密码算法IP平台

SPEC

修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修改时间 | 修改者 | 描述 |
| V0.1 | 2014-03-25 | 郭岩松、胡建兵、常乐、杨锦江 | 以与北信院合作的SPEC为基础创建文档 |
| V0.1.1 | 2014-03-25 | 常乐、胡建兵 | 按照某所的技术思路补充DATAPATH部分 |
| V0.2 | 2014-03-26 | 杨锦江 | 1. 修改总体描述的框图 2. 增加可重构计算阵列的描述 3. 删除通用寄存堆缓存以及配置反馈的章节 |
| V0.3 | 2014-03-27 | 常乐、胡建兵、郭岩松、杨锦江 | 常乐：细化可重构计算单元以及配置码的定义  郭岩松：细化互联网络描述  胡建兵：补充ALU内部结构图，更新LUT结构图  杨锦江：加入阵列计算控制单元 |
| V0.4 | 2014-03-28 |  | 胡建兵：补充ALU共描述  常乐：更新配置码定义、行模块的框图 |
| V0.5 | 2014-04-02 | 杨锦江、郭岩松、胡建兵 | 杨锦江：补充计算阵列的行读写配置及整体控制配置  郭岩松：加入Benes网络的端口定义  胡建兵：替换ALU内部结构图，增加ALU端口定义、更新配置信息定义表格、更新配置包格式、修改配置存储器数据摆放格式 |
| V0.6 | 2014-04-02 | 常乐 | 常乐：   * 更新了5副图以及总体描述，LUT描述； * 添加了总体端口定义，LUT端口定义； * 更新了配置信息定义（除了CONTROL部分）   胡建兵：   * 修改配置字中阵列控制部分 * 修改了配置包的解析的文字说明 * 初步修改了第三章的端口定义   杨锦江：  增加了Block控制单元的端口列表 |
| V0.6.1 | 2014-05-09 | 胡建兵、常乐 | 胡建兵：更新了配置存储器存储组织图  常乐：更新功能描述部分内容 |
| V0.7 | 2014-11-28 | 胡建兵 | 根据课题修改SPEC以适应课题，主要为精简计算阵列，简化为4行，1个Block。 |
| V0.7.1 | 2014-12-02 | 胡建兵 | 精简alu的opcode |
| V0.7.2 | 2014-12-04 | 胡建兵 | 1、修改Load和Store模块的端口定义  2、统一输入和输出从0开始编号 |
| V0.8 | 2015-3-17 | 李兆奇、申艾麟、尹玲 | 1、根据精简的架构修改2.1.6和2.2部分  2、修改2.1.3和2.3部分  3、修改2.1中除了2.1.3和2.1.6的其他部分 |
| V1.0 | 2015-3-18 | 胡建兵 | 修改配置部分中配置包格式；  整理文档，图和表的编号等 |
| V1.1 | 2015-4-1 | 胡建兵 | 确定配置带宽为128bit，修改与之相关的部分；  修改配置信息定义中CTR部分 |
| V1.2 | 2015-4-9 | 胡建兵 | 修改3.2部分的配置解析表格，增加了RCline开始行数一栏 |
| V1.3 | 2015-4-22 | 胡建兵 | 修改行控制配置信息格式，增加了Store2的输出选择；  修改RC的配置信息中关于out1直通的描述 |

目录

[名词定义： 6](#_Toc414451718)

[1、 总体描述 7](#_Toc414451719)

[2、 计算引擎（datapath） 7](#_Toc414451720)

[2.1 可重构阵列 8](#_Toc414451721)

[2.1.1 计算单元（ALU） 10](#_Toc414451722)

[2.1.2互联单元（CU） 12](#_Toc414451723)

[2.1.3查找表单元（LUT） 14](#_Toc414451724)

[2.1.4 Block控制单元（Block Control Unit，BCU） 16](#_Toc414451725)

[2.1.5 Load和Store单元 17](#_Toc414451726)

[2.1.6配置信息定义 18](#_Toc414451727)

[2.2 通用寄存器堆 22](#_Toc414451728)

[2.2.1通用寄存器堆（General Purpose Register File） 22](#_Toc414451729)

[**2.3 计算引擎数据接口** 23](#_Toc414451730)

[2.3.1 输入FIFO（INFIFO） 24](#_Toc414451731)

[2.3.2 输出FIFO（OUTFIFO） 25](#_Toc414451732)

[2.4计算配置接口（Computing Configuration Interface） 25](#_Toc414451733)

[2.4.1 功能描述 25](#_Toc414451734)

[2.4.2 接口定义 25](#_Toc414451735)

[3、 配置控制模块 27](#_Toc414451736)

[3.1配置信息接口（Configuration Interface）（未实现） 27](#_Toc414451737)

[3.1.1 功能描述 27](#_Toc414451738)

[3.1.2接口定义 27](#_Toc414451739)

[3.2 配置包解析（Configuration Packet Parser） 28](#_Toc414451740)

[3.2.1功能描述 28](#_Toc414451741)

[3.2.2接口定义 30](#_Toc414451742)

[3.3 配置存储器 31](#_Toc414451743)

[3.3.1功能描述 31](#_Toc414451744)

[3.3.2接口定义 32](#_Toc414451745)

[3.4 配置包存储器 33](#_Toc414451746)

[3.4.1功能描述 33](#_Toc414451747)

[3.4.2接口定义 33](#_Toc414451748)

[4、 算法映射与评估 33](#_Toc414451749)

# 名词定义：

**配置包**

包含计算引擎的配置信息索引，在运行时由配置包解析模块动态解析

**配置包存储器**

存放某个算法所有配置包的存储器，位于配置包解析模块中

**配置存储器**

存放计算引擎的所有配置信息，包括可重构阵列的配置信息、功能单元的配置信息；初始化时由片外加载

**配置寄存器**

计算引擎中配置信息的存储单元，执行过程中由配置包解析单元根据配置包的内容对其动态更新

**计算引擎**

特种应用处理器中的主要计算单元，由可重构阵列、功能单元、配置反馈模块以及其相关的数据配置接口组成

**功能模块**

计算引擎中的异构模块，用于完成某些特定的功能，具体功能保密

**通用寄存器堆**

计算引擎中的存储器，用于缓存计算引擎的中间结果

# 总体描述

面向分组密码算法可重构架构是一款面向分组密码应用的IP。该IP采用可重构技术设计，可在一块硬件上通过配置的切换来支持多种分组密码算法。IP设计运行频率MHz，制造工艺采用TSMC 45nm。其功能模块划分如图1.1图所示：



