# CGRA.vsion High Level Description

# 目录

# 修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 时间 | 修订人 | 备注 |
| V0.1 | 2020.11.11 | 唐士斌 | 创建文档、硬件编程模型 |
| V0.2 | 2020.12.31 | 唐士斌 | 更新RCE基本结构、RCE执行流程 |
| V0.3 | 2021.1.11 | 唐士斌 | 更新RCE访存通路 |
| V0.4 | 2021.1.21 | 唐士斌 | 更新RCE算子、RCE执行TASK流程 |
| V1.0 | 2021.06.21 | 唐士斌 | 更新RCE的架构文档 |
| V1.1 | 2021.07.21 | 唐士斌 | 更新RCE的架构文档：   1. 同步机制 |
|  |  |  |  |

# 问题记录

1. 控制流和数据流的问题；当前的RCE均是基于数据通道的，控制通道该如何铺开？具体什么能力？
2. 2021-06-21纪要



# 硬件编程模型（编译器提供更高级的编程抽象模型）

RCE的编程模型是硬件描述编程模型：

1. 在RCE上执行的程序需要首先表达成LOOP\_DFG+DFG（DFG，Data Flow Graph）形式，如所示：DFG表达一个程序块Program Block；

LOOP表示循环控制功能；

DFG与DFG之间，LOOP与DFG之间，通过BEAT实现同步，BEAT为三态的两比特信号{00(first),01(valid),10(last)}；一个循环，至少包含两个BEAT（frist，last），中间可以允许0~N个valid；

1. DFG表达了一个程序的数据通路片段我们称为Program Block，Program block可以包含以下信息：
   1. 输入/输出，包含：
      1. 输入/输出变量；（对应DFG的parameter）
      2. 输入/输出数组；（通过LSU中的Load操作导入DFG中）
   2. 数据表达与数据处理功能，对应数据流图DFG，包含：
      1. 数据变量：
         * 数据常量；（对应DFG的parameter）
         * 临时变量；（对应DFG中算子的本地寄存器）
      2. 数据操作，包含：
         * ALU算子：add/sub/abs/and/or/not/quant等
         * MAC算子
         * 特殊算子：div/sqrt/clz/clo等
         * 分支算子：sel，该分支仅能用于数据分支不能反馈到控制通路；
2. DFG的控制功能：
   1. 单层循环：
      1. 一个SLOOP算子；
   2. 两层循环：
      1. 一个DLOOP算子；
   3. 多层循环：
      1. SLOOP之间组合；



图 1 RCE计算示意图

# RCE（Reconfigurable Computing Engine）Architecture

## RCE基本结构



图 2 RCE Architecture

RCE基本结构主要分成了四个主要功能：

1. CMU（控制管理单元，Controller Management Unit），负责一个完整应用程序代码的执行。包含了CU与MU两个核心组件。
   1. CU（Controller Unit），用于流程控制，主要执行程序中的控制流代码。CU的典型使用流程包含指令流与数据流，通常包含ITCM （Instruction Tightly Coupled Memory,）与DTCM（Data Tightly Coupled Memory）：
      1. ITCM，映射在系统地址空间，实际存储内容由编译器决定；
      2. DTCM，映射在系统地址空间，实际存储内容有编译器与程序员共同管理。堆栈区域由程序员负责，其他部分由编译器管理。
   2. MU（Management Unit），用于管理、调度加速器，主要用于管理数据流。MU的典型使用流程用到CSPM（Context Scatch-pad-memory）与PB（Parameter Buffer）
      1. CSPM是一段RCE Context缓存区，在典型执行流程中不映射到系统地址空间，由编译器调度管理。因此编译器需要管理一份“缓存区地址空间Context”到“系统地址空间Context”的映射关系，以保证RCE可以正确工作；

备注：考虑到系统软件的简单，优先考虑通过硬件管理“映射关系”

