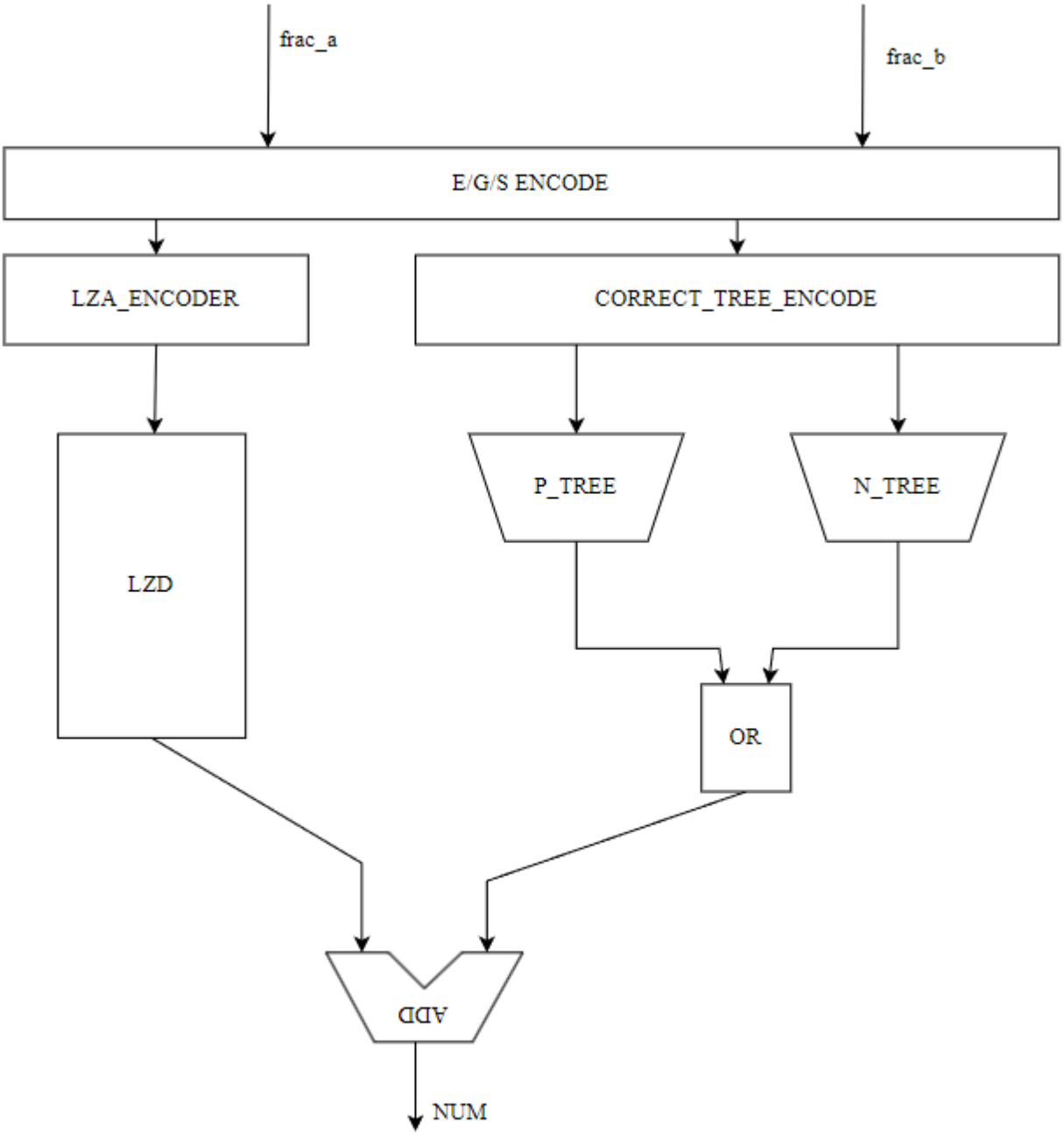


对于非2^n次方bit的信号可以进行如下的方法进行前导零检测，每一个模块都用二叉树结构进行检测。

6. LZA(leading zero anticipator)

6.1 总体架构以及说明

前导零预测包含两部分，前导零预测，前导零纠错，其整体结构如下：



7. FMA(Fused Multiply-Add Unit)

主要是进行乘法加法的运算,接口定义与FADD类似。

验证环境的搭建

验证环境的搭建主要是各个组件的实现，为了尽可能复用各个组件，在设计时尽可能细颗粒度实现各个组件，由于设计上输入格式主要是规定的浮点数，可以把各个模块分为三类，第一类是：INPUT 2 OUTPUT 1，这一类是FADD、FMUL等单元，第二类是

