目录

[1. Core类介绍 2](#_Toc532933781)

[2. Core类声明 5](#_Toc532933782)

[3. Core的构造函数 7](#_Toc532933783)

[4. ComputerEnergy函数 9](#_Toc532933784)

[5. displayEnergy函数 10](#_Toc532933785)

# Core类介绍

* Core是McPAT的Processor类中主要的用于实现核内具体逻辑的类
* Core中以阶段的形式进行划分，包括：取指阶段(instfetch)，重命名阶段(RENAMING)，执行阶段(EXECU)，存储管理阶段(MemManU)，访存阶段(LoadStoreU)。同时也包括了一些默认的对象(Pipeline, UndiffCore)和一些可能的对象(L2cache)
* Core中定义的对象

|  |  |  |  |
| --- | --- | --- | --- |
| Class | 内部定义的对象 | | |
| Core | ifu (InstFetchU)  exu(EXECU)  undiffCore(UndiffCore) | lsu(LoadStoreU)  rnu(RENAMINGU)  l2cache(SharedCache) | mmu(MemManU)  corepipe(Pipeline) |
| InstFetchU | cache\_p(Cache\_policy)  BTB(ArrayST)  ID\_operand(inst\_decoder) | icache(InstCache)  BPT(BranchPredictor)  ID\_misc(inst\_decoder) | IB(ArrayST)  ID\_inst(inst\_decoder) |
| BranchPredictor | globalBPT(ArrayST)  L1\_localBPT(ArrayST) | loaclBPT(ArrayST)  L2\_localBPT(ArrayST) | choose(ArrayST) RAS(ArrayST) |
| LoadStoreU | dcache(DataCache) | LSQ(ArrayST) | LoadQ(ArrayST) |
| MemManU | itlb(ArrayST) | dtlb(ArrayST) |  |
| EXECU | rfu(RegFU)  fp\_u(FuctionalUnit)  int\_bypass  (interconnect)  intTagBypass  (interconnect) | scheu(SchedulerU)  exeu(FuctionalUnit)  int\_mul\_bypass  (interconnect)  intTag\_mul\_Bypass  (interconnect) | mul(FuctionalUnit)  fp\_bypass  (interconnect)  fpTagBypass  (interconnect) |
| RegFU | IRF(ArrayST) | FRF(ArrayST) | RFWIN(ArrayST) |
| SchedulerU | int\_inst\_window  (ArrayST)  instruction\_selection  (selection\_logic) | fp\_inst\_window  (ArrayST) | ROB  (ArrayST) |
| RENAMING | iFRAT(ArrayST)  fFRAT(ArrayST)  idcl(dep\_resource  \_conflict\_check) | iRRAT(ArrayST)  FRRAT(ArrayST)  fdcl(dep\_resource  \_conflict\_check) | ifreeL(ArrayST)  ffreeL(ArrayST) |
| 未在core.h中定义的类 | | | |
| FuctionalUnit | logic.h 189 | local\_result(uca\_org\_t)  l\_ip(InputParameter)  ithCore(int)  FU\_height,clockRate  executionTime(double)  num\_fu(double)  energy, base\_energy,  per\_access\_energy, leakage, gate\_leakage  (double) | coredynp  (CoreDynParam)  tdp\_stats(statsDef)  rtp\_stats(statsDef)  stats\_t(statsDef)  power\_t(powerDef)  fu\_type(FU\_type) |
| inst\_decoder | logic.h 95 | local\_result(uca\_org\_t)  l\_ip(InputParameter)  opcode\_length(int)  num\_decoders(int)  num\_decoder\_segments  (int)  final\_dec(Decoder)  pre\_dec(Predec) | num\_decoded\_signals  (int)  device\_ty(Device\_ty)  core\_ty(Core\_type)  x86(bool)  tdp\_stats(statsDef)  rtp\_stats(statsDef)  stats\_t(statsDef)  power\_t(powerDef) |
| dep\_resource  \_conflict\_check | logic.h 72 | local\_result(uca\_org\_t)  WNORn, WNORp, Wevalinvp, Wevalinvn, Wcompn, Wcompp, Wcomppreequ(double)  compare\_bits(int) | l\_ip(InputParameter)  coredynp  (CoreDynParam)  tdp\_stats(statsDef)  rtp\_stats(statsDef)  stats\_t(statsDef)  power\_t(powerDef) |
| selection\_logic | logic.h 52 | local\_result(uca\_org\_t)  win\_entries(int)  issue\_width(int)  num\_threads(int) | device\_ty(Device\_ty)  core\_ty(Core\_type)  l\_ip(InputParameter) |
| interconnect | interconnect.h 48 |  |  |
| FU\_type | basic\_components.h 41 | enum FPU, ALU, MUL |  |
| Renaming\_type | basic\_components.h 47 | enum RAMbased | CAMbased |
| Scheduler\_type | basic\_components.h 52 | enum PhysicalRegFile | ReservationStation |
| cache\_level | basic\_components.h 57 | enum L2, L3 | L1Directory  L2Directory |
| Cache\_policy | basic\_components.h 77 | enum Write\_through,0 | Write\_back,1 |
| Device\_ty | basic\_components.h 82 | enum Core\_device  LLC\_device | Uncore\_device |
| Core\_type | basic\_components.h 88 | Enum OOO, InOrder |  |
| statsComponents | basic\_components.h 93 | Access, hit, miss  (double) |  |
| InstCache | array.h 70 | caches(ArrayST)  missb(ArrayST)  power\_t(powerDef) | Ifb(ArrayST)  prefetchb(ArrayST) |
| DataCache | array.h 90 | wbb(ArrayST) | 继承于InstCache |
| SharedCache | sharedcache.h 42 | l\_ip(InputParameter)  ithCore(int)  unicache(DataCache)  cachep  (CacheDynParam) | homenode\_tdp\_stats  homenode\_rtp\_stats  homenode\_stats\_t  (statsDef)  dir\_overhead(double)  executionTime  scktRatio(double) |
| Pipeline | logic.h 151 | local\_result(uca\_org\_t)  l\_ip(InputParameter)  is\_core\_pipeline, is\_default  process\_ind(bool)  load\_per\_pipeline\_stage  (double) | coredynp  (CoreDynParam)  device\_ty(Device\_ty)  num\_piperegs  WNANDn, WNANDp(double) |
| UndiffCore | logic.h 213 | local\_result(uca\_org\_t)  l\_ip(InputParameter)  ithCore(int)  clockRate  executionTime(double)  core\_ty(Core\_type)  opt\_performance, embedded(bool) | scktRatio, chip\_PR\_overhead, macro\_PR\_overhead  pipeline\_stage,  num\_hthreads,  issue\_width  (double) |
| ArrayST | array.h 45 | l\_ip(InputParameter)  local\_result(uca\_org\_t)  name(string)  opt\_local(bool)  power\_t(powerDef) | device\_ty(Device\_ty)  core\_ty(Core\_type)  tdp\_stats(statsDef)  rtp\_stats(statsDef)  stats\_t(statsDef) |