图1.1：特种应用可重构处理器总体框图

工作机制如下：

1. 系统上电,配置信息由片外加载到片上配置存储器中;
2. 执行某算法前，将此算法所有的配置包写入到配置包存储器中（配置包存储器包含在配置解析单元中）;
3. 配置解析单元解析配置索引,从配置存储器中选择相应的配置对可重构阵列及功能模块进行配置；
4. 阵列从INFIFO中读取数据进行计算，计算结果写入到OUTFIFO中；
5. 可重构阵列与功能模块计算的中间结果数据只与通用寄存器堆进行交互；
6. 阵列计算的中间结果通过通用寄存器堆缓存；

# 计算引擎（datapath）

计算引擎为可重构处理器中主要运算模块，其主要模块包括可重构阵列、功能单元、通用寄存器堆、配置接口和数据接口，结构框图如图1.1所示。计算引擎的一个典型的计算过程描述如下：

Step1：接受配置：本次计算的配置信息从配置接口传输到计算引擎内部，对计算引擎进行配置；

Step2：计算阵列或功能模块从INFIFO读取数据进行运算；

Step3：阵列计算完成后，运算结果写回到OUTFIFO；

注：计算的中间结果可以利用通用寄存器堆做缓存

## 2.1 可重构阵列

#### 功能描述

可重构阵列的基本单元是行，每行由16个8位ALU组成。每一行的输出和下一行的输入模块直接相连。第一行可以从FIFO或者GPRF（通用寄存器堆）读入数据，最后一行可以将数据读出到FIFO或者GPRF中。除首行外，每行均可选择由上一行输出作为输入；除尾行外，每行输出均可作为下一行输入。

可重构阵列为40行，每四行共享一个查找表并且由一组控制信息控制。行结构如下图2.1所示：



图2.1：可重构阵列行结构图

每一行阵列有两个输入：输入0与输入1。通过MUX选择，分别输入4个Benes网络（2个128bit置换，2个16Byte置换）。Benes网络的置换结果输出至对应ALU的IN0-IN2端口。ALU的输入端口通过4位多路选择器选择一个Benes网络的置换结果作为输入，并接受外界控制信号对数据进行运算，运算结果通过OUT\_0、OUT\_1端口输出。在与LUT查表信息进行选择后，成为本行的输出0、输出1。

#### 端口定义

表2.1 行单元定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signals | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RST\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| IFIFO\_RD\_N\_OUT \_0 | 1 | FIFO | 输入端口0读FIFO使能信号，低电平有效 |
| IFIFO\_RD\_DATA\_IN\_0 | 128 | FIFO | 输入端口0读FIFO数据信号 |
| IFIFO\_RD\_N\_OUT \_1 | 1 | FIFO | 输入端口1读FIFO使能信号，低电平有效 |
| IFIFO\_RD\_DATA\_IN\_1 | 128 | FIFO | 输入端口1读FIFO数据信号 |
| GPRF\_RD\_N\_OUT \_0 | 1 | GPRF | 输入端口0读GPRF使能信号，低电平有效 |
| GPRF\_RD\_ADR\_OUT \_0 | 5 | GPRF | 输入端口0读GPRF地址信号 |
| GPRF\_RD\_DATA\_IN\_0 | 128 | GPRF | 输入端口0读GPRF数据信号 |
| GPRF\_RD\_N\_OUT \_1 | 1 | GPRF | 输入端口1读GPRF使能信号，低电平有效 |
| GPRF\_RD\_ADR\_OUT \_1 | 5 | GPRF | 输入端口1读GPRF地址信号 |
| GPRF\_RD\_DATA\_IN\_1 | 128 | GPRF | 输入端口1读GPRF数据信号 |
| LINE\_RD\_DATA\_IN\_0 | 128 | 上一行 | 输入端口0读上一行数据信号 |
| LINE\_RD\_DATA\_IN\_1 | 128 | 上一行 | 输入端口1读上一行数据信号 |
| LUT\_RD\_DATA\_IN | 128 | LUT | LUT查表输出数据信号 |
| LOAD0\_TYPE\_IN | 2 |  | Load模块输入来源选择 |
| LOAD0\_ADDR\_IN | 3 |  | Load模块输入地址 |
| LOAD1\_TYPE\_IN | 2 |  | Load模块输入来源选择 |
| LOAD1\_ADDR\_IN | 3 |  | Load模块输入地址 |
| BENES\_SEL\_IN | 4 |  | Benes网络的数据来源选择 |
| BENES32\_0\_CONFIG\_IN | 56 |  | Benes32网络的配置（Benes0） |
| BENES32\_1\_CONFIG\_IN | 56 |  | Benes32网络的配置（Benes3） |
| BENES256\_0\_CONFIG\_IN | 832 |  | Benes256网络的配置（Benes1） |
| BENES256\_1\_CONFIG\_IN | 832 |  | Benes256网络的配置（Benes2） |
| RC\_LINE\_CONFIG\_IN | 1024 |  | RC行配置 |
| STORE0\_TYPE\_IN | 2 |  | Store模块输出目标选择 |
| STORE0\_ADDR\_IN | 3 |  | Store模块输出地址 |
| STORE0\_ADDR\_MASK\_IN | 8 |  | Store输出到gprf的mask信号 |
| STORE0\_TYPE\_IN | 2 |  | Store模块输出目标选择 |
| STORE0\_ADDR\_IN | 3 |  | Store模块输出地址 |
| STORE0\_ADDR\_MASK\_IN | 8 |  | Store输出到gprf的mask信号 |
| Write Port | | | |
| OFIFO\_WR\_DATA\_OUT\_0 | 128 | FIFO | 输出端口0写FIFO数据信号 |
| OFIFO\_WR\_N\_OUT \_0 | 1 | FIFO | 输出端口0写FIFO使能信号，低电平有效 |
| OFIFO\_WR\_DATA\_OUT\_1 | 128 | FIFO | 输出端口1写FIFO数据信号 |
| OFIFO\_WR\_N\_OUT \_1 | 1 | FIFO | 输出端口1写FIFO使能信号，低电平有效 |
| GPRF\_WR\_N\_OUT \_0 | 1 | GPRF | 输出端口0写GPRF使能信号，低电平有效 |
| GPRF\_WR\_ADR\_OUT \_0 | 5 | GPRF | 输出端口0写GPRF地址信号 |
| GPRF\_WR\_MASK\_OUT \_0 | 8 | GPRF | 输出端口0写GPRF掩码信号 |
| GPRF\_WR\_DATA\_OUT \_0 | 128 | GPRF | 输出端口0写GPRF数据信号 |
| GPRF\_WR\_N\_OUT \_1 | 1 | GPRF | 输出端口1写GPRF使能信号，低电平有效 |
| GPRF\_WR\_ADR\_OUT \_1 | 5 | GPRF | 输出端口1写GPRF地址信号 |
| GPRF\_WR\_MASK\_OUT\_1 | 8 | GPRF | 输出端口1写GPRF掩码信号 |
| GPRF\_WR\_DATA\_OUT\_1 | 128 | GPRF | 输出端口1写GPRF数据信号 |
| LUT\_WR\_DATA\_IN | 128 | LUT | 输出作LUT查表输入数据信号 |
| LINE\_WR\_DATA\_OUT\_0 | 128 | 下一行 | 输出端口0作下一行数据信号 |
| LINE\_WR\_DATA\_OUT\_1 | 128 | 下一行 | 输出端口1作下一行数据信号 |

