**目录**

[1. Main.cc 2](#_Toc531716599)

[2. XML\_Parse.h/XML\_Parse.cc 2](#_Toc531716600)

[ 主要工作： 2](#_Toc531716601)

[ 定义ParseXML类 2](#_Toc531716602)

[ 结构体嵌套关系 3](#_Toc531716603)

[ CC文件中的解析函数的主要函数 5](#_Toc531716604)

[ CC文件中parse函数中的解析顺序 6](#_Toc531716605)

[ 其它 6](#_Toc531716606)

# Main.cc

* Main函数中包含指令参数解析、xml文件解析、创建processor对象、输出processor对象的信息

|  |
| --- |
| int main**(**int argc**,**char **\***argv**[])**  **{**  //参数解析  **...**  //parse XML-based interface  ParseXML **\***p1**=** **new** ParseXML**();**  p1**->**parse**(**fb**);**  //使用xml信息，创建processor对象  Processor proc**(**p1**);**  //输出energy等信息  proc**.**displayEnergy**(**2**,** plevel**);**  **delete** p1**;**  **return** 0**;**  **}** |

# XML\_Parse.h/XML\_Parse.cc

* 主要工作： 使用结构体的方式定义了XML可能包含的所有参数；所有参数的初始化；使用xml的解析库完成xml的解析，并将xml获得的信息写入到参数中
* 定义ParseXML类

|  |
| --- |
| //typdef predictor\_systemcore;  //typdef itlb\_systemcore;  //typdef icache\_systemcore;  //typdef dtlb\_systemcore;  //typdef dcache\_systemcore;  //typdef BTB\_systemcore;  //typdef system\_core;  //typdef system\_L1Directory;  //typdef system\_L2Directory;  //typdef system\_L2;  //typdef system\_L3;  //typdef xbar0\_systemNoC;  //typdef system\_NoC;  //typdef system\_mem;  //typdef system\_mc;  //typdef system\_niu;  //typdef system\_pcie;  //typdef root\_system;  class ParseXML  **{**  public**:**  //调用ParseXML类解析XML，并且设置定义的参数  void parse**(**char**\*** filepath**);**  //初始化定义的参数  void initialize**();**  public**:**  //系统参数  root\_system sys**;**  **};** |

* 结构体嵌套关系

|  |
| --- |
| //sys.core[i].predictor  **typedef** struct**{**  **}** predictor\_systemcore**;**  //sys.core[i].itlb  **typedef** struct**{**  **}** itlb\_systemcore**;**  //sys.core[i].icache  **typedef** struct**{**  **}** icache\_systemcore**;**  //sys.core[i].dtlb  **typedef** struct**{**  **}** dtlb\_systemcore**;**  //sys.core[i].dcache  **typedef** struct**{**  **}** dcache\_systemcore**;**  //sys.core[i].BTB  **typedef** struct**{**  **}** BTB\_systemcore**;**  //sys.core[i]  **typedef** struct**{**  predictor\_systemcore predictor**;**  itlb\_systemcore itlb**;**  icache\_systemcore icache**;**  dtlb\_systemcore dtlb**;**  dcache\_systemcore dcache**;**  BTB\_systemcore BTB**;**  **}** system\_core**;**  //sys.L1Directory[i]  **typedef** struct**{**  **}** system\_L1Directory**;**  //sys.L2Directory[i]  **typedef** struct**{**  **}** system\_L2Directory**;**  //sys.L2[i]  **typedef** struct**{**  **}** system\_L2**;**  //sys.L3[i]  **typedef** struct**{**  **}** system\_L3**;**  //sys.NoC[i].xbar0  **typedef** struct**{**  **}** xbar0\_systemNoC**;**  //sys.Noc[i]  **typedef** struct**{**  xbar0\_systemNoC xbar0**;**  **}** system\_NoC**;**  //sys.mem  **typedef** struct**{**  **}** system\_mem**;**  //sys.mc sys.flashc  **typedef** struct**{**  **}** system\_mc**;**  //sys.niu  **typedef** struct**{**  **}** system\_niu**;**  //sys.system\_pcie  **typedef** struct**{**  **}** system\_pcie**;**  //sys  **typedef** struct**{**  system\_core core**[**64**];**  system\_L1Directory L1Directory**[**64**];**  system\_L2Directory L2Directory**[**64**];**  system\_L2 L2**[**64**];**  system\_L3 L3**[**64**];**  system\_NoC NoC**[**64**];**  system\_mem mem**;**  system\_mc mc**;**  system\_mc flashc**;**  system\_niu niu**;**  system\_pcie pcie**;**  **}** root\_system**;** |

