**Core中ArrayST对象的参数设置**

[1. BranchPredictor中的globalBPT 2](#_Toc532140726)

[2. BranchPredictor中的L1\_localBPT 4](#_Toc532140727)

[3. BranchPredictor中的L2\_localBPT 6](#_Toc532140728)

[4. BranchPredictor中的choose 8](#_Toc532140729)

[5. BranchPredictor中的RAS 10](#_Toc532140730)

[6. InstFetchU中的BTB 11](#_Toc532140731)

[7. InstFetchU中的IB 12](#_Toc532140732)

[8. RegFU中的IRF 13](#_Toc532140733)

[9. RegFU中的FRF 14](#_Toc532140734)

[10. RegFU中的RFWIN（应该是为了保存异常发生时的寄存器堆使用） 15](#_Toc532140735)

[11. SchedulerU中的int\_inst\_window（按序情况下） 16](#_Toc532140736)

[12. SchedulerU中的int\_inst\_window（乱序情况下） 17](#_Toc532140737)

[13. SchedulerU中的fp\_inst\_window（只在乱序中定义） 19](#_Toc532140738)

[14. SchedulerU中的ROB (只在乱序中定义) 20](#_Toc532140739)

[15. RENAMING中的FRAT，RRAT，RAHT解释 22](#_Toc532140740)

[16. RENAMINF中的iFRAT（PRF+RAMbased） 23](#_Toc532140741)

[17. RENAMINF中的iFRAT（PRF+CAMbased） 24](#_Toc532140742)

[18. RENAMINF中的iFRAT（RS+RAMbased） 25](#_Toc532140743)

[19. RENAMINF中的iFRAT（RS+CAMbased） 26](#_Toc532140744)

[20. RENAMINF中的fFRAT（PRF+RAMbased） 27](#_Toc532140745)

[21. RENAMINF中的fFRAT（PRF+CAMbased） 28](#_Toc532140746)

[22. RENAMINF中的iFRAT（RS+RAMbased） 29](#_Toc532140747)

[23. RENAMINF中的iFRAT（RS+CAMbased） 30](#_Toc532140748)

[24. RENAMINF中的iRRAT (PRF，没有RS) 31](#_Toc532140749)

[25. RENAMINF中的fRRAT（PRF，没有RS） 32](#_Toc532140750)

[26. RENAMINF中的ifreeL(PRF，只有RAM类型的) 33](#_Toc532140751)

[27. RENAMINF中的ifreeL(RS, 这种情况下，freelist同时包括了int和fp) 34](#_Toc532140752)

[28. RENAMINF中的ffreeL（PRF，在RS的情况下是统一的freelist） 35](#_Toc532140753)

[29. LoadStoreU 中的LSQ 36](#_Toc532140754)

[30. LoadStoreU 中的LoadQ（仅在乱序处理器中定义） 37](#_Toc532140755)

[31. MemManU中的itlb 38](#_Toc532140756)

[32. MemManU中的dtlb 39](#_Toc532140757)

[33. InstCache中的caches 40](#_Toc532140758)

[34. InstCache中的ifb 41](#_Toc532140759)

[35. InstCache中的missb 42](#_Toc532140760)

[36. InstCache中的prefetchb 43](#_Toc532140761)

[37. DataCache中的caches 44](#_Toc532140762)

[38. DataCache中的ifb 45](#_Toc532140763)

[39. DataCache中的missb 46](#_Toc532140764)

[40. DataCache中的prefetchb 47](#_Toc532140765)

[41. DataCache中的wbb（当cache采用Write\_back策略时定义） 48](#_Toc532140766)

# BranchPredictor中的globalBPT

|  |  |
| --- | --- |
| 参数 | 解释 |
| globalBPT = new ArrayST(&interface\_ip, "Global Predictor", Core\_device, coredynp.opt\_local, coredynp.core\_ty); | |
| interface\_ip.assoc  nterface\_ip.pure\_cam | 需要设置array的相连度和类型  Assoc=1，因为分支预测器中的预测表都是直接映射  Pure\_cam=false，因为预测表不是CAM类型的存储单元 |
| interface\_ip.is\_cache  interface\_ip.pure\_ram | 这两个参数需要根据core的类型进行设置  当core是多线程时（不是多线程的时候结果相反）  Is\_cache=true，因为此时需要为表项增加tag域，指代不同的线程，这种结构是类cache的结构  Pure\_ram=false，此时不是单纯的锁存器类型的结构 |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 当core的类型是多线程时，表项需要增加tag域，因此需要给出tag域的信息  Specific\_tag=1，代表使用tag，并且将会给出tag的宽度  Tag\_w= int(log2(coredynp.num\_hthreads)+ EXTRA\_TAG\_BITS)  //EXTRA\_TAG\_BITS为cacti中的参数，const int = 5 |
| Data(临时使用) | int(ceil(XML->sys.core[ithCore].predictor.global\_predictor\_bits/8.0))  向上取整，以字节为单位，计算每个表项的宽度 |
| interface\_ip.line\_sz | Data  单位字节，每个cache line的大小 |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].predictor.global\_predictor\_entries  单位字节，整个cache的大小，表项数\*每个表项的大小 |
| interface\_ip.nbanks | 1  Bank的个数，预测表都没有分bank |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast  fast(2): 数据和tag同时访问（具体见interface\_ip解析） |
| interface\_ip.  throughput | 1.0/clockRate  吞吐量 |
| interface\_ip.latency | 1.0/clockRate，延迟 |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0  读写端口数 |
| interface\_ip.  num\_rd\_ports | 单独的读端口数，=coredynp.predictionW  prediction\_width，core的参数，一般设置为1 |
| interface\_ip.  num\_wr\_ports | 单独的写端口数，=coredynp.predictionW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# BranchPredictor中的L1\_localBPT