### 2.1.1 计算单元（ALU）

#### 2.1.1.1 功能描述

可重构阵列的每行有16个ALU，ALU的结构图如图2.2。



图2.2：ALU结构图

下图2.3是ALU内部结构图，内部包含算术逻辑单元（ALU），输入寄存器（reg），输出选择（MUX），通过配置信息实现对ALU配置，实现相应的功能。



图2.3：ALU内部结构图

图中的ALU为8位的，包含三个输入寄存器，分别对应于三个8位的输入数据，ALU接受Config\_in的配置，实现计算，输出的数据为8位的Out\_0，Out\_1的结果可以是In3的旁路输出或者和Out\_0相同，同样需要配置。与此同时，ALU可以通过Cin和Cout的级联，4个ALU级联可以实现32位加法器。

#### 2.1.1.2 端口定义

表2.2 ALU定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signals | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RST\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| BENES0\_DATA\_IN | 8 |  | BENES0输入数据 |
| BENES1\_DATA\_IN | 8 |  | BENES1输入数据 |
| BENES2\_DATA\_IN | 8 |  | BENES2输入数据 |
| BENES3\_DATA\_IN | 8 |  | BENES3输入数据 |
| ALU\_CONFIG\_IN | 15 |  | 输入配置信息 |
| ALU\_C\_IN | 1 | ALU | 输入进位 |
| Write Port | | | |
| ALU\_WR\_DATA\_OUT\_0 | 8 |  | 输出计算结果0 |
| ALU\_WR\_DATA\_OUT\_1 | 8 |  | 输出计算结果1 |
| ALU\_C\_OUT | 1 | ALU | 输出进位 |

### 2.1.2互联单元（CU）

#### 2.1.2.1 功能描述

每行的互联结构的框图如图2.4



图2.4：行间互连



图2.5：128×128 BENES网络

* 行输入MUX

各行输入两个128比特数据：IN0，IN1，输出两个128比特数据：OUT0，OUT1。IN0和IN1分别连接到四个128位二选一的行输入MUX：IMUX0，IMUX1，IMUX2，IMUX3。

* BENES网络

在行输入MUX的输出和RC的输入之间用BENES网络实现任意无重复置换模式，每个行输入MUX对应一个BENES网络，三个组合的256位RC操作数可以通过RC输入MUX选取来自任何一个BENES网络的输出。有两种类型的BENES网络：BENES16和BENES128。BENES16实现一个128比特数据中的连续16个字节的置换，BENES128实现一个128比特数据中的128个比特的置换。图2.5示出了BENES128的结构。在一行中的四个BENES的类型从左至右依次为：BENES16，BENES128，BENES128，BENES16。

通过行输入MUX和BENES网络的组合，各个RC操作数可以来自两个行输入IN0和IN1中的任何一个，并且可以在输入RC时完成字节级和比特级的置换。

N×N的BENES需要Nlog2N-N/2个2×2开关，BENES16需要56位配置码，BENES128需要832位配置码。

#### 2.1.2.2 端口定义

表2.3 Benes1接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| DATA\_IN | 128 | REG/FIFO/LINE | 128比特输入数据 |
| DATA\_OUT | 128 | ALU | 128比特输出数据 |
| CONFIG\_IN | 56 | CPP | 56比特配置码的写入口 |

表2.X Benes2接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| DATA\_IN | 128 | REG/FIFO/LINE | 128比特输入数据 |
| DATA\_OUT | 128 | ALU | 128比特输出数据 |
| CONFIG\_IN | 832 | CPP | 832比特配置码的写入口 |

表2.X Benes3接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| DATA\_IN | 128 | REG/FIFO/LINE | 128比特输入数据 |
| DATA\_OUT | 128 | ALU | 128比特输出数据 |
| CONFIG\_IN | 832 | CPP | 832比特配置码的写入口 |

表2.X Benes4接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| DATA\_IN | 128 | REG/FIFO/LINE | 128比特输入数据 |
| DATA\_OUT | 128 | ALU | 128比特输出数据 |
| CONFIG\_IN | 56 | CPP | 56比特配置码的写入口 |

### 2.1.3查找表单元（LUT）

#### 2.1.3.1 功能描述

查找表（LUT）在Block中的互联结构图如图2.6，在每个Block的RC层间，输出的结果可以通过MUX到LUT中进行查表操作，查找到的数据返回到相应的下一层的输出。(下图只以层间直连的形式说明LUT互联结构)

图2.6：LUT的互连结构

由图可见，RC的计算和LUT的查表操作不可以同时进行，LUT的通过data\_sel实现配置。每一个group(4行)中，可以选择line0，line1，line2的输出作为LUT的查表输出，并将输出作为line1，line2，line3的输出。