* + 1. PB是一段实时的参数空间，CU将TASK执行需要的参数通过PB/MU传给PEA阵列用于执行。

1. PEA（执行单元阵列，Processing Element Array），主要负责加速数据流代码的执行。PEA主要包含PE、SEL、RC、LSU、Permutation等核心组成单元，具体功能如下：
   1. PE（处理单元，Processing Element）是具备数据处理能力的功能单元的统称，通过配置可以实现不同的计算功能。当前的PE主要分成以下几类：
      1. LOOP算子，通过“时钟”、“Beat”与“循环控制三元组”实现循环控制。控制三元组{Initial，END，Step}，分别表示：
         * + Initial：循环起始变量；
           + End：循环结束变量；
           + Step：循环步进；
         * Beat，表示在DFG中流动的数据以及附着在数据通路上的控制信号。Beat是一个三元组{State，C，Data}，分别表示：

State：状态信号；状态信号循环控制算子产生，在DFG中不发生变化，不同的PE根据算子功能与State状态执行独立的功能。

First：循环开始，发出的第一个Beat状态；

Last：循环结束，发出的最后一个Beat状态；

Valid：(第一个Beat，最后一个Beat)中间所有Beat的状态；

C进位、借位信号：

Data：数据负载；

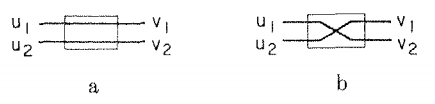
RCE提供两类LOOP算子：

* + - * 单层循环算子：
      * 两层循环算子：

备注：存在风险，相当于硬化了For循环的功能；然而循环的本质是While，对while的支持天然存在缺陷。

* + 1. ALU算子包括以下功能：
       - Arithmetical：ADD/SUB/ABS/NEG
       - Shifter：SRL/ARL/SLL
       - Condition：CEQ/CLT/CLE
       - Bitwise Logical：AND/OR/NOT/XOR
       - Comparison：MAX/MIN
       - Conversion：W2B/W2H/H2B/T2FW/ROUND
    2. MAC算子包括以下功能：
       - MUL：乘法
       - MADD：乘加
    3. SEL算子包括以下功能：
       - 八选一（SEL81）：
       - 四选一（SEL41）：
       - 二选一（SEL21）：
    4. SPU算子包括以下功能：
       - DIV/MOD：除法、取模
       - CADD/CSUB/ADDC/SUBC：带进位（借位）功能操作
       - SQRT：开方
       - Floor：移位下取整
       - Ceiling：移位上取整
       - CLO：前导1
       - CLZ：前导0
  1. Pnet（置换网络，Permutation Network），置换网络将所有的PE连接到一起。通过配置不同的配置信息PNet与PE一起组成数据流图（DFG，Data flow graph）执行数据流计算。
     1. Pnet结构图如图 1所示。置换网络由N\*Log2N个置换节点组成，每个置换节点，通过1bit配置完成输入u1/u2到输出v1/v2的置换。
     2. Pnet优点：
        + 全互连的能力，可以连接任意的一对输入与输出；
        + 配置信息少，N输入N输出的置换网络仅需要N\*Log2N bits配置信息。
        + 资源少，MUX与线少；
     3. Pnet缺点：
        + 仅能提供置换能力，不能实现多播（一对多）传输；
  2. RC（寄存器链，Register Chain），实现数据打拍与多播功能。每个寄存器链都是一输入M输出，RC可以实现0~M-1拍的延迟，M个输出可以任意选择0~M-1级寄存器上的数据输出；

