# FunctionalUnit类声明

|  |
| --- |
| //核中的功能部件，目前是三个：ALU，FPU，MUL/DIV  class FunctionalUnit **:**public Component**{**  public**:**  //获取xml文件的内容  //sys.Embedded,根据处理器的应用场景不同，设备功耗有差别  //用于计算动态功耗使用  //sys.core[ithCore].mul\_accesses  //sys.core[ithCore].ialu\_accesses  //sys.core[ithCore].fpu\_accesses  ParseXML **\***XML**;**  //配合XML使用  int ithCore**;**  //用于在cacti和mcpat之间传递参数  //用到了interface\_ip.F\_sz\_um  InputParameter interface\_ip**;**  //core的参数信息  //clockRate，executionTime  //num\_fpus，num\_alus，num\_muls  //core\_ty， MUL\_duty\_cycle  //FPU\_duty\_cycle，ALU\_duty\_cycle  CoreDynParam coredynp**;**    //  double FU\_height**;**  //计算动态功耗使用，时钟频率和执行事件  double clockRate**,**executionTime**;**  //功能单元的个数  double num\_fu**;**  //中间参数  double energy**,** base\_energy**,**per\_access\_energy**,** leakage**,** gate\_leakage**;**  //  bool is\_default**;**  //功能单元的类型  //FU\_type: basic\_components.h:41  //ALU,FPU,MUL  enum FU\_type fu\_type**;**  //记录功耗的参数  statsDef tdp\_stats**;**  statsDef rtp\_stats**;**  statsDef stats\_t**;**  powerDef power\_t**;**    //构造函数  FunctionalUnit**(**ParseXML **\***XML\_interface**,** int ithCore\_**,**  InputParameter**\*** interface\_ip\_**,**const CoreDynParam **&** dyn\_p\_**,**  enum FU\_type fu\_type**);**  //计算功耗，在EXEU的computerEnergy中被调用  void computeEnergy**(**bool is\_tdp**=true);**  //显示结果，功耗和面积，在EXECU的displayEnergy被调用  void displayEnergy**(**uint32\_t indent **=** 0**,**int plevel **=** 100**,** bool is\_tdp**=true);**  //似乎为使用  void leakage\_feedback**(**double temperature**);**  **};** |

# 定义FunctionalUnit的对象

* 该类定义了处理器中的执行部件，目前支持三种结构：ALU，FPU，MUL
* 该类在Core的EXEU类中定义，有三个对象：fp\_u，exeu，mul
* Exeu即为ALU，core一定会定义该对象；fp\_u和mul需要根据xml指定的参数确定是否需要定义

|  |
| --- |
| exeu **=** **new** FunctionalUnit**(**XML**,** ithCore**,&**interface\_ip**,** coredynp**,** ALU**);**  **if** **(**coredynp**.**num\_fpus **>**0**)**  fp\_u **=** **new** FunctionalUnit**(**XML**,** ithCore**,&**interface\_ip**,** coredynp**,** FPU**);**  **if** **(**coredynp**.**num\_muls **>**0**)**  mul **=** **new** FunctionalUnit**(**XML**,** ithCore**,&**interface\_ip**,** coredynp**,** MUL**);** |

* FunctionalUnit中函数使用到的外部参数

|  |  |
| --- | --- |
| Name | Explanation |
| coredynp.clockRate | 时钟频率 |
| coredynp.executionTime | 执行时间 |
| coredynp.num\_fpus | Fpu的个数 |
| coredynp.num\_alus | Alu的个数 |
| coredynp.num\_muls | Mul的个数 |
| coredynp.core\_ty | 核的类型，inorder/OOO  按序情况下 base\_energy=0，没有基础消耗能量（未知） |
| coredynp.FPU\_duty\_cycle | FPU的占空比，一个周期内的工作比例 |
| coredynp.ALU\_duty\_cycle |  |
| coredynp.MUL\_duty\_cycle |  |
| XML->sys.core[ithCore]  .fpu\_accesses | Fpu的在执行过程中的访问次数 |
| XML->sys.core[ithCore]  .ialu\_accesses | Alu在执行过程中的访问次数 |
| XML->sys.core[ithCore]  .mul\_accesses | Mul在执行过程中的访问次数 |
| XML->sys.Embedded | 系统是否为嵌入式设备，例如ARM |
| interface\_ip.F\_sz\_um | 硬件参数 |