|  |  |
| --- | --- |
| 参数 | 解释 |
| L1\_localBPT = new ArrayST(&interface\_ip, "L1 local Predictor", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.assoc  nterface\_ip.pure\_cam | 需要设置array的相连度和类型  Assoc=1，因为分支预测器中的预测表都是直接映射  Pure\_cam=false，因为预测表不是CAM类型的存储单元 |
| interface\_ip.is\_cache  interface\_ip.pure\_ram | 这两个参数需要根据core的类型进行设置  当core是多线程时（不是多线程的时候结果相反）  Is\_cache=true，因为此时需要为表项增加tag域，指代不同的线程，这种结构是类cache的结构  Pure\_ram=false，此时不是单纯的锁存器类型的结构 |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 当core的类型是多线程时，表项需要增加tag域，因此需要给出tag域的信息  Specific\_tag=1，代表使用tag，并且将会给出tag的宽度  Tag\_w= int(log2(coredynp.num\_hthreads)+ EXTRA\_TAG\_BITS)  //EXTRA\_TAG\_BITS为cacti中的参数，const int = 5 |
| Data(临时使用) | int(ceil(XML->sys.core[ithCore].predictor.local\_predictor\_size[0]/8.0))  向上取整，以字节为单位，计算每个表项的宽度 |
| interface\_ip.line\_sz | Data  单位字节，每个cache line的大小 |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].predictor.local\_predictor\_entries  单位字节，整个cache的大小，表项数\*每个表项的大小 |
| interface\_ip.nbanks | 1  Bank的个数，预测表都没有分bank |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast  fast(2): 数据和tag同时访问（具体见interface\_ip解析） |
| interface\_ip.  throughput | 1.0/clockRate  吞吐量 |
| interface\_ip.latency | 1.0/clockRate，延迟 |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0  读写端口数 |
| interface\_ip.  num\_rd\_ports | 单独的读端口数，=coredynp.predictionW  prediction\_width，core的参数，一般设置为1 |
| interface\_ip.  num\_wr\_ports | 单独的写端口数，=coredynp.predictionW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# BranchPredictor中的L2\_localBPT

|  |  |
| --- | --- |
| 参数 | 解释 |
| L2\_localBPT = new ArrayST(&interface\_ip, "L2 local Predictor", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.assoc  nterface\_ip.pure\_cam | 需要设置array的相连度和类型  Assoc=1，因为分支预测器中的预测表都是直接映射  Pure\_cam=false，因为预测表不是CAM类型的存储单元 |
| interface\_ip.is\_cache  interface\_ip.pure\_ram | 这两个参数需要根据core的类型进行设置  当core是多线程时（不是多线程的时候结果相反）  Is\_cache=true，因为此时需要为表项增加tag域，指代不同的线程，这种结构是类cache的结构  Pure\_ram=false，此时不是单纯的锁存器类型的结构 |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 当core的类型是多线程时，表项需要增加tag域，因此需要给出tag域的信息  Specific\_tag=1，代表使用tag，并且将会给出tag的宽度  Tag\_w= int(log2(coredynp.num\_hthreads)+ EXTRA\_TAG\_BITS)  //EXTRA\_TAG\_BITS为cacti中的参数，const int = 5 |
| Data(临时使用) | int(ceil(XML->sys.core[ithCore].predictor.local\_predictor\_size[1]/8.0))  向上取整，以字节为单位，计算每个表项的宽度 |
| interface\_ip.line\_sz | Data  单位字节，每个cache line的大小 |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].predictor.local\_predictor\_entries  单位字节，整个cache的大小，表项数\*每个表项的大小 |
| interface\_ip.nbanks | 1  Bank的个数，预测表都没有分bank |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast  fast(2): 数据和tag同时访问（具体见interface\_ip解析） |
| interface\_ip.  throughput | 1.0/clockRate  吞吐量 |
| interface\_ip.latency | 1.0/clockRate，延迟 |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0  读写端口数 |
| interface\_ip.  num\_rd\_ports | 单独的读端口数，=coredynp.predictionW  prediction\_width，core的参数，一般设置为1 |
| interface\_ip.  num\_wr\_ports | 单独的写端口数，=coredynp.predictionW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# BranchPredictor中的choose

|  |  |
| --- | --- |
| 参数 | 解释 |
| chooser = new ArrayST(&interface\_ip, "Predictor Chooser", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.assoc  nterface\_ip.pure\_cam | 需要设置array的相连度和类型  Assoc=1，因为分支预测器中的预测表都是直接映射  Pure\_cam=false，因为预测表不是CAM类型的存储单元 |
| interface\_ip.is\_cache  interface\_ip.pure\_ram | 这两个参数需要根据core的类型进行设置  当core是多线程时（不是多线程的时候结果相反）  Is\_cache=true，因为此时需要为表项增加tag域，指代不同的线程，这种结构是类cache的结构  Pure\_ram=false，此时不是单纯的锁存器类型的结构 |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 当core的类型是多线程时，表项需要增加tag域，因此需要给出tag域的信息  Specific\_tag=1，代表使用tag，并且将会给出tag的宽度  Tag\_w= int(log2(coredynp.num\_hthreads)+ EXTRA\_TAG\_BITS)  //EXTRA\_TAG\_BITS为cacti中的参数，const int = 5 |
| Data(临时使用) | int(ceil(XML->sys.core[ithCore].predictor. chooser\_predictor\_bits/8.0))  向上取整，以字节为单位，计算每个表项的宽度 |
| interface\_ip.line\_sz | Data  单位字节，每个cache line的大小 |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].predictor. chooser\_predictor\_entries  单位字节，整个cache的大小，表项数\*每个表项的大小 |
| interface\_ip.nbanks | 1  Bank的个数，预测表都没有分bank |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast  fast(2): 数据和tag同时访问（具体见interface\_ip解析） |
| interface\_ip.  throughput | 1.0/clockRate  吞吐量 |
| interface\_ip.latency | 1.0/clockRate，延迟 |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0  读写端口数 |
| interface\_ip.  num\_rd\_ports | 单独的读端口数，=coredynp.predictionW  prediction\_width，core的参数，一般设置为1 |
| interface\_ip.  num\_wr\_ports | 单独的写端口数，=coredynp.predictionW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# BranchPredictor中的RAS