#### 2.1.3.2 端口定义

表2.4 LUT接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signals | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RST\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| LUT\_RD\_DATA\_IN\_0 | 128 | RCline0 | Line0的输出进行查表 |
| LUT\_RD\_DATA\_IN\_1 | 128 | RCline1 | Line1的输出进行查表 |
| LUT\_RD\_DATA\_IN\_2 | 128 | RCline2 | Line2的输出进行查表 |
| LUT\_RD\_DATA\_IN\_3 | 128 | RCline3 | Line3的输出进行查表 |
| LUT\_CONFIG | 2048 |  |  |
| LUT\_SEL\_IN | 3 | BCU | LUT配置信息 |
| Write Port | | | |
| LUT\_WR\_DATA\_OUT\_0 | 128 | RCline1 | 查表结果作为Line1输出 |
| LUT\_WR\_DATA\_OUT\_1 | 128 | RCline2 | 查表结果作为Line2输出 |
| LUT\_WR\_DATA\_OUT\_2 | 128 | RCline3 | 查表结果作为Line3输出 |
| LUT\_WR\_DATA\_OUT\_3 | 128 | 下一行RCline0 | 查表结果作为下一行Line0输出 |

### 2.1.4 RCline控制单元（RCline Control Unit，RCU）

#### 2.1.4.1功能描述

RCline的计算控制单元负责RCline配置信息的解析，并根据时序配置信息（详见配置信息定义中的Control部分）控制整个RCline执行时的时序。配置控制器与RCline间的握手机制也由该模块实现。

#### 2.1.4.2端口定义

表2.5 BCCU接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signals | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RST\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
|  | | | |
| RCLINE\_EN\_IN | 1 | CPP | RCline的使能 |
| RCLINE\_READY\_OUT | 1 | RCline运算完成标志 |
| CTR\_CONFIG\_IN | 128 | CPP写入的配置信息接口 |
| INFIFO\_EMPTY\_N\_IN | 1 | INFIFO | INFIFO的空信号 |
| OUTFIFO\_FULL\_N\_IN | 1 | OUTFIFO | OUTFIFO的满信号 |
| LOAD0\_EN\_N \_OUT | 1 | LOAD0 | LOAD的使能信号，低电平有效 |
| LOAD1\_EN\_N \_OUT | 1 | LOAD1 | LOAD的使能信号，低电平有效 |
| STORE0\_EN\_N\_OUT | 1 | STORE0 | STORE的使能信号，低电平有效 |
| STORE1\_EN\_N\_OUT | 1 | STORE1 | STORE的使能信号，低电平有效 |
| LOAD0\_TYPE\_OUT | 2 | LOAD0 | LOAD的数据来源类型  00：上一行；  01：FIFO；  10：GPRF |
| LOAD0\_ADDR\_OUT | 6 | LOAD0 | LOAD读取GPRF的地址 |
| LOAD1\_TYPE\_OUT | 2 | LOAD1 | LOAD的数据类型  00：上一行；  01：FIFO；  10：GPRF |
| LOAD1\_ADDR\_OUT | 6 | LOAD1 | LOAD读取GPRF的地址 |
| STORE0\_TYPE\_OUT | 2 | STORE0 | STORE的数据去向类型  00：上一行；  01：FIFO；  10：GPRF |
| STORE0\_ADDR\_OUT | 6 | STORE0 | STORE写入GPRF的地址 |
| STORE0\_MASK\_OUT | 4 | STORE0 | STORE写入GPRF的mask信号 |
| STORE1\_TYPE\_OUT | 2 | STORE1 | STORE的数据去向类型  00：上一行；  01：FIFO；  10：GPRF |
| STORE1\_ADDR\_OUT | 6 | STORE1 | STORE写入GPRF的地址 |
| STORE1\_MASK\_OUT | 4 | STORE1 | STORE写入GPRF的mask信号 |
| LUT\_SEL\_OUT | 1 | LUT | LUT的选择信号 |
| BENES\_SEL | 8 | BENES | BENES的选择信号 |

### 2.1.5 Load和Store单元

#### 2.1.5.1功能描述

通过Load和Store模块将数据在每层计算阵列行结构之间传递。其中Load模块负责把数据从INFIFO、上一行或者GPRF中读进来，传递给Benes网络处理。Store模块负责把数据从计算阵列行结构的数据输出到OUTFIFO、下一行或者GPRF中去。

#### 2.1.5.2端口定义

表2.6 Load模块接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signals | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RST\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| LOAD\_EN\_N\_IN | 1 | BCU | Load模块的使能信号，低电平有效 |
| LOAD\_TYPE\_IN | 2 | Load模块输入来源选择：  00：从上一行读取数据  01：从FIFO读数据  10：从GPRF读取数据 |
| LOAD\_ADDR\_IN | 3 | Load模块的输入地址 |
| FIFO\_RD\_DATA\_IN | 128 | 读FIFO的数据 |
| GPRF\_RD\_DATA\_IN | 128 | 读GPRF的数据 |
| RCLINE\_DATA\_IN | 128 | RCLINE | 上一行的数据 |
| Write Port | | | |
| FIFO\_RD\_N\_OUT | 1 | FIFO | 读FIFO或者GPRF的使能信号，低电平有效 |
| GPRF\_RD\_N\_OUT | 1 | GPRF | 读FIFO或者GPRF的使能信号，低电平有效 |
| GPRF\_RD\_ADDR\_OUT | 3 | 读FIFO或者GPRF的地址信号 |
| LOAD\_DATA\_OUT | 128 | RCLINE | Load模块的输出数据 |

表2.7 Store模块接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signals | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RST\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| STORE\_EN\_N\_IN | 1 | BCU | Store模块的使能信号，低电平有效 |
| STORE\_TYPE\_IN | 2 | Store模块输入来源选择：  00：从上一行读取数据  01：从FIFO读数据  10：从GPRF读取数据 |
| STORE\_ADDR\_IN | 3 | Store模块的输入地址 |
| STORE\_ADDR\_MASK\_IN | 8 | Store模块写GPRF的mask信号 |
| STORE\_DATA\_IN | 128 | RCLINE | Store模块输入数据 |
| Write Port | | | |
| FIFO\_WR\_N\_OUT | 1 | OUTFIFO | 写FIFO的使能信号，低电平有效 |
| FIFO\_WR\_DATA\_OUT | 128 | 写FIFO的数据 |
| GPRF\_WR\_N\_OUT | 1 | GPRF | 写GPRF的使能信号，低电平有效 |
| GPRF\_WR\_ADDR\_OUT | 3 | 写GPRF的地址信号 |
| GPRF\_WR\_MASK\_OUT | 8 | 写GPRF的mask的信号 |
| GPRF\_WR\_DATA\_OUT | 128 | 写GPRF的数据 |
| RCLINE\_DATA\_OUT | 128 | RCLINE | 写到下一行数据 |