INPUT 1 OUTPUT1，这类包括了Pre_MAP以及整个单元，第三类则是FMA，也就是INPUT 3 OUTPUT 1。

组件实现

组件实现主要是设计最核心的几个类，也就是transaction类，以及reference model类

1、f_transaction类

> 继承于uvm_sequence_item，主要是作为最基本的数据单元。其成员变量以及主要方法如下：

```
real             f64_value;
bit       [10:0] f64_exp11  ;
rand  bit        sign        ;
bit       [51:0] f64_frac52 ;
rand  bit  [7 :0] f41_exp8   ;
rand  bit  [31:0] f41_frac32 ;
```

其中主要是构建了制定浮点数格式f41的各个数据，以及转换成64位浮点数类型后的比特级数据

```
get_f41_exp8()  [f_transaction]
get_f41_frac32()  [f_transaction]
get_f64_bin()  [f_transaction]
get_sign()  [f_transaction]
get_value()  [f_transaction]
init_transaction()  [f_transaction]
new()  [f_transaction]
post_randomize()  [f_transaction]
```

主要方法如上，主要是进行transaction的初始化，以及获取该类的各个值，同时利用constraint块，可以对产生的浮点数进行更好的受约束的随机化。

2、Reference Model

继承于component。主要是完成数据的对比与转化。将输入的数据转换real类型后，直接利用内置的三角函数、加法、乘法加法运算符得到对应的结果，并将结果再次转化为输入格式的浮点数，交付给scoreboard

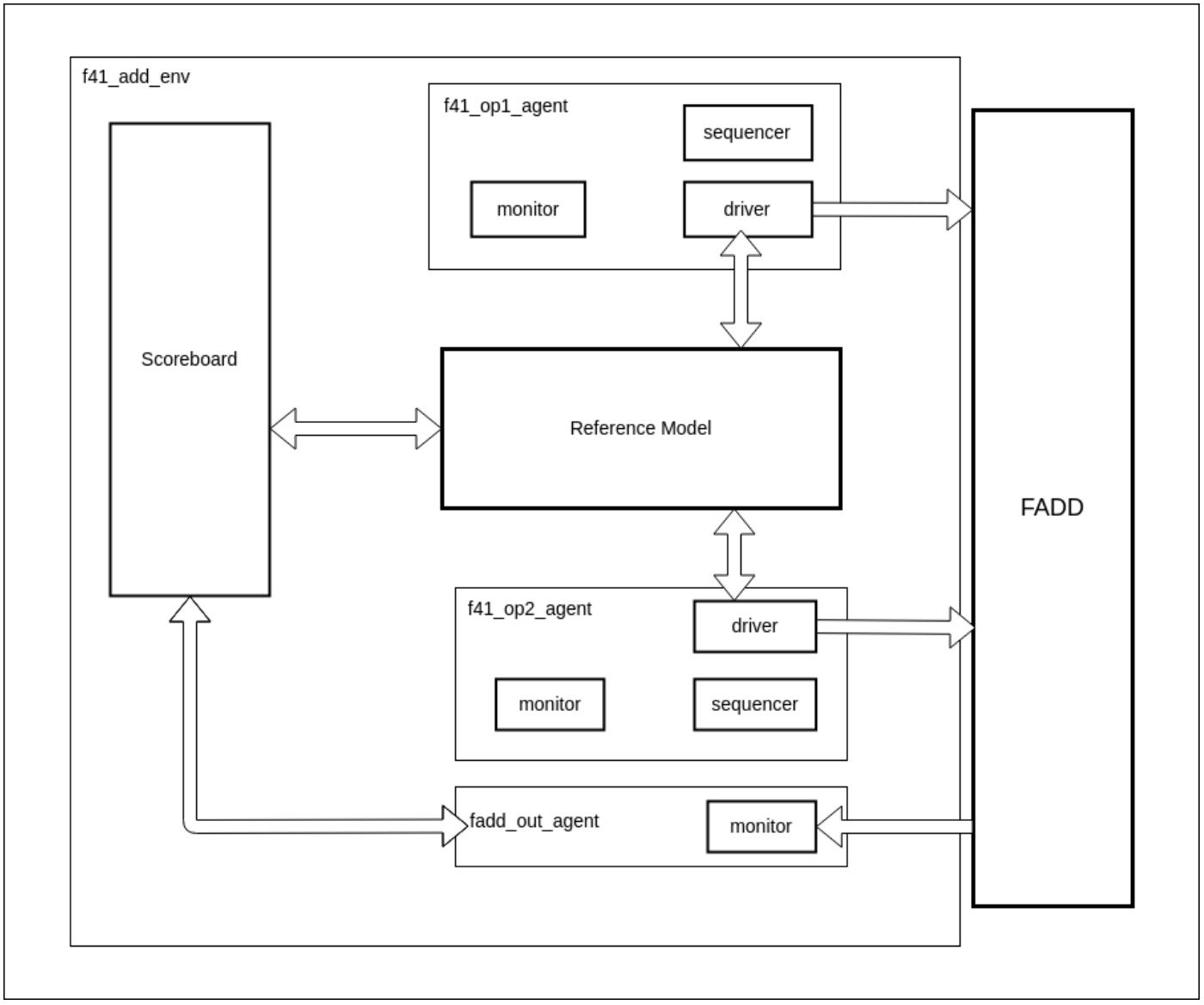
主要方法如下：

```
extern function void build_phase(uvm_phase phase);
extern virtual task main_phase(uvm_phase phase);
extern virtual task ref_add(f_transaction i_tr0,i_tr1,ref_tr);
extern virtual task ref_mul(f_transaction i_tr0,i_tr1,ref_tr);
extern virtual task f64tof41(real float_value,f_transaction tr);
extern virtual task f64toround(bit [10:0] f64_exp,bit [51:0] f64_frac);
```