|  |  |
| --- | --- |
| 参数 | 解释 |
| RAS = new ArrayST(&interface\_ip, "RAS", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.assoc  nterface\_ip.pure\_cam | 需要设置array的相连度和类型  Assoc=1，因为分支预测器中的预测表都是直接映射  Pure\_cam=false，因为预测表不是CAM类型的存储单元 |
| interface\_ip.is\_cache  interface\_ip.pure\_ram | False  True（不是类cache的结构，类寄存器/锁存器） |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 当core的类型是多线程时，表项需要增加tag域，因此需要给出tag域的信息  Specific\_tag=1，代表使用tag，并且将会给出tag的宽度  Tag\_w= int(log2(coredynp.num\_hthreads)+ EXTRA\_TAG\_BITS)  //EXTRA\_TAG\_BITS为cacti中的参数，const int = 5 |
| Data(临时使用) | int(ceil(coredynp.pc\_width/8.0))  向上取整，以字节为单位，计算每个表项的宽度 |
| interface\_ip.line\_sz | Data  单位字节，每个cache line的大小 |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore]. RAS\_size  单位字节，整个cache的大小，表项数\*每个表项的大小 |
| interface\_ip.nbanks | 1  Bank的个数，预测表都没有分bank |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast  fast(2): 数据和tag同时访问（具体见interface\_ip解析） |
| interface\_ip.  throughput | 1.0/clockRate  吞吐量 |
| interface\_ip.latency | 1.0/clockRate，延迟 |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0  读写端口数 |
| interface\_ip.  num\_rd\_ports | 单独的读端口数，=coredynp.predictionW  prediction\_width，core的参数，一般设置为1 |
| interface\_ip.  num\_wr\_ports | 单独的写端口数，=coredynp.predictionW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# InstFetchU中的BTB

|  |  |
| --- | --- |
| 参数 | 解释 |
| BTB = new ArrayST(&interface\_ip, "Branch Target Buffer", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.assoc | XML->sys.core[ithCore].BTB.BTB\_config[2] |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.virtual\_address\_width + int(ceil(log2(XML->sys.core[ithCore].number\_hardware\_threads))) +EXTRA\_TAG\_BITS |
| interface\_ip.line\_sz | XML->sys.core[ithCore].BTB.BTB\_config[1] |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].BTB.BTB\_config[0] |
| interface\_ip.nbanks | XML->sys.core[ithCore].BTB.BTB\_config[3] |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 0  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].BTB.BTB\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].BTB.BTB\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1  读写端口数 |
| interface\_ip.  num\_rd\_ports | coredynp.predictionW |
| interface\_ip.  num\_wr\_ports | coredynp.predictionW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# InstFetchU中的IB

|  |  |
| --- | --- |
| 参数 | 解释 |
| IB = new ArrayST(&interface\_ip, "InstBuffer", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | false  true  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.physical\_address\_width + EXTRA\_TAG\_BITS  (prefetch buffer遗留的设置) |
| Data | 临时变量  XML->sys.core[ithCore].instruction\_length\*  XML->sys.core[ithCore].peak\_issue\_width |
| interface\_ip.line\_sz | int(ceil(data/8.0)) |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].number\_hardware\_threads\* XML->sys.core[ithCore].instruction\_buffer\_size\*interface\_ip.line\_sz  （最大只能为64，如果大于64，强制设为64） |
| interface\_ip.assoc | 1 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 0  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1/clockRate |
| interface\_ip.latency | 1/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | XML->sys.core[ithCore].number\_instruction\_fetch\_ports;  读写端口数 |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# RegFU中的IRF

|  |  |
| --- | --- |
| 参数 | 解释 |
| IRF = new ArrayST(&interface\_ip, "Integer Register File", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | false  true  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 0 (在不是cache的情况下，这两个标志没有关系)  - |
| Data | 临时变量 coredynp.int\_data\_width |
| interface\_ip.line\_sz | int(ceil(data/32.0))\*4，（32位，4个字节） |
| interface\_ip.cache\_sz | coredynp.num\_IRF\_entry\*interface\_ip.line\_sz |
| interface\_ip.assoc | 1 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1/clockRate |
| interface\_ip.latency | 1/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于在异常发生时的保存和恢复使用 |
| interface\_ip.  num\_rd\_ports | 2\*coredynp.peak\_issueW  （默认每条指令每周期读两次） |
| interface\_ip.  num\_wr\_ports | coredynp.peak\_issueW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# RegFU中的FRF

|  |  |
| --- | --- |
| 参数 | 解释 |
| FRF = new ArrayST(&interface\_ip, "Floating point Register File", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | false  true  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 0 (在不是cache的情况下，这两个标志没有关系)  - |
| Data | 临时变量 coredynp.fp\_data\_width |
| interface\_ip.line\_sz | int(ceil(data/32.0))\*4，（32位，4个字节） |
| interface\_ip.cache\_sz | coredynp.num\_FRF\_entry\*interface\_ip.line\_sz |
| interface\_ip.assoc | 1 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1/clockRate |
| interface\_ip.latency | 1/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于在异常发生时的保存和恢复使用 |
| interface\_ip.  num\_rd\_ports | 2\*XML->sys.core[ithCore].issue\_width  （默认每条指令每周期读两次） |
| interface\_ip.  num\_wr\_ports | XML->sys.core[ithCore].issue\_width |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# RegFU中的RFWIN（应该是为了保存异常发生时的寄存器堆使用）