### 2.1.6配置信息定义

表2.8 配置信息定义

|  |  |  |  |
| --- | --- | --- | --- |
| 对象 | 配置位置 | 位数 | 含义 |
| RC Row#0 | Word0 | [1:0] | Rin\_mux0控制位，RC第一个输入端MUX选择信号：  00：Benes0（16字节替换）  01：Benes1（128位替换）  10：Benes2（128位替换）  11：Benes3（16字节替换） |
| [3:2] | Rin\_mux1控制位 |
| [5:4] | Rin\_mux2控制位 |
| [6] | bypass\_en：  0: RC的OUT1端口配置为OUT0端口相同输出  1: RC的OUT1端口配置为bypass态，输出IN3的数据  （如果需要用到OUT1的端口作为进位输出，配置为0时，输出为进位制不是OUT0的值） |
| [12:7] | opcode\_in操作码（见附表1） |
| [13] | Ro\_mux0控制信号OMUX\_config0  RC第一个输出端MUX的选择信号： 0 RC本身 1 查找表 |
| [14] | Ro\_mux1控制信号OMUX\_config1 |
| [15] | 保留 |
| Word0[31:16] |  | …… |
| Word1[15:0] |  | …… |
| …… |  | …… |
| Word7[31:16] |  | …… |
| Benes Row#0 | Word0 |  | Benes0配置码 |
| Word1 | [23:0] |
| [31:24] | 保留 |
| Word2 |  | Benes3配置码 |
| Word3 | [23:0] |
| [31:24] | 保留 |
| Word29~Word4 |  | Benes1配置码 |
| Word55~Word30 |  | Benes2配置码 |
| Word56 | [0] | Benes0输入来源选择信号：  0 来自输入0(load  0或者out0)；1 来自输入1（load1或者out1） |
| [1] | Benes1输入来源选择信号 |
| [2] | Benes2输入来源选择信号 |
| [3] | Benes3输入来源选择信号 |
| [31:4] | 保留 |
| Control Row#0 | Word0 | [0] | LUT的输入选择：  0：不进行查表  1：上一行输出作为查表的输入 |
| [2:1] | 保留 |
| [9:3] | 当前数据流图循环的次数，最大128次 |
| [11:10] | 每行之间第三个通道数据来源：  00：benes0；  01：benes1；  10：benes2；  11：benes3； |
| [15:12] | 保留 |
| [17:16] | 第0行读端口0的输入选择  00：端口0不从外部读取数据  01：端口0从FIFO读数据  10：端口0从GPRF读取数据  11：保留 |
| [19:18] | 第0行读端口1的输入选择 |
| [21:20] | 第0行写端口0的输出选择  00：端口0的数据不向外部流出  01：端口0的数据写入FIFO  10：端口0的数据写入GPRF  11：保留 |
| [23:22] | 第0行写端口1的输出选择 |
| [25:24] | Benes16\_0数据来源选择  00：load0  01：load1  10：load2（为新增的直通项） |
| [27:26] | Benes128\_0数据来源选择 |
| [29:28] | Benes128\_1数据来源选择 |
| [31:30] | Benes16\_1数据来源选择 |
| Word1 | [5:0] | 第0行读延时：表示该Block开始工作后，某行从FIFO或GPRF读取数据的时间延时 |
| [12:6] | 第0行读数据个数 |
| [17:13] | 第0行读地址个数 |
| [23:18] | 第0行端口0读地址 |
| [29:24] | 第0行端口1读地址 |
| [31:30] | 保留 |
| Word2 | [5:0] | 第0行写延时：表示该Block开始工作后，某行从FIFO或GPRF读取数据的时间延时 |
| [12:6] | 第0行写数据个数 |
| [17:13] | 第0行写地址个数 |
| [31:18] | 保留 |
| Word3 | [5:0] | 第0行端口0写地址 |
| [9:6] | 第0行端口0写地址掩码 |
| [15:10] | 第0行端口1写地址 |
| [19:16] | 第0行端口1写地址掩码 |
| [31:20] | 保留 |
| RC Row#1 |  |  |  |
| Benes Row#1 |  |  |  |
| Control Row#1 |  |  |  |
| …… |  |  |  |
| RC Row#3 |  |  |  |
| Benes Row#3 |  |  |  |
| Control Row#3 |  |  |  |

表2.9 ALU中opcode的定义

|  |  |  |  |
| --- | --- | --- | --- |
| 信号 | 类型 | 值 | 功能 |
| Opcode\_in[5:0] | 二操作数布尔计算指令集 | 000001 | 2输入XOR |
| 000010 | 2输入OR |
| 000011 | 2输入AND |
| 输出全0 | 000000 | O0输出8位0 |
| 三操作布尔计算指令集 | 100000 | A&B&C |
| 100001 | A&B|C |
| 100010 | A&B^C |
| 100011 | A|B&C |
| 100100 | A|B|C |
| 100101 | A|B^C |
| 100110 | A^B&C |
| 100111 | A^B|C |
| 101000 | A^B^C |
|  | 110000 | X2(A^B) |
| 算数指令集 | 000101 | 2输入8位无符号全加器（进位位于O1） |
| 000110 | 2输入32位无符号全加器的部件（4个RC组成32位全加器，最高位进位位于第4个RC的O1） |
| 001000 | 3输入32位无符号全加器的部件（4个RC组成32位全加器，最高位进位位于第4个RC的O2） |
| 受控选择 | 110011 | 通过IN3选择O0的输出，表达式为（A&C）|B&（~C） |
| 输出旁路 | 110111 | O0输出IN0，O1输出IN2 |

## 2.2 通用寄存器堆

通用寄存器堆模块用于可重构阵列内部各Block之间，从而起到类似于一级缓存的作用。

### 2.2.1通用寄存器堆（General Purpose Register File）

通用寄存器堆（General Purpose Register File）的容量为32\*128比特。其分别有10个独立的128比特读端口与10个独立的128比特写端口这些端口可以实现对寄存器堆中10个128比特数据的读写操作。特别的其写端口都支持以字为单位的mask操作，即每个128比特写端口都由4位的mask信号控制。mask信号低位定义为有效，例如，当某个写端口的mask信号为1101时，写操作仅将数据写入对应128比特寄存器的第3个字，而其余3字的寄存器值保持不变。