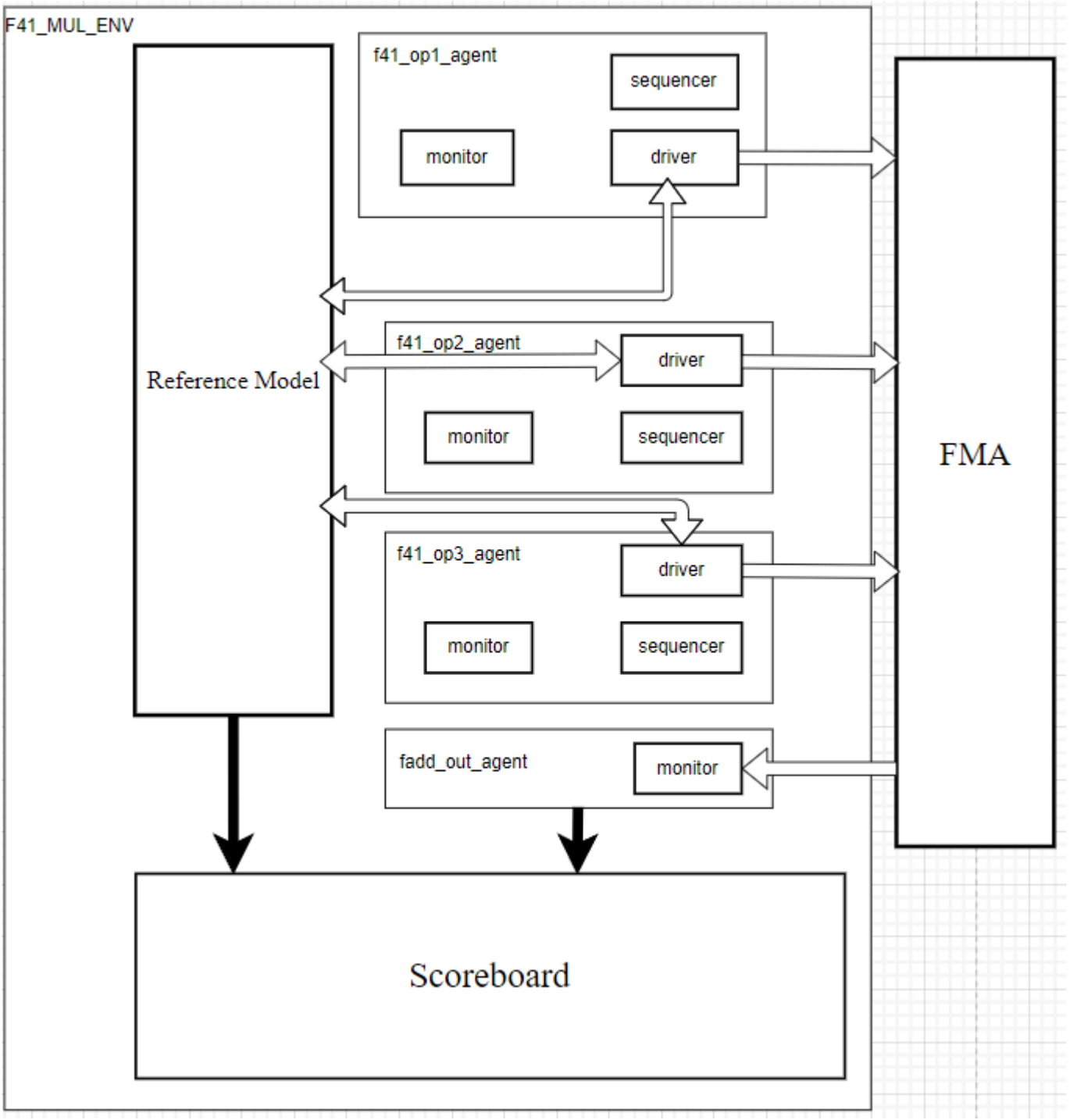
如图设计了，标准加、标准乘的方法进行生成标准数据。

各模块验证环境

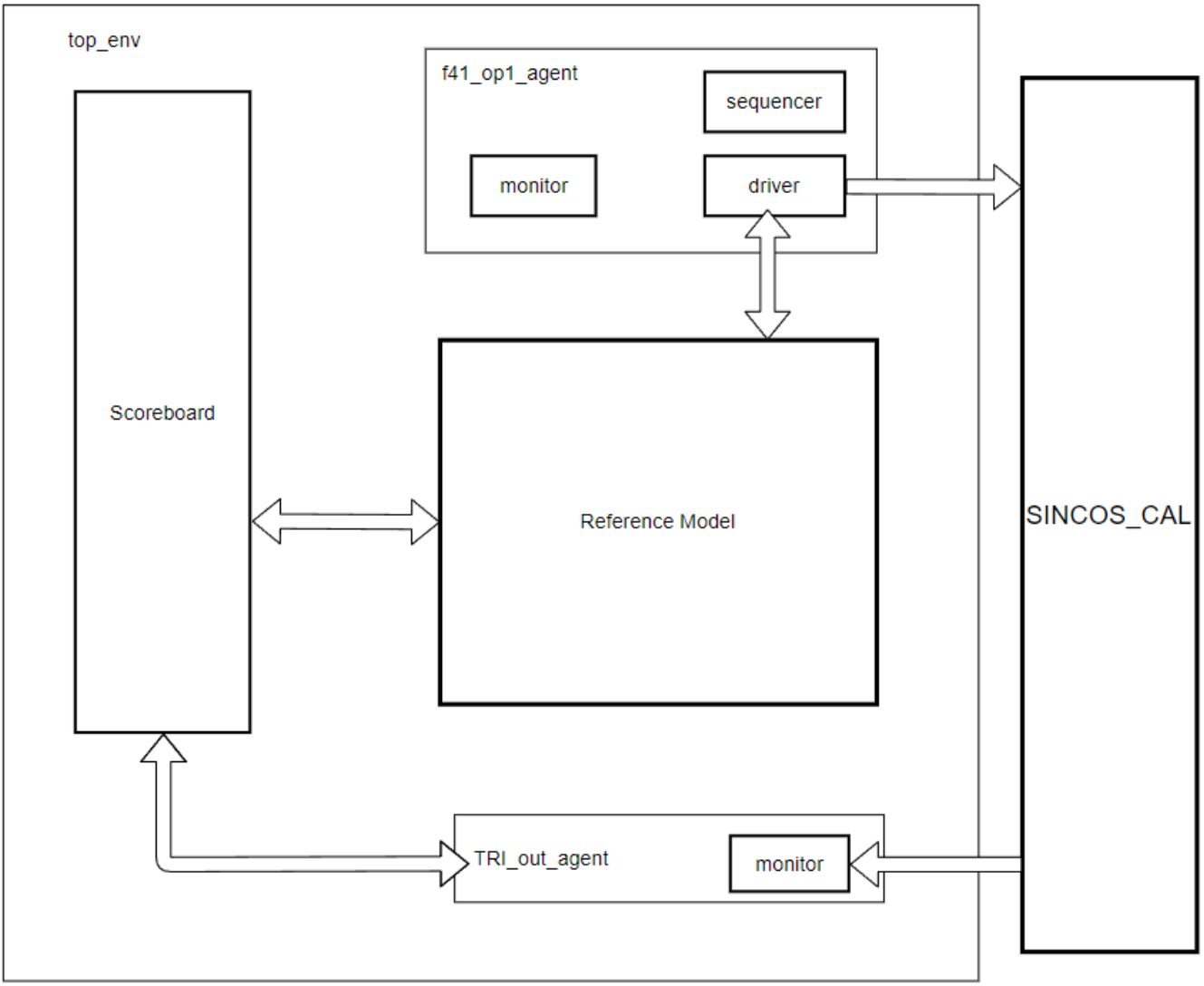
1.FADD_ENV



2.FMA_ENV


F41_MUL_ENV
f41_op1_agent
sequencer
monitor
driver
Reference Model
f41_op2_agent
driver
monitor
sequencer
FMA
f41_op3_agent
driver
monitor
sequencer
fadd_out_agent
monitor
Scoreboard

3.TOP_ENV


top_env
f41_op1_agent
sequencer
monitor
driver
Scoreboard
Reference Model
SINCOS_CAL
TRI_out_agent
monitor