# Core类声明

|  |
| --- |
| class Core **:**public Component **{**  public**:**  //用于获取解析得到的参数  ParseXML **\***XML**;**  //第几个核，标号  int ithCore**;**  //用于和cacti通信的接口  InputParameter interface\_ip**;**  //基础参数：时钟频率，运行时间  double clockRate**,**executionTime**;**  //通过cacti中的参数赋值：cacti/parameter.cc:TechnologyParameter g\_tp  //暂时不知道什么含义，没有在core.cc文件中使用  double scktRatio**,** chip\_PR\_overhead**,** macro\_PR\_overhead**;**    //定义一个核中的组成部分：类似于流水线的定义  //取指单元  InstFetchU **\*** ifu**;**  //LSQ单元  LoadStoreU **\*** lsu**;**  //存储管理单元，包括TLB等  MemManU **\*** mmu**;**  //执行单元，FU  EXECU **\*** exu**;**  //重命名单元，RAT等  RENAMINGU **\*** rnu**;**    //流水线,logic.h 151  Pipeline **\*** corepipe**;**    //logic.h 213  UndiffCore **\*** undiffCore**;**    //sharedcache.h 42  //私有的L2 cache，虽然类名是share  //共享的l2 cache在processor中计算  SharedCache **\*** l2cache**;**  //core的动态参数  CoreDynParam coredynp**;**      //构造函数,设置参数，初始化对象，计算面积，计算功率  Core**(**ParseXML **\***XML\_interface**,** int ithCore\_**,** InputParameter**\*** interface\_ip\_**);**  //设置一些参数  void set\_core\_param**();**  //计算功耗  void computeEnergy**(**bool is\_tdp**=true);**  //显示计算的结果  void displayEnergy**(**uint32\_t indent **=** 0**,**int plevel **=** 100**,** bool is\_tdp**=true);**  **~**Core**();**  **};** |