通用寄存器堆的数据由可重构阵列写入，由可重构阵列读出。特别的。

表2.10 GPRF接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signal | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效。 |
| RESET\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| RCLINE0\_RD\_N\_IN | 1 | 可重构阵列  BLOCK0 | BLOCK0对应的读端口的读使能，低电平有效 |
| RCLINE0\_RD\_ADDR\_IN | 5 | BLOCK0对应的读端口的读地址 |
| RCLINE0\_RD\_DATA\_OUT | 128 | BLOCK0对应的读端口的读数据 |
| RCLINE1\_RD\_N\_IN | 1 | 可重构阵列  BLOCK1 | BLOCK1对应的读端口的读使能，低电平有效 |
| RCLINE1\_RD\_ADDR\_IN | 5 | BLOCK1对应的读端口的读地址 |
| RCLINE1\_RD\_DATA\_OUT | 128 | BLOCK1对应的读端口的读数据 |
| RCLINE2\_RD\_N\_IN | 1 | 可重构阵列  BLOCK2 | BLOCK2对应的读端口的读使能，低电平有效 |
| RCLINE2\_RD\_ADDR\_IN | 5 | BLOCK2对应的读端口的读地址 |
| RCLINE2\_RD\_DATA\_OUT | 128 | BLOCK2对应的读端口的读数据 |
| ................ |  |  | ....... |
| ................ |  | ....... |
| ................ |  | ....... |
| RCLINE9\_RD\_N\_IN | 1 | 可重构阵列  BLOCK9 | BLOCK9对应的读端口的读使能，低电平有效 |
| RCLINE9\_RD\_ADDR\_IN | 5 | BLOCK9对应的读端口的读地址 |
| RCLINE9\_RD\_DATA\_OUT | 128 | BLOCK9对应的读端口的读数据 |
| Write Port | | | |
| RCLINE0\_WR\_N\_IN | 1 | 可重构阵列  BLOCK0 | 可重构阵列BLOCK0对应的写端口的写使能，低电平有效 |
| RCLINE0\_WR\_ADDR\_IN | 5 | 可重构阵列BLOCK0对应的写端口的写地址 |
| RCLINE0\_WR\_MASK\_IN | 4 | 可重构阵列BLOCK0对应的写端口的按字Mask写信号，低电平有效。 |
| RCLINE0\_WR\_DATA\_IN | 128 | 可重构阵列BLOCK0对应的写端口的写数据 |
| RCLINE1\_WR\_N\_IN | 1 | 可重构阵列  BLOCK1 | 可重构阵列BLOCK1对应的写端口的写使能，低电平有效 |
| RCLINE1\_WR\_ADDR\_IN | 5 | 可重构阵列BLOCK1对应的写端口的写地址 |
| RCLINE1\_WR\_MASK\_IN | 4 | 可重构阵列BLOCK1对应的写端口的按字Mask写信号，低电平有效。 |
| RCLINE1\_WR\_DATA\_IN | 128 | 可重构阵列BLOCK1对应的写端口的写数据 |
| ............ |  |  | ............ |
| ............ |  | ............ |
| ............ |  | ............ |
| ............ |  | ............ |
| RCLINE9\_WR\_N\_IN | 1 | 可重构阵列  BLOCK9 | 可重构阵列BLOCK9对应的写端口的写使能，低电平有效 |
| RCLINE9\_WR\_ADDR\_IN | 5 | 可重构阵列BLOCK9对应的写端口的写地址 |
| RCLINE9\_WR\_MASK\_IN | 4 | 可重构阵列BLOCK9对应的写端口的按字Mask写信号，低电平有效。 |
| RCLINE9\_WR\_DATA\_IN | 128 | 可重构阵列BLOCK9对应的写端口的写数据 |

**2.3 计算引擎数据接口**



图2.7：计算引擎数据接口

计算引擎数据接口负责将外部数据导入计算引擎和导出计算结果，如图所示。两个片上模块是输入FIFO（I\_FIFO）和输出FIFO（O\_FIFO）。

### 2.3.1 输入FIFO（INFIFO）

#### 2.3.1.1 功能描述

输入FIFO是一个宽度为128b、深度为32，总计大小为0.5KB的FIFO，存储由片外导入的数据，RCA读出。

#### 2.3.1.2 接口定义

表2.11 输入FIFO接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signal | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RESET\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| EMPTY\_N\_OUT | 1 | RCA | 空信号，低电平有效 |
| RD\_N\_IN | 1 | 读使能，低电平有效 |
| RD\_ DATA\_OUT | 128 | 读数据 |
| Write Port | | | |
| FULL\_N\_OUT | 1 | IEDI | 满信号，低电平有效 |
| DATA\_CNT\_OUT | 5 | FIFO中当前数据个数 |
| WR\_N\_IN | 1 | 写使能，低电平有效 |
| WR\_ DATA\_IN | 128 | 写数据 |

### 2.3.2 输出FIFO（OUTFIFO）

#### 2.3.2.1 功能描述

输出FIFO是一个宽度为128b、深度为32，总计大小为0.5KB的FIFO，存储向片外导出的数据。输出FIFO中的数据由RCA写入。

#### 2.3.2.2 接口定义

表2.12 输出FIFO接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signal | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RESET\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| EMPTY\_N\_OUT | 1 | OEDI | 空信号，低电平有效 |
| DATA\_CNT\_OUT | 5 | FIFO中当前数据个数 |
| RD\_N\_IN | 1 | 读使能，低电平有效 |
| RD\_DATA\_OUT | 128 | 读数据 |
| Write Port | | | |
| FULL\_N\_OUT | 1 | RCA | 满信号，低电平有效 |
| WR\_N\_IN | 1 | 写使能，低电平有效 |
| WR\_ DATA\_IN | 128 | 写数据 |

## 2.4计算配置接口（Computing Configuration Interface）

### 2.4.1 功能描述

计算配置接口用于可重构阵列（包含四个Block）及宏单元与配置包解析器（CPP）之间的配置信息与使能信号的传递。

### 2.4.2 接口定义

表2.13 计算配置接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signal | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RST\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| 配置 | | | |
| CFG\_WDATA\_0\_IN | 128 | CPP | 阵列配置信息写数据 |
| CFG\_WADDR\_0\_IN | 10 | CPP | 阵列配置信息写地址： |
| CFG\_W\_ENABLE\_0\_N\_IN | 1 | CPP | 阵列配置信息写使能，低电平有效 |
|  | | | |
| BLOCK\_0\_READY\_N\_OUT | 1 | CPP | Block0准备就绪，可以接收新的配置，低电平有效 |
| BLOCK\_0\_ ENABLE\_N\_IN | 1 | CPP | Block0计算使能信号，表示可以执行当前的配置，低电平有效 |

