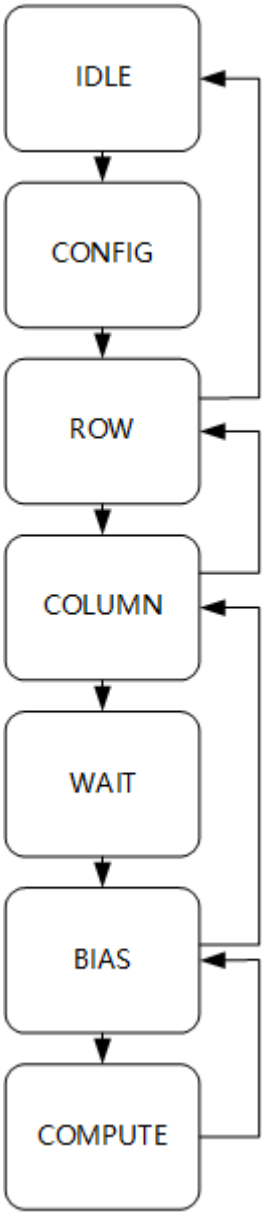
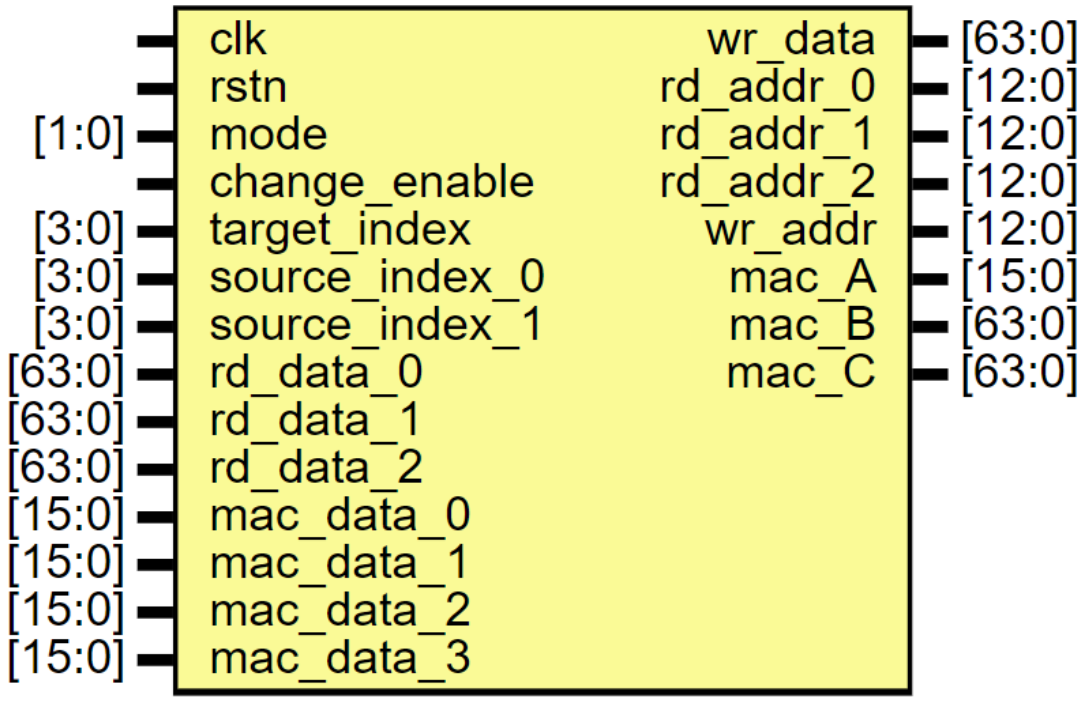
一些基本说明



AGU的基本状态机转移如图所示(可能需要增加中间状态)