|  |  |
| --- | --- |
| 参数 | 解释 |
| 存在该对象的前提条件：按序处理器并且指定了寄存器窗口的大小  (XML->sys.core[ithCore].register\_windows\_size>0&&coredynp.core\_ty==Inorder) | |
| RFWIN = new ArrayST(&interface\_ip, "RegWindow", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | false  true  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 0 (在不是cache的情况下，这两个标志没有关系)  - |
| Data | 临时变量 coredynp.int\_data\_width |
| interface\_ip.line\_sz | int(ceil(data/8.0)) |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].register\_windows\_size\* IRF->l\_ip.cache\_sz\* XML->sys.core[ithCore].number\_hardware\_threads  （寄存器窗口大小\*整数寄存器大小\*线程数） |
| interface\_ip.assoc | 1 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 （单位bit，输出宽度，每个表项的宽度） |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 4.0/clockRate |
| interface\_ip.latency | 4.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于在异常发生时的保存和恢复使用 |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# SchedulerU中的int\_inst\_window（按序情况下）

|  |  |
| --- | --- |
| 参数 | 解释 |
| 指令发射队列，在按序多发射或者多线程中有这种结构 | |
| int\_inst\_window = new ArrayST(&interface\_ip, "InstFetchQueue", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  int(log2(XML->sys.core[ithCore].number\_hardware\_threads) \*coredynp.perThreadState)  （基于Niagara设计而提供的参数） |
| Data | XML->sys.core[ithCore].instruction\_length |
| interface\_ip.line\_sz | int(ceil(data/8.0)) |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].instruction\_window\_size \*interface\_ip.line\_sz  （最大只能为64B，如果超过强制设为64） |
| interface\_ip.assoc | 0（全相联） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1/clockRate |
| interface\_ip.latency | 1/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0 |
| interface\_ip.  num\_rd\_ports | coredynp.peak\_issueW |
| interface\_ip.  num\_wr\_ports | coredynp.peak\_issueW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# SchedulerU中的int\_inst\_window（乱序情况下）

|  |  |
| --- | --- |
| 参数 | 解释 |
| 乱序中，指令窗口为CAM类型  当指令队列的结构是基于物理寄存器的OOO时，只存储物理寄存器索引  当基于RS的OOO时，物理寄存器的tag和数据都要被保存  在指令可以被发射之前，会有两次读操作，一次写操作 | |
| int\_inst\_window = new ArrayST(&interface\_ip, tmp\_name, Core\_device, coredynp.opt\_local, coredynp.core\_ty);  tmp\_name=IntReservationStation/InstIssueQueue | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  coredynp.phy\_ireg\_width |
| Data | (PhysicalRegFile)  int((ceil((coredynp.instruction\_length+2\*(coredynp.phy\_ireg\_width - coredynp.arch\_ireg\_width))/2.0)/8.0))  (ReservationStation)  int(ceil(((coredynp.instruction\_length+2\*(coredynp.phy\_ireg\_width - coredynp.arch\_ireg\_width)+2\*coredynp.int\_data\_width)/2.0)/8.0))  //phy\_ireg\_width-arch\_ireg\_width索引位数之差直接映射？）  //\*2代表两个物理寄存器的索引  //+指令长度  //数据宽度除以2意味着只有当两个操作数可用时，整个数据才会被读出(model) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].instruction\_window\_size |
| interface\_ip.assoc | 0（全相联） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 0  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 2.0/clockRate |
| interface\_ip.latency | 2.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0 |
| interface\_ip.  num\_rd\_ports | coredynp.peak\_issueW |
| interface\_ip.  num\_wr\_ports | coredynp.peak\_issueW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# SchedulerU中的fp\_inst\_window（只在乱序中定义）

|  |  |
| --- | --- |
| 参数 | 解释 |
| fp\_inst\_window = new ArrayST(&interface\_ip, tmp\_name, Core\_device, coredynp.opt\_local, coredynp.core\_ty)  tmp\_name=FPReservationStation/FPIssueQueue | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  coredynp.phy\_freg\_width |
| Data | (PhysicalRegFile)  int(ceil((coredynp.instruction\_length+2\*(coredynp.phy\_freg\_width - coredynp.arch\_freg\_width))/8.0))  (ReservationStation)  int(ceil((coredynp.instruction\_length+2\*(coredynp.phy\_freg\_width - coredynp.arch\_freg\_width) + 2\*coredynp.fp\_data\_width)/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].instruction\_window\_size |
| interface\_ip.assoc | 0（全相联） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 0  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0 |
| interface\_ip.  num\_rd\_ports | Coredynp.fp\_issueW |
| interface\_ip.  num\_wr\_ports | Coredynp.fp\_issueW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# SchedulerU中的ROB (只在乱序中定义)

|  |  |
| --- | --- |
| 参数 | 解释 |
| 如果ROB的大小为0，则不创建ROB  此时将意味这乱序执行也会乱序提交，同时应该保证分支指令必须在之后的指令发射之前被解决  ROB包括了浮点指令和定点指令 | |
| ROB = new ArrayST(&interface\_ip, "ReorderBuffer", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | False  True  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | -  - |
| robExtra | 临时变量  int(ceil(5 + log2(coredynp.num\_hthreads)))  5bits：busy, Issued, Finished, speculative, valid |
| Data | (PhysicalRegFile)  int(ceil((robExtra+coredynp.pc\_width + coredynp.phy\_ireg\_width)/8.0))  (Reservation Station)  int(ceil((robExtra + coredynp.pc\_width + coredynp.phy\_ireg\_width + coredynp.fp\_data\_width)/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore]. ROB\_size  (XML中给出的ROB size是包括了所有线程的) |
| interface\_ip.assoc | 1（直接映射） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8  单位bit，输出宽度，每个表项的宽度 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0 |
| interface\_ip.  num\_rd\_ports | Coredynp. peak\_commitW |
| interface\_ip.  num\_wr\_ports | Coredynp. peak\_issueW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 0  搜索端口（一般应该是在CAM中使用） |