表2.14配置寄存器定义

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 大类 | 类型 | 小类 | 大小/128b | 十进制累计 | 十六进制表示 | 首地址 |
| Block0 | Row0 | RCLINE | 2 | 2 | 0x2 | 0x0 |
| BENES0 | 2 | 4 | 0x4 | 0x2 |
| BENES1\_0 | 7 | 11 | 0xB | 0x4 |
| BENES1\_1 | 7 | 18 | 0x12 | 0xB |
| CTR | 1 | 19 | 0x13 | 0x12 |
| Row1 | RCLINE | 2 | 21 | 0x15 | 0x13 |
| BENES0 | 2 | 23 | 0x17 | 0x15 |
| BENES1\_0 | 7 | 30 | 0x1E | 0x17 |
| BENES1\_1 | 7 | 37 | 0x25 | 0x1E |
| CTR | 1 | 38 | 0x26 | 0x25 |
| Row2 | RCLINE | 2 | 40 | 0x28 | 0x26 |
| BENES0 | 2 | 42 | 0x2A | 0x28 |
| BENES1\_0 | 7 | 49 | 0x31 | 0x2A |
| BENES1\_1 | 7 | 56 | 0x38 | 0x31 |
| CTR | 1 | 57 | 0x39 | 0x38 |
| Row3 | RCLINE | 2 | 59 | 0x3B | 0x39 |
| BENES0 | 2 | 61 | 0x3D | 0x3B |
| BENES1\_0 | 7 | 68 | 0x44 | 0x3D |
| BENES1\_1 | 7 | 75 | 0x4B | 0x44 |
| CTR | 1 | 76 | 0x4C | 0x4B |

# 配置控制模块

配置控制器作为计算引擎的配置单元，会根据配置包配置计算引擎，配置控制器的主要工作流程如图3.，配置控制器的主要功能如下：

1. 配置信息的初始化，配置包的初始化，以及配置包的解析与配置信息的发送；
2. 解析配置包内容，发送相应的配置信息到计算引擎切换配置；



图3.1：配置控制器的工作流程

## 3.1配置信息接口（Configuration Interface）（未实现）

### 3.1.1 功能描述

配置信息接口（CI）作为主控CPU通过AXI总线与配置控制器进行交互的接口，主要完成以下功能：

1. 初始化配置控制器中的配置存储器。系统上电之后，主控CPU通过CI接口把所有的配置信息写入配置控制器的配置存储器中。
2. 初始化配置包存储器。功能2必须在功能1完成之后进行。主控CPU通过CI接口把所有的配置包写入配置包存储器中。

### 3.1.2接口定义

表3.1Configuration Interface接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signal | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RST\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| AHB | | | |
| HADDR\_IN | 32 |  | AHB总线地址 |
| HTRANS\_IN | 2 | AHB总线传输类型 |
| HWRITE\_IN | 1 | AHB总线读写控制 |
| HSIZE\_IN | 3 | AHB总线传输大小 |
| HBURST\_IN | 3 | AHB总线传输长度 |
| HWDATA\_IN | 32 | AHB总线数据输入 |
| HSEL\_IN | 1 | AHB总线片选 |
| HRDATA\_OUT | 32 | AHB总线读出数据 |
| HREADY\_OUT | 1 | AHB总线响应 |
| HRESP\_OUT | 2 | AHB总线响应类型 |
| Configuration Memory Initialization | | | |
| CM\_DATA\_OUT | 128 | CM | CM初始化数据 |
| CM\_ADDR\_OUT |  | CM初始化数据写地址 |
| CM\_WREN\_N\_OUT | 1 | CM初始化数据写使能，低电平有效 |
| Configuration Packet Memory Initialization | | | |
| CFG\_DATA\_OUT | 32 | CPM | CPM初始化数据 |
| CFG\_ADDR\_OUT | 7 | CPM初始化数据写地址 |
| CFG\_WREN\_N\_OUT | 1 | CPM初始化数据写使能，低电平有效 |

## 3.2 配置包解析（Configuration Packet Parser）

### 3.2.1功能描述

解析由配置接口写入的配置包，配置控制器中的配置解析模块对配置包进行解析，得到对应的信息对阵列进行配置。配置包的格式如表：

表3.1 配置包的格式

|  |  |  |
| --- | --- | --- |
| 配置位置 | | 含义 |
| 包头 | | |
| Word0[3:0] | | 算法编号，如：00000：AES算法等 |
| Word0[11:4] | | 此算法的配置包的个数 |
| Word0[31:9] | | 保留 |
| Word1[15:0] | | BENES16的首地址 |
| Word1[31:16] | | BENES128的首地址 |
| Word2[15:0] | | LUT的首地址 |
| Word2[31:16] | | Control的首地址 |
| 子包头格式 | | |
| Word3[31:0] | | 此配置包包含哪几个部分的配置字（主要是每行的配置和LUT的配置），最少1个，最多41个；如：  0x0000000001：仅包含RCline0的配置字  0x0000000011：包含RCline4和RCline0的配置字  ……  0x1FFFFFFFFF：包含40行计算单元和LUT |
| Word4[8:0] | |
| Word4[11:9] | | 此配置所需要的同步机制  000：不要同步  0001-111保留 |
| Word4[31:12] | | 此配置包使能哪几个部分的配置字（只包含行单元），最少1个，最多40个；如：  0x000000001：仅使能line0的配置字  0x000000011：使能RCline4和RCline0的配置字  ……  0xFFFFFFFFF：使能40行RCline |
| Word5[20:0] | |
| Word5[26:21] | | RCline起始行数（1~40） |
| Word5[27] | | 保留 |
| Word5[31:28] | | LUT的使用套数：  00：只用一套LUT  01：使用两套LUT  10：使用四套LUT  11：使用八套LUT |
| Word6[7:0] | | LUT0的配置编号 |
| Word6[15:8] | | LUT1的配置编号 |
| Word6[23:16] | | LUT2的配置编号 |
| Word6[31:24] | | LUT3的配置编号 |
| Word7[7:0] | | LUT4的配置编号 |
| Word7[15:8] | | LUT5的配置编号 |
| Word7[23:16] | | LUT6的配置编号 |
| Word7[31:24] | | LUT7的配置编号 |
| 配置字 | | |
| Word8[5:0] | RCline0的配置字 | RCline0包含哪几种配置，最少1种最多6种，分别为：  000001：只有PE的配置  000010：只有Benes16\_0的配置  000100：只有Benes16\_1的配置  001000：只有Benes128\_0的配置  010000：只有Benes128\_1的配置  100000：只有行控制的配置 |
| Word8[11:6] | RCline0的配置周期数 |
| Word8[15:12] | 保留 |
| Word8[23:16] | Rcline0的RC配置信息编号 |
| Word8[31:24] | RCline0的Benes16\_0配置信息编号 |
| Word9[7:0] | RCline0的Benes16\_1配置信息编号 |
| Word9[15:8] | RCline0的Benes128\_0配置信息编号 |
| Word9[23:16] | RCline0的Benes128\_1配置信息编号 |
| Word9[31:24] | RCline0的行控制的配置信息编号 |