# Core的构造函数

* 主要工作：定义需要的对象，计算每个对象的面积和整体的面积
* 在使用流水线对象占有的面积时，没有将其单独作为一部分显示出来，而是将其分摊到各个部件中，即每个部件都会增加一部分流水线带来的面积开销，功耗也是如此

|  |
| --- |
| Core**::**Core**(**ParseXML**\*** XML\_interface**,** int ithCore\_**,** InputParameter**\*** interface\_ip\_**)**  **:**XML**(**XML\_interface**),**ithCore**(**ithCore\_**),**  interface\_ip**(\***interface\_ip\_**),**  ifu **(**0**),**lsu **(**0**),**mmu **(**0**),**  exu **(**0**),**rnu **(**0**),**corepipe **(**0**),**  undiffCore **(**0**)** **,**l2cache **(**0**)**  **{**  bool exit\_flag **=** **true;**  //临时参数，用于就算流水线平摊到功能单元的面积  double pipeline\_area\_per\_unit**;**  //设置Core的一些基本参数，从xml文件中获取信息  set\_core\_param**();**  //如果有私有的L2cache，将创建对象，在Core中计算  **if** **(**XML**->**sys**.**Private\_L2**)**  **{**  l2cache **=** **new** SharedCache**(**XML**,**ithCore**,** **&**interface\_ip**);**  **}**  //创建Core的各个阶段的对象和undifferentiated core(未分化的core)  ifu **=** **new** InstFetchU**(**XML**,** ithCore**,** **&**interface\_ip**,**coredynp**,**exit\_flag**);**  lsu **=** **new** LoadStoreU**(**XML**,** ithCore**,** **&**interface\_ip**,**coredynp**,**exit\_flag**);**  mmu **=** **new** MemManU **(**XML**,** ithCore**,** **&**interface\_ip**,**coredynp**,**exit\_flag**);**  exu **=** **new** EXECU**(**XML**,** ithCore**,** **&**interface\_ip**,**lsu**->**lsq\_height**,** coredynp**,**  exit\_flag**);**  undiffCore**=** **new** UndiffCore**(**XML**,** ithCore**,** **&**interface\_ip**,**coredynp**,**exit\_flag**);**  //如果是乱序核，创建重命名逻辑的对象  **if** **(**coredynp**.**core\_ty**==**OOO**)**  **{**  rnu **=** **new** RENAMINGU**(**XML**,** ithCore**,** **&**interface\_ip**,**coredynp**);**  **}**  //创建流水线对象，主要计算每个流水级需要保存的数据，因此产生的开销  corepipe **=** **new** Pipeline**(&**interface\_ip**,**coredynp**);**  //将流水线带来的面积，平均分散到每个部件中，最终不会单独显示流水级的面积  //pipeline\_area\_per\_unit  **if** **(**coredynp**.**core\_ty**==**OOO**)**  **{**  //OOO中给定了5个unit，ifu, lsu, mmu, exu, rnu  //将流水线的面积平均到每个单元  pipeline\_area\_per\_unit **=** **(**corepipe**->**area**.**get\_area**()\***  coredynp**.**num\_pipelines**)/**5.0**;**  **if** **(**rnu**->**exist**)**  rnu**->**area**.**set\_area**(**rnu**->**area**.**get\_area**()** **+** pipeline\_area\_per\_unit**);**  **}**  **else** //inorder中不包括rnu  **{**  pipeline\_area\_per\_unit **=** **(**corepipe**->**area**.**get\_area**()\***  coredynp**.**num\_pipelines**)/**4.0**;**  **}**  //lsu,exu, mmu,undiffcore, private\_l2,rnu  **if** **(**ifu**->**exist**)**  **{**  ifu**->**area**.**set\_area**(**ifu**->**area**.**get\_area**()** **+** pipeline\_area\_per\_unit**);**  area**.**set\_area**(**area**.**get\_area**()** **+** ifu**->**area**.**get\_area**());**  **}**  **}** |

# ComputerEnergy函数

* ComputerEnergy函数包括一个参数is\_tdp，bool类型
* 当is\_tdp为true时，power = energy\_per\_cycle\* clock\_rate。在该函数中只计算得到每个周期该组件会消耗的能量energy\_per\_cycle，在displayEnergy函数中，将会使用该公式计算得到峰值power，即每个周期都在工作时的功率。此时的计算结果保存在power中
* 当is\_tdp为false时，power = total energy / Total execution time。同样该函数中只计算该组件在整个执行过程中会消耗的所有能量（使用组件的访问次数等计算），在displayEnergy函数中，将其除以整体的执行时间（cycle count / clock rate），得到运行时的动态功耗。此时的计算结果保存在rt\_power中
* 在计算power的时候，会将流水线的每周期的能耗分摊到每个组件中（但不是平均的）。在计算rnu的时候没有考虑到rnu的占空比，为了更精确可以添加进去。corepipe->power代表的是每个周期流水线的能耗参数

|  |
| --- |
| //平均的结果=流水线的参数\*流水线个数/单元数  //对于每周期的能耗开销，需要考虑到当前单元在一个周期内的占空比  //即分摊只在当前单元工作的时候会分摊能耗  double t**=**coredynp**.**num\_pipelines**/**num\_units**;**  pppm\_t**={**t**\***coredynp**.**IFU\_duty\_cycle**,**t**,**t**,**t**}**  ifu**->**power **=** ifu**->**power **+** corepipe**->**power**\***pppm\_t**;** |