# RENAMING中的FRAT，RRAT，RAHT解释

* RAHT是检查点的硬件结构，用于减少分支预测错误所带来的影响（根据alpha 21264设计）
* 基于物理寄存器堆的设计中，物理寄存器文件中包括了推测的数据和非推测的数据。对于非推测式的数据必须保证在线程切换或者中断的时候能够被正确保存下来
* 因此Hinton提出了双RAT的设计方案，McPAT就是基于这样的设计，类似的结构有（Intel Netburst, Sandy Bridge）
* FRAT(frontend RAT) 用于正常情况下的重命名
* RRAT(retired RAT)则是存储着当前体系结构寄存器的映射。RRAT只会在提交阶段被存取，不会影响FRAT
* FRAT可以是CAM结构或者是RAM结构，但是RRAT只是RAM结构。并且RRAT的写端口数量和提交宽度相同

# RENAMINF中的iFRAT（PRF+RAMbased）

|  |  |
| --- | --- |
| 参数 | 解释 |
| iFRAT只在乱序中才会定义  调度策略的种类：PRF / RS  重命名策略：RAMbased / CAMbased | |
| iFRAT = new ArrayST(&interface\_ip, "Int FrontRAT", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | False  True  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | -  - |
| Data | 33 |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].archi\_Regs\_IRF\_size |
| interface\_ip.assoc | 1（直接映射） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | 8 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于和检查点硬件之间的传输（RAHT） |
| interface\_ip.  num\_rd\_ports | 2\*coredynp.decodeW |
| interface\_ip.  num\_wr\_ports | coredynp.decodeW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# RENAMINF中的iFRAT（PRF+CAMbased）

|  |  |
| --- | --- |
| 参数 | 解释 |
| iFRAT = new ArrayST(&interface\_ip, "Int FrontRAT", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  coredynp.arch\_ireg\_width |
| Data | int(ceil ((coredynp.arch\_ireg\_width+1\*coredynp.globalCheckpoint)/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].phy\_Regs\_IRF\_size |
| interface\_ip.assoc | 0（全相连） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | 8\*int(ceil (coredynp.arch\_ireg\_width/8.0)) |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于和检查点硬件之间的传输（RAHT） |
| interface\_ip.  num\_rd\_ports | coredynp.decodeW |
| interface\_ip.  num\_wr\_ports | coredynp.decodeW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 2\* coredynp.decodeW |

# RENAMINF中的iFRAT（RS+RAMbased）

|  |  |
| --- | --- |
| 参数 | 解释 |
| iFRAT = new ArrayST(&interface\_ip, "Int FrontRAT", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  coredynp.phy\_ireg\_width |
| Data | int(ceil(coredynp.phy\_ireg\_width/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].archi\_Regs\_IRF\_size  \*(1+coredynp.globalCheckpoint) |
| interface\_ip.assoc | 0（全相连） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | 8\* int(ceil(coredynp.phy\_ireg\_width/8.0)) |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于和检查点硬件之间的传输（RAHT） |
| interface\_ip.  num\_rd\_ports | 2\*coredynp.decodeW |
| interface\_ip.  num\_wr\_ports | coredynp.decodeW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | coredynp.commitW |

# RENAMINF中的iFRAT（RS+CAMbased）

|  |  |
| --- | --- |
| 参数 | 解释 |
| iFRAT = new ArrayST(&interface\_ip, "Int FrontRAT", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  coredynp.arch\_ireg\_width |
| Data | int(ceil (coredynp.arch\_ireg\_width+1\*coredynp.globalCheckpoint/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].phy\_Regs\_IRF\_size |
| interface\_ip.assoc | 0（全相连） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | 8\* int(ceil (coredynp.arch\_ireg\_width/8.0)) |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于和检查点硬件之间的传输（RAHT） |
| interface\_ip.  num\_rd\_ports | XML->sys.core[ithCore].decode\_width |
| interface\_ip.  num\_wr\_ports | XML->sys.core[ithCore].decode\_width |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 2\*XML->sys.core[ithCore].decode\_width |

# RENAMINF中的fFRAT（PRF+RAMbased）

|  |  |
| --- | --- |
| 参数 | 解释 |
| fFRAT = new ArrayST(&interface\_ip, "FP FrontRAT", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | False  True  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | -  - |
| Data | int(ceil(coredynp.phy\_freg\_width\* (1+coredynp.globalCheckpoint)/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].archi\_Regs\_FRF\_size |
| interface\_ip.assoc | 1（直接映射） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | 8\* int(ceil(coredynp.phy\_freg\_width/8.0)) |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于和检查点硬件之间的传输（RAHT） |
| interface\_ip.  num\_rd\_ports | 2\*coredynp.fp\_decodeW |
| interface\_ip.  num\_wr\_ports | coredynp.fp\_decodeW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# RENAMINF中的fFRAT（PRF+CAMbased）

|  |  |
| --- | --- |
| 参数 | 解释 |
| iFRAT = new ArrayST(&interface\_ip, "Int FrontRAT", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  coredynp.arch\_freg\_width |
| Data | int(ceil ((coredynp.arch\_freg\_width+1\*coredynp.globalCheckpoint)/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].phy\_Regs\_FRF\_size |
| interface\_ip.assoc | 0（全相连） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | 8\*int(ceil (coredynp.arch\_freg\_width/8.0)) |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于和检查点硬件之间的传输（RAHT） |
| interface\_ip.  num\_rd\_ports | coredynp.fp\_decodeW |
| interface\_ip.  num\_wr\_ports | coredynp.fp\_decodeW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 2\* coredynp.fp\_decodeW |