配置包不定长，配置算法包头为3个word，标志了该算法名称，算法中包含的配置套数，以及算法的各类配置信息的起始地址。

配置子包头为5个word，标志了该配置包所配置的RCline的编号和它们之间的同步机制，还包含了算法的每张DFG图中需要使能的RCline编号，同时还包括了每张DFG图中LUT的使用情况。每行RCline的配置字为2个word，64 bit，标志了当前这行RCline的PE、Benes16、Benes128\_0、Benes128\_1和行控制的配置编号。

当在配置切换的时候，被换掉的RCline按照新的配置运算，其余的RCline按照之前的配置运算，如果其余RCline不需要运算，应该配置这些RCline的使能标志失效。

配置包解析模块在接收到CI接口发送的配置包存储器初始化完成的信号之后，从配置包存储器中读取配置包开始解析。在接收到计算引擎中发送过来的准备信号，开始从CM中读取配置信息发送到相应的配置接口。

### 3.2.2接口定义

表3.3 配置包解析单元接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signal | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RST\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| CPM\_ADDR\_OUT | 5 | CPM | 读配置包地址 |
| CPM\_DATA\_IN | 32 | CPM | 读配置包数据 |
| CPM\_READY\_N\_IN | 1 | CI | 配置包初始化完成信号，低电平有效 |
| Write Port | | | |
| CFG\_WDATA\_OUT | 128 | CCI | Block\_x配置信息写数据 |
| CFG\_WADDR\_OUT | 8 | CCI | Block\_x配置信息写地址 |
| CFG\_W\_EN\_N \_OUT | 1 | CCI | Block\_x配置信息写有效 |
| CFG\_RDATA\_IN | 128 | CM | Block\_x配置信息读数据 |
| CFG\_RADDR\_OUT | 8 | CM | Block\_x配置信息读地址 |
| CFG\_R\_EN\_N\_OUT | 1 | CM | Block\_x配置信息读使能 |
| RCline0\_READY\_N\_IN | 1 | CCI | 计算引擎RCline0准备就绪，可以接受配置 |
| RCline0\_EN\_N\_OUT | 1 | CCI | 计算引擎RCline0计算使能信号 |
| RCline1\_READY\_N\_IN | 1 | CCI | 计算引擎RCline1准备就绪，可以接受配置 |
| RCline1\_EN\_N\_OUT | 1 | CCI | 计算引擎RCline1计算使能信号 |
| …… |  |  |  |
| RCline39\_READY\_N\_IN | 1 | CCI | 计算引擎RCline39准备就绪，可以接受配置 |
| RCline39\_EN\_N\_OUT | 1 | CCI | 计算引擎RCline39计算使能信号 |

## 3.3 配置存储器

### 3.3.1功能描述

配置存储器Configuration Memory（CM）由一片数据位宽为128b，地址位宽为8b，总计大小为128\*28=32Kb的存储器构成。CM存储某一特定领域算法中所有的配置信息，包括8入8出LUT的配置、Benes网络的配置、PE的配置以及RCline行控制的配置。在系统上电之后，主控CPU把所有的配置信息写入CM，存放格式如图3.2。PE的配置、Benes网络的配置、8入8出LUT以及行控制的配置组成一组，构成Block的一套配置，按照自然序编号存放在配置存储器中。



图3.2：配置信息在配置存储器中的组织格式

CM的初始化由配置信息接口（Configuration Interface）完成，初始化完成之后，配置控制器的配置包解析模块根据解析的配置包，从CM中读取配置信息发送到计算引擎的配置接口。

如果配置包里需要计算引擎的反馈信息参与配置，计算引擎反馈信息解析模块会解析反馈信息，从CM中读取配置信息到计算引擎的配置接口。

### 3.3.2接口定义

表3.5 配置存储器接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signal | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RESET\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| CPP\_RD \_N\_IN | 1 | CPP | CPP读使能，低电平有效 |
| CPP\_RD\_ADDR\_IN | 8 | CPP读地址 |
| CPP\_RD\_DATA\_OUT | 128 | CPP读数据 |
| Write Port | | | |
| CI\_WR\_ N\_IN | 1 | CI | CI写使能，低电平有效 |
| CI\_WDATA\_IN | 128 | CI写数据 |
| CI\_WADDR\_IN | 8 | CI写地址 |

## 3.4 配置包存储器

### 3.4.1功能描述

配置包存储器Configuration Packet Memory（CPM）由一片数据位宽为32bit，地址位宽为7，总计大小为4Kb的存储器构成。CPM存储某一特定领域算法中所有的配置包。在系统上电之后，配置信息初始化完成之后，主控CPU通过CI接口把配置包写入配置包存储器中，存放格式如图3.5。



图3.5：置包在配置包存储器中的组织格式

### 3.4.2接口定义

表3.6 配置包存储器接口定义

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 位宽 | 源/目标 | 描述 |
| Global Signal | | | |
| CLK\_IN | 1 | - | 系统时钟，上升沿有效 |
| RESET\_N\_IN | 1 | - | 异步复位信号，低电平有效 |
| Read Port | | | |
| CPP\_RD \_N\_IN | 1 | CPP | CPP读使能，低电平有效 |
| CPP\_RD\_ADDR\_IN | 7 | CPP读地址 |
| CPP\_RD\_DATA\_OUT | 32 | CPP读数据 |
| Write Port | | | |
| CI\_WR\_ N\_IN | 1 | CI | CI写使能，低电平有效 |
| CI\_WR\_ADDR\_IN | 7 | CI写地址 |
| CI\_WR\_DATA\_IN | 32 | CI写数据 |

# 算法映射与评估

详见附件2