RC与Pnet组合可以实现任意两个节点的互连、延迟打拍与数据组播。



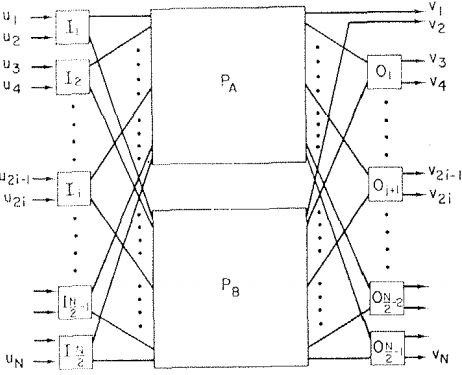


图 3 Permutation Network

* 1. Load/Store，根据配置模式选择寻址方式计算内部缓存空间地址，从缓存中读取数据，并同时返回到计算阵列中。Load/Store单元需要具备的特性如下：
     1. Load/Store单元最多同时从计算单元接收M个访存请求；
     2. 支持多种寻址模式：
        + 立即数寻址Load/Store[Offset]，Offset来着Load/Store局部寄存器；
        + 寄存器寻址Load/Store[Base]，Base来自计算阵列；
        + 基址相对寻址Load/Store[Base+Offset]，Base来自计算阵列，Offset来自Load/Store局部寄存器；
     3. 支持访存冲突检测与仲裁，假设LD/ST模块发起M个请求同时访问N个SPM-Bank。LD/ST需要将M个请求加入到N个访存队列中，将冲突的请求顺序完成。等待M个请求均完成以后，同时将结果返回到计算阵列中。~~LD/ST需要反馈暂停信息给CMU，CMU控制阵列暂停计算~~。LD/ST需要反馈信息给PEA，PEA暂停整个阵列的执行。
     4. 支持数据格式转换。当DSPM中的数据是以字节（Byte）或者半字（half word）为粒度的数组时，
     5. 支持高带宽的连续取数模式。当四个地址A1、A2、A3、A4是连续的128bit，并且起始地址是128bit对齐的。

1. DMA（Direct Memory Access），是RCE实现内部缓存（DSPM）与外部Memory交换数据的模块。分成两个主要的功能RDMA与WDMA。
   1. RDMA将外部Memory的数据搬到内部DSPM中。在两个地址空间中同时支持Gather/Scatter（性能损失：Gather/Scatter的一次数据长度不要低于128B，否则性能会降低非常严重）。
   2. WDMA将内部DSPM的数据搬到外部Memory中。在两个地址空间中同时支持Gather/Scatter（性能损失：Gather/Scatter的一次数据长度不要低于128B，否则性能会降低非常严重）。
   3. 支持AXI总线接口：
      1. 支持Outstanding；
      2. 地址（Address）与数据长度（Length）需要支持1字节（Byte）粒度；（目前仅RDMA有此需求，WDMA尚未明确该需求）；
   4. 支持ISP Linebuffer的特殊接口；
   5. DMA支持通途的压缩、解压缩功能；（待确认澄清）；
2. DSPM（片上数据缓存，Data Scratch-Pad-Memory）
   1. DSPM是一块RCE独有的Buffer，典型的执行过程中不映射到系统地址空间（Debug功能可以映射到系统地址空间）；Buffer地址空间通过编译器以软件的方式管理；
   2. 支持多BANK划分；
   3. 支持多Master（Load/Store、DMA、CMU）访问DSPM的视角一致；
   4. 可配置的地址交织方式
      1. ~~中位交织~~高位交织；[9:0]：高位[9:8]，地位[1:0]，[6:5]

## RCE中TASK执行流程

### 概念介绍

执行模型中核心概念介绍：

* Kernel，代表一段可以在RCE-PEA/LSU上执行的程序段；Kernel可以与C代码循环对应：
  + Kernel由代码块（basic blocks）、循环控制、分支控制组成；
    - 代码块：是一组表达式（statements）的集合，顺序执行；
    - 循环控制支持For循环；
    - 分支控制限制支持If-else；
  + Kernel不能嵌套；
  + Kernel内部支持变量声明；内部不支持数组声明；
  + Kernel的输入支持：常量、变量、数组；
  + Kernel的输出支持：常量、变量、数组；
* TASK，表示一个独立的任务，可以完成一个功能函数或者完成一个算法的实现。
  + TASK可以由一个或者多个Kernel组成；
  + 典型的TASK由Kernel数据准备、Kernel执行、Kernel结果回写组成；
    - 数据准备，包括DMA操作、CU通过MU向Kernel传递数据（典型为参数）；
    - Kernel执行，由kernel配置驱动PEA与LSU完成功能实现；
    - 数据回传，通过DMA复制数据进入系统地址空间；