基本接口如下

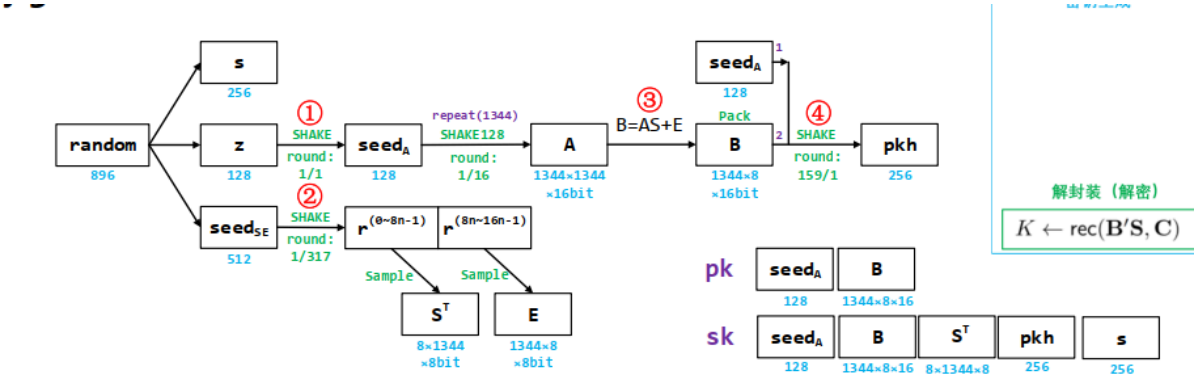


存储采样矩阵和decode,encode时会复用这些接口

内部主要是4层循环,compute循环(矩阵乘法右矩阵的列),bias和column(合起来一起是左矩阵的列,bias是取出来数据的偏置(比如取了64位数据,一次只需要8位,那bias就需要8次循环)),row是左矩阵的行.

4层循环也分别对应了4个计数器.

Key generation




1. ```
   absorbload z(地址) 数据长度2(一次吸收64bit) 安全等级 吸收
   // shake模块从z地址load数据,load2个后进行吸收
   squeezestore seedA(索引) 不转置 更新AGU配置 8(一次输出16bit) 安全等级 挤压 不采样
   // shake模块开始挤压,地址始终由AGU模块提供(注意,由于AGU的地址并非一直有效,因此这里的输出应该需要使能).AGU为mode2(非转置存储). 无compute循环,bias循环4(一次16bit,4个周期满64bit),column循环2(128bit 2次输出完)
   ```

2.
   ```

```
absorbload seedSE(地址) 数据长度8 安全等级 吸收
// shake模块从seedSE的地址load数据,load8个后进行吸收
squeeze S(索引) 转置 更新AGU 159(轮数) 安全等级 采样
//shake模块一次输出16bit(要经过采样),且地址由AGU提供,地址无效时不输出,需要一轮输出完后自动进行下一轮输出!!! AGU为mode1(转置存储) compute循环1344(B的行数)(这里是不是可能需要进入wait状态等待shake模块?),bias循环8(控制每次读出ram数据后,新来的8位写在64位的哪一位)
squeezestore S(索引) 转置 不更新AGU(AGU地址将继续自己跑,不更新) 数据长度(这一轮剩余的长度) 安全等级 采样
// 这个是S与E共用的一次挤压,AGU独立继续产生地址不受影响
squeeze E 不转置 更新AGU 159(轮数) 安全等级 采样
squeezestore E 不转置 不更新AGU 剩余数据长度 安全等级 采样
```