# RENAMINF中的iFRAT（RS+RAMbased）

|  |  |
| --- | --- |
| 参数 | 解释 |
| iFRAT = new ArrayST(&interface\_ip, "Int FrontRAT", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  coredynp.phy\_freg\_width |
| Data | int(ceil(coredynp.phy\_freg\_width/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].archi\_Regs\_FRF\_size  \*(1+coredynp.globalCheckpoint) |
| interface\_ip.assoc | 0（全相连） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | 8\* int(ceil(coredynp.phy\_freg\_width/8.0)) |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于和检查点硬件之间的传输（RAHT） |
| interface\_ip.  num\_rd\_ports | 2\*coredynp.fp\_decodeW |
| interface\_ip.  num\_wr\_ports | coredynp.fp\_decodeW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | coredynp.fp\_decodeW |

# RENAMINF中的iFRAT（RS+CAMbased）

|  |  |
| --- | --- |
| 参数 | 解释 |
| iFRAT = new ArrayST(&interface\_ip, "Int FrontRAT", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  coredynp.arch\_freg\_width |
| Data | int(ceil (coredynp.arch\_freg\_width+1\*coredynp.globalCheckpoint/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].phy\_Regs\_FRF\_size |
| interface\_ip.assoc | 0（全相连） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | 8\* int(ceil (coredynp.arch\_freg\_width/8.0)) |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1，用于和检查点硬件之间的传输（RAHT） |
| interface\_ip.  num\_rd\_ports | XML->sys.core[ithCore].decode\_width |
| interface\_ip.  num\_wr\_ports | coredynp.fp\_decodeW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 2\* coredynp.fp\_decodeW |

# RENAMINF中的iRRAT (PRF，没有RS)

|  |  |
| --- | --- |
| 参数 | 解释 |
| iRRAT = new ArrayST(&interface\_ip, "Int RetireRAT", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | False  True  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | -  - |
| Data | int(ceil(coredynp.phy\_ireg\_width/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].archi\_Regs\_IRF\_size\*2 |
| interface\_ip.assoc | 1（直接映射） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0 |
| interface\_ip.  num\_rd\_ports | XML->sys.core[ithCore].commit\_width |
| interface\_ip.  num\_wr\_ports | XML->sys.core[ithCore].commit\_width |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# RENAMINF中的fRRAT（PRF，没有RS）

|  |  |
| --- | --- |
| 参数 | 解释 |
| fRRAT = new ArrayST(&interface\_ip, "FP RetireRAT", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | False  True  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | -  - |
| Data | int(ceil(coredynp.phy\_freg\_width/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].archi\_Regs\_FRF\_size\*2 |
| interface\_ip.assoc | 1（直接映射） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0 |
| interface\_ip.  num\_rd\_ports | coredynp.fp\_decodeW |
| interface\_ip.  num\_wr\_ports | coredynp.fp\_decodeW |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# RENAMINF中的ifreeL(PRF，只有RAM类型的)

|  |  |
| --- | --- |
| 参数 | 解释 |
| ifreeL = new ArrayST(&interface\_ip, "Int Free List", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | False  True  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | -  - |
| Data | int(ceil(coredynp.phy\_ireg\_width/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*coredynp.num\_ifreelist\_entries |
| interface\_ip.assoc | 1（直接映射） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1 |
| interface\_ip.  num\_rd\_ports | coredynp.decodeW |
| interface\_ip.  num\_wr\_ports | coredynp.decodeW -1 + XML->sys.core[ithCore].commit\_width  every cycle, (coredynp.decodeW -1) inst may need to send back it dest tags, committW insts needs to update freelist buffers |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# RENAMINF中的ifreeL(RS, 这种情况下，freelist同时包括了int和fp)

|  |  |
| --- | --- |
| 参数 | 解释 |
| ifreeL = new ArrayST(&interface\_ip, "Unified Free List", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | False  True  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | -  - |
| Data | int(ceil(coredynp.phy\_ireg\_width/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*coredynp.num\_ifreelist\_entries |
| interface\_ip.assoc | 1（直接映射） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1 |
| interface\_ip.  num\_rd\_ports | XML->sys.core[ithCore].decode\_width |
| interface\_ip.  num\_wr\_ports | XML->sys.core[ithCore].decode\_width -1 + XML->sys.core[ithCore].commit\_width |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# RENAMINF中的ffreeL（PRF，在RS的情况下是统一的freelist）

|  |  |
| --- | --- |
| 参数 | 解释 |
| ffreeL = new ArrayST(&interface\_ip, "FP Free List", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | False  True  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | -  - |
| Data | int(ceil(coredynp.phy\_freg\_width/8.0)) |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*coredynp.num\_ffreelist\_entries |
| interface\_ip.assoc | 1（直接映射） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 1 |
| interface\_ip.  num\_rd\_ports | coredynp.fp\_decodeW |
| interface\_ip.  num\_wr\_ports | coredynp.fp\_decodeW -1 + XML->sys.core[ithCore].commit\_width |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |

# LoadStoreU 中的LSQ

|  |  |
| --- | --- |
| 参数 | 解释 |
| LSQ在按序处理器中同时提供LQ和SQ的功能，在乱序处理器中只是SQ | |
| LSQ = new ArrayST(&interface\_ip, "Load(Store)Queue", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.core[ithCore].opcode\_width+ XML->sys.virtual\_address\_width + int(ceil(log2(XML->sys.core[ithCore].number\_hardware\_threads))) + EXTRA\_TAG\_BITS |
| Data | XML->sys.machine\_bits |
| interface\_ip.line\_sz | int(ceil(data/32.0))\*4 |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].store\_buffer\_size\*interface\_ip.line\_sz\* XML->sys.core[ithCore].number\_hardware\_threads |
| interface\_ip.assoc | 0（全相联） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0 |
| interface\_ip.  num\_rd\_ports | XML->sys.core[ithCore].memory\_ports |
| interface\_ip.  num\_wr\_ports | XML->sys.core[ithCore].memory\_ports |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | XML->sys.core[ithCore].memory\_ports |