### TASK执行流程

本节通过图示的形式介绍一个TASK的典型执行流程；



图 4 TASK的C代码示意（fir滤波）



图 5 TASK映射示意



图 6 TASK执行流程示意

图 4展示了一个TASK在RCE上的执行流程：

1. CU向MU发送ContextDMA参数信息，包括：
   1. DDR地址，系统地址空间地址；
   2. CSPM地址，CSPM缓存空间地址；
   3. Length，数据长度；
2. CU向MU发送ContextDMA命令；
3. MU将ContextDMA操作加入指令队列，等待调度执行；
4. DMA执行ContextDMA命令，将Context从DDR搬运到CSPM中；
5. CU向MU发送RDMA参数信息，包括：
   1. DDR地址，系统地址空间地址；
   2. DSPM地址，DSPM缓存空间地址；
   3. Length，数据长度；
6. CU向MU发送RDMA命令；
7. MU将RDMA操作加入指令队列，等待调度执行；
8. DMA执行RDMA命令，将Input Data从DDR搬运到DSPM中；
9. CU向MU传递PEX的参数信息，MU将参数保存到PB（parameter buffer）中，包括：
   1. 运行时，调用TASK的参数；
10. CU向MU发送PEX命令；
11. MU将PEX（对应CSPM信息、PB信息）加入到指令队列，等待调度执行；
12. MU发射PEX到PEA/LSU单元，驱动PEA/LSU执行TASK中Kernel对应的任务；从DSPM中读取数据，经过计算，将结果写回到DSPM中；
13. CU向MU发送WDMA命令；
14. MU将WDMA操作加入指令队列，等待调度执行；
15. DMA执行WDMA命令，将Output Data从DSPM中搬回到DDR的系统地址空间。

至此，完成整个TASK的执行。一个典型的TASK至少包括：CDMA、RDMA、PEX、WDMA等四个操作。

## PEA执行DFG流程

如“TX511-RCE HLD (high level description)流程说明”所示。

## RCE访存通路

### 访存通路介绍

本节通过一个典型的执行流程说明，RCE的访存通路能力。RCE的访存通路需要在三个方面提供竞争力：

1. 总线的带宽；DMA提供三种访存方式，Contiguous address space、Gather、Scatter，任意一种方式均需要将总线带宽充分利用。为了充分利用总线带宽，软硬件需要满足以下能力：
   1. Burst能力，单次的连续访问需要以Burst方式进行。因此单次的数据长度BurstLength = BurstNumber \* bus\_width；
   2. Outstanding能力，Outstanding能力分成两种情况：
      1. 单次连续地址的数据访问超过Burst支持的最大数据长度。将一次连续的访存，拆成多个Burst，为了充分利用带宽连续发送多个Outstanding请求出去。其中，Outstanding能力需要满足以下等式：

BusBandwidth \* Latency = MinOutstandingBurstNumber \* BurstLength

其中，

BusBandwidth表示我们需要的总线带宽，典型情况为最大带宽；

Latency表示Master访问Memory的数据通路延迟；

MinOutstandingBurstNumber表示最小Outstanding Burst请求的数量；

BurstLenght表示burst数据长度，典型为最大Burst数据长度；

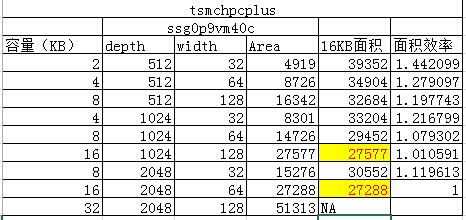
备注：Outstanding情况下，需要足够大的数据Buffer保证Master不反压总线。

* + 1. Gather、Scatter，多次连续地址的数据访问。多个Gather请求，需要以Outstanding的请求方式发送出去。此时，

BusBandwidth \* Latency = OutstandingBurstNumber \* BurstLength

中BurstLength受限于Gather/Scatter请求，因此BusBandwidth可能无法充分利用。

1. 高效ASIC数据格式转换；DMA的首要任务是充分利用总线带宽，其次，DMA要提供典型的数据变化能力，实现DSPM内部的数据格式转换。基于DSPM存储空间，DMA需要提供多种数据格式变换的能力，包括：
   1. MatrixTranspose；基于DSPM-BANK的最大位宽，实现基于子矩阵转置的矩阵转置；
   2. SPM Gather；
   3. SPM Scatter；
   4. SPM码位倒序；
   5. **待工具链同事补充**；
2. LSU模块实现PEA与DSPM之间带宽匹配，处理PEA与DSPM在物理实现中“位宽”与“数量”的差异。
   1. 典型情况下，DSPM总容量为512KB，分成32个BANK，单个BANK容量为16KB，规格为128bit宽\*1024深；如下是TSMC28HPCPLUS，ssg0p9vm40c的SRAM面积情况：
      1. 最大提供：2048\*144；
      2. 典型情况下：1024\*128在面积、容量、位宽上是较好的选择；