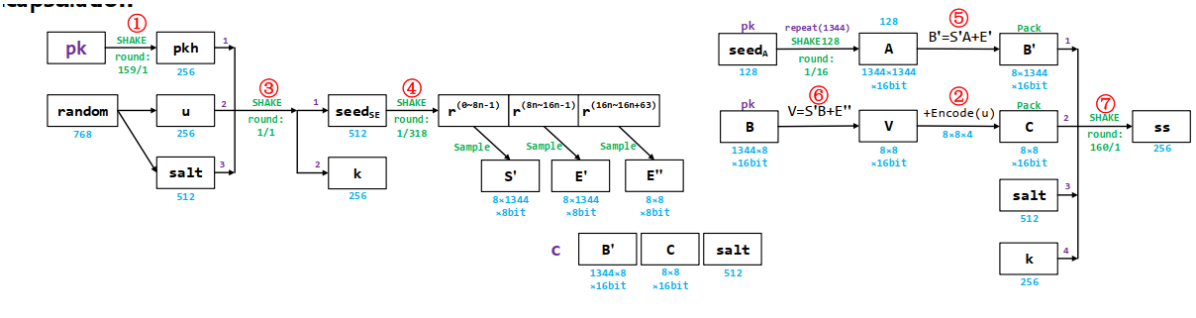
3.

```
matmul E A S
//矩阵乘法,这里的问题是,乘加器的多周期,让compute循环不是1周期加一个,也许需要一个分频?矩阵乘法shake模块一次输出64bit,选择16bit由AGU控制,但这里应该需要shake模块几个周期输出一次
```

4.

```
absorbload seedA(地址) 数据长度2 安全等级 不吸收
// 加载seedA进去不吸收
absorbload B(地址) 首轮B的数据长度 安全等级 吸收
//把第一轮要拼起来的B加载进去,并吸收
absorb B(地址,这里考虑算出这里开始的地址) 157 安全等级
//每次吸收满后,自动吸收,没有额外的控制(不过可以考虑control模块里控制)
absorbload B(地址) 剩余数据长度 安全等级 吸收
// 最后一轮B的数据
squeezestore pkh(索引) 不转置 更新AGU 数据长度256/16 安全等级 不采样
```

Encapsulation



1. 过

2. 图中2,3改顺序,因为encode完u就没有了,不能先encode,多步吸收,过

3.

```
encode u(索引)
// encode和decode其实没太细想,AGU里本身有个64位的buffer,循环控制读取数据,然后数据读进来进行相应的操作再写应该就行了
```

4. 过

5.

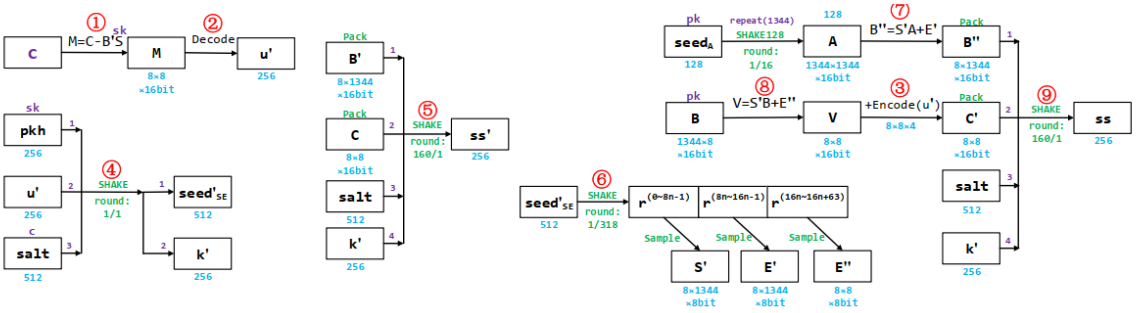
```
matmul E' S' A
//这里A右乘,shake一样需要输出64bit,但是间隔会比左乘短(本身乘法需要多周期,因此对shake模块而言可能其实没区别,只是间隔不同的问题?)
```

6.

```
matmul E'' S' B
//三个矩阵都来自ram,我有点担心这种情况的时序和有A的不一样,可能还是得单开,但是再说吧
matadd V u
```

7. 过

Decapsulation



1.

```
matmul C -B' S
//多个负号
```

2.

```
decode M(索引)
```

3. 相同问题,3,4换顺序 过

4.

```
encode u'(索引)
```

写到这感觉encode和decode感觉可以考虑直接指令输入地址,查表有点太多了...

5. 过

6. 过

7. 过

8. 过