# LoadStoreU 中的LoadQ（仅在乱序处理器中定义）

|  |  |
| --- | --- |
| 参数 | 解释 |
| (coredynp.core\_ty==OOO) && (XML->sys.core[ithCore].load\_buffer\_size >0) | |
| LoadQ = new ArrayST(&interface\_ip, "LoadQueue", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.core[ithCore].opcode\_width+ XML->sys.virtual\_address\_width + int(ceil(log2(XML->sys.core[ithCore].number\_hardware\_threads))) + EXTRA\_TAG\_BITS |
| Data | XML->sys.machine\_bits |
| interface\_ip.line\_sz | int(ceil(data/32.0))\*4 |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].load\_buffer\_size\*interface\_ip.line\_sz\* XML->sys.core[ithCore].number\_hardware\_threads |
| interface\_ip.assoc | 0（全相联） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | 1.0/clockRate |
| interface\_ip.latency | 1.0/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0 |
| interface\_ip.  num\_rd\_ports | XML->sys.core[ithCore].memory\_ports |
| interface\_ip.  num\_wr\_ports | XML->sys.core[ithCore].memory\_ports |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | XML->sys.core[ithCore].memory\_ports |

# MemManU中的itlb

|  |  |
| --- | --- |
| 参数 | 解释 |
| itlb = new ArrayST(&interface\_ip, "ITLB", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.virtual\_address\_width- int(floor(log2(XML->sys.virtual\_memory\_page\_size))) + int(ceil(log2(XML->sys.core[ithCore].number\_hardware\_threads)))+ EXTRA\_TAG\_BITS |
| Data | XML->sys.physical\_address\_width- int(floor(log2(XML->sys.virtual\_memory\_page\_size))) |
| interface\_ip.line\_sz | int(ceil(data/8.0)) |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].itlb.number\_entries\*interface\_ip.line\_sz |
| interface\_ip.assoc | 0（全相联） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].icache.icache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].icache.icache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0 |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | XML->sys.core[ithCore].number\_instruction\_fetch\_ports |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | XML->sys.core[ithCore].number\_instruction\_fetch\_ports |

# MemManU中的dtlb

|  |  |
| --- | --- |
| 参数 | 解释 |
| dtlb = new ArrayST(&interface\_ip, "DTLB", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.virtual\_address\_width- int(floor(log2(XML->sys.virtual\_memory\_page\_size))) + int(ceil(log2(XML->sys.core[ithCore].number\_hardware\_threads)))+ EXTRA\_TAG\_BITS |
| Data | XML->sys.physical\_address\_width- int(floor(log2(XML->sys.virtual\_memory\_page\_size))) |
| interface\_ip.line\_sz | int(ceil(data/8.0)) |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].dtlb.number\_entries\*interface\_ip.line\_sz |
| interface\_ip.assoc | 0（全相联） |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 1  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].dcache.dcache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].dcache.dcache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到 |
| interface\_ip.  num\_rw\_ports | 0 |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | XML->sys.core[ithCore].memory\_ports |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | XML->sys.core[ithCore].memory\_ports |

# InstCache中的caches

|  |  |
| --- | --- |
| 参数 | 解释 |
| icache.caches = new ArrayST(&interface\_ip, "icache", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  (int)XML->sys.physical\_address\_width-idx-int(ceil(log2(line))) + EXTRA\_TAG\_BITS |
| Data |  |
| interface\_ip.line\_sz | (int)XML->sys.core[ithCore].icache.icache\_config[1] |
| interface\_ip.cache\_sz | (int)XML->sys.core[ithCore].icache.icache\_config[0] |
| interface\_ip.assoc | (int)XML->sys.core[ithCore].icache.icache\_config[2] |
| interface\_ip.nbanks | (int)XML->sys.core[ithCore].icache.icache\_config[3] |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 0  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].icache.icache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].icache.icache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到  (未设置) |
| interface\_ip.  num\_rw\_ports | XML->sys.core[ithCore].number\_instruction\_fetch\_ports |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 未设置 |

# InstCache中的ifb

|  |  |
| --- | --- |
| 参数 | 解释 |
| icache.ifb = new ArrayST(&interface\_ip, "icacheFillBuffer", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.physical\_address\_width + EXTRA\_TAG\_BITS |
| Data | icache.caches->l\_ip.line\_sz |
| interface\_ip.line\_sz | data |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].icache.buffer\_sizes[1] |
| interface\_ip.assoc | 0 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 0  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].icache.icache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].icache.icache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到  (未设置) |
| interface\_ip.  num\_rw\_ports | XML->sys.core[ithCore].number\_instruction\_fetch\_ports |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | XML->sys.core[ithCore].number\_instruction\_fetch\_ports |

# InstCache中的missb

|  |  |
| --- | --- |
| 参数 | 解释 |
| icache.missb = new ArrayST(&interface\_ip, "icacheMissBuffer", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.physical\_address\_width + EXTRA\_TAG\_BITS |
| Data | (XML->sys.physical\_address\_width) + int(ceil(log2(size/line))) + icache.caches->l\_ip.line\_sz\*8 |
| interface\_ip.line\_sz | int(ceil(data/8.0)) |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].icache.buffer\_sizes[0]\*interface\_ip.line\_sz |
| interface\_ip.assoc | 0 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 0  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].icache.icache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].icache.icache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到  (未设置) |
| interface\_ip.  num\_rw\_ports | XML->sys.core[ithCore].number\_instruction\_fetch\_ports |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | XML->sys.core[ithCore].number\_instruction\_fetch\_ports |