* CC文件中的解析函数的主要函数

|  |
| --- |
| //获取第一层，主节点root  XMLNode xMainNode**=**XMLNode**::**openFileHelper**(**filepath**,**"component"**);**  //获取第二层，组件system  XMLNode xNode2**=**xMainNode**.**getChildNode**(**"component"**);**  //获取system组件中参数/状态的个数  itmp**=**xNode2**.**nChildNode**(**"param"**);**  itmp**=**xNode2**.**nChildNode**(**"stat"**);**  //获取system组件中第i个参数/状态的名称，返回值为char\*，使用strcmp进行比较  char **\*** **=** xNode2**.**getChildNode**(**"param"**,**i**).**getAttribute**(**"name"**);**  char **\*** **=** xNode2**.**getChildNode**(**"stat"**,**i**).**getAttribute**(**"name"**);**  //获取system组件中第i个参数/状态的数值，返回值为char\*，使用atoi转换为数值  char **\*** **=** xNode2**.**getChildNode**(**"param"**,**i**).**getAttribute**(**"value"**);**  char **\*** **=** xNode2**.**getChildNode**(**"stat"**,**i**).**getAttribute**(**"value"**);**  //获取system组件中子组件的个数  NumofCom\_3**=**xNode2**.**nChildNode**(**"component"**);**  //如果是同构核，则只需要提取一个core的信息即可(same core)  **if** **(**sys**.**homogeneous\_cores**==**1**)**  OrderofComponents\_3layer**=**0**;**  **else**  OrderofComponents\_3layer**=**sys**.**number\_of\_cores**-**1**;**  //获取system下一层的第i个组件，从0开始  xNode3**=**xNode2**.**getChildNode**(**"component"**,**i**);**  //比较组件的名称中是否包含“core”  //strstr用于判断后者是不是前者的子串  //用于寻找组件名称类似于core0的组件  strstr**(**xNode3**.**getAttribute**(**"name"**),**"core"**)!=NULL**  //获取core组件的子组件个数  NumofCom\_4**=**xNode3**.**nChildNode**(**"component"**);**  //类似于之前的功能  strstr**(**xNode3**.**getAttribute**(**"id"**),**"L1Directory"**)!=NULL**  strstr**(**xNode3**.**getAttribute**(**"id"**),**"L2Directory"**)!=NULL** |

* CC文件中parse函数中的解析顺序

|  |
| --- |
| //第一层为root  //第二层为system  //param+stat  //第三层  //core0..n  //param+stat  //PBT,itlb,icache,dtlb,dcache,BTB  //param+stat    //L1directory0..n  //param+stat  //L2directory0..n  //param+stat  //L2 cache  //param+stat  //L3 cache  //param+stat  //NoC  //param+stat  //MC  //param+stat  //NIU  //param+stat  //PCIe  //param+stat  //Flashc  //param+stat |

* 其它

1. 当value有多个值时，这些值以逗号分割，此时读入进来为char\*，需要自己解析出每一个值
2. 当某些组件可能有多个时，必须明确指明。因为获取组件时，需要具体指明是第几个组件

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*提取core组件时，判断core的个数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  //如果是同构核，则只需要提取一个core的信息即可(same core)  **if** **(**sys**.**homogeneous\_cores**==**1**)**  OrderofComponents\_3layer**=**0**;**  **else**  OrderofComponents\_3layer**=**sys**.**number\_of\_cores**-**1**;**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*提取L1/L2directory组件时，判断个数和当前位置\*\*\*\*\*\*\*\*\*\*\*\*/  //当前组件的位置  int w**=**OrderofComponents\_3layer**+**1**;**  //判断L1directory的个数(OrderofComponents\_3layer-sys.tmpOrderofComponents\_3layer)  int tmpOrderofComponents\_3layer**=**OrderofComponents\_3layer**;**  **if** **(**sys**.**homogeneous\_L1Directories**==**1**)**  OrderofComponents\_3layer**=**OrderofComponents\_3layer**+**1**;**  **else**  OrderofComponents\_3layer**=**OrderofComponents\_3layer**+**sys**.**number\_of\_L1Directories**;**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*提取L2/L3 cache组件时，判断个数和当前位置\*\*\*\*\*\*\*\*\*\*\*\*/  w**=**OrderofComponents\_3layer**+**1**;**  tmpOrderofComponents\_3layer**=**OrderofComponents\_3layer**;**  **if** **(**sys**.**homogeneous\_L2s**==**1**)**  OrderofComponents\_3layer**=**OrderofComponents\_3layer**+**1**;**  **else**  OrderofComponents\_3layer**=**OrderofComponents\_3layer**+**sys**.**number\_of\_L2s**;**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*提取NoC组件时，判断个数和当前位置\*\*\*\*\*\*\*\*\*\*\*\*/  w**=**OrderofComponents\_3layer**+**1**;**  tmpOrderofComponents\_3layer**=**OrderofComponents\_3layer**;**  **if** **(**sys**.**homogeneous\_NoCs**==**1**)**  OrderofComponents\_3layer**=**OrderofComponents\_3layer**+**1**;**  **else**  OrderofComponents\_3layer**=**OrderofComponents\_3layer**+**sys**.**number\_of\_NoCs**;**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*提取MC/NIU/PCIe/Flashc组件时，判断当前位置,一个\*\*\*\*\*\*\*\*\*\*\*\*/  **if** **(**OrderofComponents\_3layer**>**0**)**  OrderofComponents\_3layer**=**OrderofComponents\_3layer**+**1**;** |