* 1. PEA阵列通过LSU模块访问DSPM，LSU提供的能力包括：
     1. 在“PEA-LD/ST请求”与“DSPM-BANK访存端口”之间匹配“数量”与“位宽”的差别。实现访存带宽的匹配，满足：

LD/ST \* 32bit = BANK\_Number \* BANK\_width；

* + 1. LSU具备处理Memory Bank冲突的能力。LSU允许多拍完成所有的LD/ST请求，并反压住PEA阵列，当所有请求的数据准备好以后，同时反馈给PEA阵列，PEA阵列不感知多拍的处理。（目的：简化软件编程，尤其是边界处理的困难，但是性能优化仍需要避免BANK冲突）；
    2. CouplingOperations：最大支持128个32bit的LD/ST；此时需要满足以下条件：
       - 128个LD/ST，分成32组CouplingLoad或者CouplingStore，每个CouplingOperations访问128bit宽，128bit地址对齐的DSPM空间。
       - LD与ST之间，无法组成CouplingOperations；
    3. LoneOperation：最大支持32个32bit的LD/ST；此时
       - 32个LD/ST，任意两个LD或者任意两个ST不访问连续的128bit地址空间（该连续地址空间128bit地址对齐）；
    4. MixOperation：同时存在CouplingOperation与LoneOperation，LSU支持对LD/ST自动分类：CouplingOperation或者LoneOperation；在不冲突的情况下，同时发起所有的CouplingOperation与LoneOperation；

### 典型访存通路介绍



图 7 RCE执行的典型流程

如图 2所示，典型的RCE执行流程分成如下步骤：

1. DMA从DDR（或者OCRAM）搬运数据到片上的DSPM中。需要DMA充分利用总线带宽。
2. DSPM-DMA通过ASIC的数据搬运在DSPM中完成数据格式变换。当ASIC功能无法支持时，通过阵列完成。
3. 阵列执行“循环加速任务”，通过LSU模块实现PEA与DSPM之间带宽匹配，处理PEA与DSPM在物理实现中“位宽”与“数量”的差异。

### RCE同步机制

CU与MU是RCE内部的两个主要的控制Master，CU与MU通过控制流同步共同完成一项工作；

#### 控制同步

##### CU与MU间同步

1. CU通过LOAD/STORE访问MU的寄存器，给MU发送执行命令；
2. CU访问MU的启动寄存器，支持阻塞式启动与非阻塞式启动两种：
   1. 阻塞式启动，当且仅当CU发送给MU的命令执行完成后，CU的STORE操作才会完成；
   2. 非阻塞时启动，当CU给MU 发送启动命令后，CU的STORE操作立即完成，CU可以继续执行后续操作；

##### MU内部同步

1. MU内部的控制流同步，通过指令队列完成。MU内部有四条指令队列：RDMA、WDMA、B2B、PEX，不同的队列之间的指令可以并行执行，同一个指令队列内的指令按照程序序顺序执行；
2. 并行的指令队列之间通过Sync.id
   1. PEX与B2B有两个Sync.id；
   2. RDMA与WDMA仅有一个Sync.id；

#### 数据同步

1. RCE内部无数据同步机制，包括但不限于访问SRAM的仲裁、Coherence与Consistency等；CU与MU独占并分别管理各自的SRAM；

CU管理的SRAM资源包括：ITCM、DTCM；

MU管理的SRAM资源包括：CSPM、DSPM；

1. 通过“控制流”管理“数据同步”
   1. CU需要访问DSPM的数据时，需要保证CU发到MU中的命令（RDMA/WDMA/B2B/PEX）均已经完成，否则将造成DSPM访问冲突；
   2. “CU通过DMA将PEA的执行结果从DSPM搬运到DTCM（此非典型通路）”或者“CU通过DMA将一些数据从DDR搬运到ITCM或DTCM（此过程是一种典型的计算过程）”，需要CU执行阻塞式DMA，当发送DMA以后CU就挂起任务，直到DMA执行完成，CU再继续执行计算；此时，DMA与CU是串行工作，DMA仅作为CU一个高速的搬运数据部件，并不能DMA与CU并发工作。

# 特性说明

# CGRA.Vsion 接口与时序

# 内部模块功能与接口

# 低功耗（Low Power）

# DFx特性

# SRAM

# 开发进度

# 附录

## ISP-CGRA