# FunctionalUnit的构造函数

|  |
| --- |
| //coredynp = dyn\_p\_  //XML = XML\_interface  //ithCore = ithCore\_  //interface\_ip = \*interface\_ip\_  //fu\_type = fu\_type\_  FunctionalUnit**::**FunctionalUnit**(**ParseXML **\***XML\_interface**,** int ithCore\_**,**  InputParameter**\*** interface\_ip\_**,**const CoreDynParam **&** dyn\_p\_**,**  enum FU\_type fu\_type\_**)**  **:**XML**(**XML\_interface**),**ithCore**(**ithCore\_**),**interface\_ip**(\***interface\_ip\_**),**  coredynp**(**dyn\_p\_**),**fu\_type**(**fu\_type\_**)**  **{**  //临时变量  double area\_t**;**  double pmos\_to\_nmos\_sizing\_r **=** pmos\_to\_nmos\_sz\_ratio**();**  //获取时钟和执行时间  clockRate **=** coredynp**.**clockRate**;**  executionTime **=** coredynp**.**executionTime**;**  //XML\_interface=\_XML\_interface;  uca\_org\_t result2**;**  //似乎没有直接使用到  result2 **=** init\_interface**(&**interface\_ip**);**    //是否是嵌入式处理器，例如ARM  **if** **(**XML**->**sys**.**Embedded**)**  **{**  **if** **(**fu\_type **==** FPU**){**  //获取fpu的个数  num\_fu**=**coredynp**.**num\_fpus**;**  //基本上都是电路计算，暂时不关心  **}**  **else** **if** **(**fu\_type **==** ALU**){**  num\_fu**=**coredynp**.**num\_alus**;**  **}**  **else** **if** **(**fu\_type **==** MUL**){**  num\_fu**=**coredynp**.**num\_muls**;**  **}**  **else{**  **}**  per\_access\_energy **\*=**0.5**;**  //According to ARM data embedded processor has much lower per acc energy  **}**  **else{**//非嵌入式的计算方式相同，但是参数不同  **}**    //计算面积, area\_t取决于系统类型(是否为嵌入式)，功能部分的类型  //n个功能部件  area**.**set\_area**(**area\_t**\***num\_fu**);**  //都是单个的值，需要乘以倍数  //leakage, gate\_leakage都和面积有关(area\_t)  leakage **\*=** num\_fu**;**  gate\_leakage **\*=**num\_fu**;**  //EDA设计时的参数  double macro\_layout\_overhead **=** g\_tp**.**macro\_layout\_overhead**;**  area**.**set\_area**(**area**.**get\_area**()\***macro\_layout\_overhead**);**  **}** |

# computeEnergy和displayEnergy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

|  |
| --- |
| void FunctionalUnit**::**computeEnergy**(**bool is\_tdp**)**  **{**  //  double pppm\_t**[**4**]** **=** **{**1**,**1**,**1**,**1**};**  double FU\_duty\_cycle**;**  //计算静态功耗/峰值功耗  **if** **(**is\_tdp**)**  **{**  **if** **(**fu\_type **==** FPU**)**  **{**  //认为每周期每个FPU都会被访问一次  stats\_t**.**readAc**.**access **=** num\_fu**;**  //tdp\_stats暂时未发现被使用  tdp\_stats **=** stats\_t**;**  //FU的占空比,(工作时间比例)  FU\_duty\_cycle **=** coredynp**.**FPU\_duty\_cycle**;**  **}**  //dynamic：每次访问的能量\*每周期的访问次数+正常状态每周期的能量开销\*周期  //此时记录的时工作周期的能量  power**.**readOp**.**dynamic **=** per\_access\_energy**\***stats\_t**.**readAc**.**access  **+** base\_energy**/**clockRate**;**  double sckRation **=** g\_tp**.**sckt\_co\_eff**;**  //结果再乘以每周期的工作时间比例\*sckRation  power**.**readOp**.**dynamic **\*=** sckRation**\***FU\_duty\_cycle**;**  power**.**writeOp**.**dynamic **\*=** sckRation**;**  power**.**searchOp**.**dynamic **\*=** sckRation**;**    //之前在构造函数中计算  power**.**readOp**.**leakage **=** leakage**;**  power**.**readOp**.**gate\_leakage **=** gate\_leakage**;**  double long\_channel\_device\_reduction **=**  longer\_channel\_device\_reduction**(**Core\_device**,** coredynp**.**core\_ty**);**  power**.**readOp**.**longer\_channel\_leakage **=**  power**.**readOp**.**leakage**\***long\_channel\_device\_reduction**;**  double pg\_reduction **=** power\_gating\_leakage\_reduction**(false);**  power**.**readOp**.**power\_gated\_leakage **=** power**.**readOp**.**leakage**\***pg\_reduction**;**  power**.**readOp**.**power\_gated\_with\_long\_channel\_leakage **=**  power**.**readOp**.**power\_gated\_leakage **\*** long\_channel\_device\_reduction**;**  **}**  **else**//计算动态功耗rt\_power  **{**  **if** **(**fu\_type **==** FPU**)**  **{**  //一共的访问次数  stats\_t**.**readAc**.**access **=** XML**->**sys**.**core**[**ithCore**].**fpu\_accesses**;**  //似乎也未被再次使用  rtp\_stats **=** stats\_t**;**  **}**    //dynamic=每次访问的能量\*所有的访问次数+每周期的基础能量\*整体的运行时间  //此时dynamic记录的是动态所消耗的能量  rt\_power**.**readOp**.**dynamic **=**  per\_access\_energy**\***stats\_t**.**readAc**.**access **+** base\_energy**\***executionTime**;**  double sckRation **=** g\_tp**.**sckt\_co\_eff**;**  rt\_power**.**readOp**.**dynamic **\*=** sckRation**;**  rt\_power**.**writeOp**.**dynamic **\*=** sckRation**;**  rt\_power**.**searchOp**.**dynamic **\*=** sckRation**;**  **}**  **}**  void FunctionalUnit**::**displayEnergy**(**uint32\_t indent**,**int plevel**,**bool is\_tdp**)**  **{**  bool long\_channel **=** XML**->**sys**.**longer\_channel\_device**;**  bool power\_gating **=** XML**->**sys**.**power\_gating**;**  /\*  Floating Point Units (FPUs) (Count: coredynp.num\_fpus  Area = area.get\_area()\*1e-6  Peak Dynamic = power.readOp.dynamic\*clockRate(每周期的能耗\*时钟周期)  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(所有时间的能耗/时间)  \*/  **}** |