# InstCache中的prefetchb

|  |  |
| --- | --- |
| 参数 | 解释 |
| icache.prefetchb = new ArrayST(&interface\_ip, "icacheprefetchBuffer", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.physical\_address\_width + EXTRA\_TAG\_BITS |
| Data | icache.caches->l\_ip.line\_sz |
| interface\_ip.line\_sz | Data |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].icache.buffer\_sizes[2]\*interface\_ip.line\_sz |
| interface\_ip.assoc | 0 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 0  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].icache.icache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].icache.icache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到  (未设置) |
| interface\_ip.  num\_rw\_ports | XML->sys.core[ithCore].number\_instruction\_fetch\_ports |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | XML->sys.core[ithCore].number\_instruction\_fetch\_ports |

# DataCache中的caches

|  |  |
| --- | --- |
| 参数 | 解释 |
| dcache.caches = new ArrayST(&interface\_ip, "dcache", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.physical\_address\_width-idx-int(ceil(log2(line))) + EXTRA\_TAG\_BITS |
| Data |  |
| interface\_ip.line\_sz | (int)XML->sys.core[ithCore].dcache.dcache\_config[1] |
| interface\_ip.cache\_sz | (int)XML->sys.core[ithCore].dcache.dcache\_config[0] |
| interface\_ip.assoc | (int)XML->sys.core[ithCore].dcache.dcache\_config[2] |
| interface\_ip.nbanks | (int)XML->sys.core[ithCore].dcache.dcache\_config[3] |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].dcache.dcache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].dcache.dcache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到  (未设置) |
| interface\_ip.  num\_rw\_ports | XML->sys.core[ithCore].memory\_ports  (通常情况下，按序1个，乱序2个至少) |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 未设置 |

# DataCache中的ifb

|  |  |
| --- | --- |
| 参数 | 解释 |
| dcache.ifb = new ArrayST(&interface\_ip, "dcacheFillBuffer", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.physical\_address\_width + EXTRA\_TAG\_BITS |
| Data |  |
| interface\_ip.line\_sz | dcache.caches->l\_ip.line\_sz |
| interface\_ip.cache\_sz | data\*XML->sys.core[ithCore].dcache.buffer\_sizes[1] |
| interface\_ip.assoc | 0 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].dcache.dcache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].dcache.dcache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到  (未设置) |
| interface\_ip.  num\_rw\_ports | XML->sys.core[ithCore].memory\_ports  (通常情况下，按序1个，乱序2个至少) |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 未设置 |

# DataCache中的missb

|  |  |
| --- | --- |
| 参数 | 解释 |
| dcache.missb = new ArrayST(&interface\_ip, "dcacheMissBuffer", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.physical\_address\_width + EXTRA\_TAG\_BITS |
| Data | (XML->sys.physical\_address\_width) + int(ceil(log2(size/line))) + dcache.caches->l\_ip.line\_sz\*8 |
| interface\_ip.line\_sz | int(ceil(data/8.0)) |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].dcache.buffer\_sizes[0]\*interface\_ip.line\_sz |
| interface\_ip.assoc | 0 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].dcache.dcache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].dcache.dcache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到  (未设置) |
| interface\_ip.  num\_rw\_ports | XML->sys.core[ithCore].memory\_ports  (通常情况下，按序1个，乱序2个至少) |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 未设置 |

# DataCache中的prefetchb

|  |  |
| --- | --- |
| 参数 | 解释 |
| dcache.prefetchb = new ArrayST(&interface\_ip, "dcacheprefetchBuffer", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.physical\_address\_width + EXTRA\_TAG\_BITS |
| Data | dcache.caches->l\_ip.line\_sz |
| interface\_ip.line\_sz | Data |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].dcache.buffer\_sizes[2]\*interface\_ip.line\_sz |
| interface\_ip.assoc | 0 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].dcache.dcache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].dcache.dcache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到  (未设置) |
| interface\_ip.  num\_rw\_ports | XML->sys.core[ithCore].memory\_ports  (通常情况下，按序1个，乱序2个至少) |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 输出结果为2 |

# DataCache中的wbb（当cache采用Write\_back策略时定义）

|  |  |
| --- | --- |
| 参数 | 解释 |
| dcache.prefetchb = new ArrayST(&interface\_ip, "dcacheprefetchBuffer", Core\_device, coredynp.opt\_local, coredynp.core\_ty) | |
| interface\_ip.is\_cache  interface\_ip.pure\_ram  interface\_ip.pure\_cam | True  False  False |
| interface\_ip.specific\_tag  interface\_ip.tag\_w | 1  XML->sys.physical\_address\_width + EXTRA\_TAG\_BITS |
| Data | dcache.caches->l\_ip.line\_sz |
| interface\_ip.line\_sz | Data |
| interface\_ip.cache\_sz | XML->sys.core[ithCore].dcache.buffer\_sizes[3]\*interface\_ip.line\_sz |
| interface\_ip.assoc | 0 |
| interface\_ip.nbanks | 1 |
| interface\_ip.out\_w | interface\_ip.line\_sz\*8 |
| interface\_ip.  access\_mode | 2  访问模式，0 normal, 1 seq, 2 fast |
| interface\_ip.  throughput | XML->sys.core[ithCore].dcache.dcache\_config[4]/clockRate |
| interface\_ip.latency | XML->sys.core[ithCore].dcache.dcache\_config[5]/clockRate |
| interface\_ip.  obj\_func\_dyn\_energy  obj\_func\_dyn\_power  obj\_func\_leak\_power  obj\_func\_cycle\_t | 0, 0, 0, 1  这些参数在cacti中未使用到  (未设置) |
| interface\_ip.  num\_rw\_ports | XML->sys.core[ithCore].memory\_ports  (通常情况下，按序1个，乱序2个至少) |
| interface\_ip.  num\_rd\_ports | 0 |
| interface\_ip.  num\_wr\_ports | 0 |
| interface\_ip.  num\_se\_rd\_ports | 0  单端读端口 |
| interface\_ip.  num\_search\_ports | 输出结果为2 |