* 在计算rt\_power时，也会将流水线整体功耗分摊。此时考虑了多核的情况，但是之前却没有

|  |
| --- |
| //t=coredynp.pipeline\_duty\_cycle\*XML->sys.total\_cycles  //计算单个流水线在整体执行过程中会执行的时钟周期数  //t\*coredynp.IFU\_duty\_cycle  //计算在IFU工作的时间内流水线的工作时长  double t**=**coredynp**.**pipeline\_duty\_cycle**\***XML**->**sys**.**total\_cycles**;**  double t1**=**coredynp**.**num\_pipelines**/**num\_units**;**  **if** **(**XML**->**sys**.**homogeneous\_cores**==**1**)**  //如果是同构多核的情况，需要将核数考虑进去  rtp\_pipeline\_coe **=** t**\***coredynp**.**IFU\_duty\_cycle**\***XML**->**sys**.**number\_of\_cores**;**  **else**  rtp\_pipeline\_coe **=** t**\***coredynp**.**IFU\_duty\_cycle**;**  pppm\_t**=** **{**t1**\***rtp\_pipeline\_coe**,** t1**,** t1**,** t1**};**  //corepipe->power代表着每周期流水线的消耗的能耗  //dynamic \*(t1\*rtp\_pipeline\_coe)  //代表在整个运行时间中，流水线会消耗的能耗/单元数  ifu**->**rt\_power **=** ifu**->**rt\_power **+** corepipe**->**power**\***pppm\_t**;** |

# displayEnergy函数

* 显示Core的功耗和面积等参数，同时进一步显示更细节的组件的信息

|  |
| --- |
| void Core**::**displayEnergy**(**uint32\_t indent**,**int plevel**,**bool is\_tdp**)**  **{**  **if** **(**is\_tdp**)**  **{**  /\*  Core:  Area = area.get\_area()\*1e-6  Peak Dynamic = power.readOp.dynamic\*clockRate  Subthreshold Leakage = (long\_channel?  power.readOp.longer\_channel\_leakage:power.readOp.leakage)  Subthreshold Leakage with power gating = (long\_channel?  power.readOp.power\_gated\_with\_long\_channel\_leakage  : power.readOp.power\_gated\_leakage)  Gate Leakage = power.readOp.gate\_leakage  Runtime Dynamic = rt\_power.readOp.dynamic/executionTime  \*/  **if** **(**ifu**->**exist**)**  **{**  //Instruction Fetch Unit:ifu  **if** **(**plevel **>**2**){**  ifu**->**displayEnergy**(**indent**+**4**,**plevel**,**is\_tdp**);**  **}**  **}**  **if** **(**coredynp**.**core\_ty**==**OOO**)**  **{**  **if** **(**rnu**->**exist**)**  **{**  //Renaming Unit:rnu  **if** **(**plevel **>**2**){**  rnu**->**displayEnergy**(**indent**+**4**,**plevel**,**is\_tdp**);**  **}**  **}**  **}**  **if** **(**lsu**->**exist**)**  **{**  //Load Store Unit:lsu  **if** **(**plevel **>**2**){**  lsu**->**displayEnergy**(**indent**+**4**,**plevel**,**is\_tdp**);**  **}**  **}**  **if** **(**mmu**->**exist**)**  **{**  //Memory Management Unit:mmu  **if** **(**plevel **>**2**){**  mmu**->**displayEnergy**(**indent**+**4**,**plevel**,**is\_tdp**);**  **}**  **}**  **if** **(**exu**->**exist**)**  **{**  //Execution Unit:exu  **if** **(**plevel **>**2**){**  exu**->**displayEnergy**(**indent**+**4**,**plevel**,**is\_tdp**);**  **}**  **}**  **if** **(**XML**->**sys**.**Private\_L2**)**  **{**  l2cache**->**displayEnergy**(**4**,**is\_tdp**);**  **}**  **}**  **else**  **{**  **}**  **}** |