| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibVsb/src](http://docs.google.com/index.html) - vchip.cpp (source / [functions](http://docs.google.com/vchip.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 76 | 176 | 43.2 % | | Date: | 2012-03-23 |  | Functions: | 9 | 40 | 22.5 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 12 | 48 | 25.0 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   2 : : \* Copyrighted 2007 by Interra Systems, USA  3 : : \* All rights reserved   4 : : \*  5 : : \* Date of Creation: Fri Jul 20 15:56:46 IST 2007  6 : : \* File Name: vchip.cpp  7 : : \* Purpose:   8 : : \* Author:   9 : : \*  10 : : \* Revision History:   11 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/   12 : :   13 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  14 : : \* Header Include section   15 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  16 : : #include <iostream>  17 : :   18 : : #include <stdio.h>  19 : : #include "vchip.h"  20 : : #include "chipcnf.h"  21 : : #include "print.h"  22 : : namespace nft{  23 : :   24 : : ChipCnfFileParams \*  25 : 2250 : ChipCnfFileParams::instance() {  26 [ + + ][ + - ]: 2250 : static ChipCnfFileParams obj;  27 : 2250 : return &obj;  28 : : }  29 : :   30 : 0 : void ChipCnfFileParams::set\_sFrmtVer\_(const std::string sFrmtVer) {  31 : 0 : m\_sFrmtVer = sFrmtVer;  32 : : }  33 : :   34 : 0 : void ChipCnfFileParams::set\_sConvSys\_(const std::string sConvSys) {  35 : 0 : m\_sConvSys = sConvSys;  36 : : }  37 : :   38 : 0 : void ChipCnfFileParams::set\_sConvMchn\_(const std::string sConvMchn) {  39 : 0 : m\_sConvMchn = sConvMchn;  40 : : }  41 : :   42 : 0 : void ChipCnfFileParams::set\_sConvMchnID\_(const std::string sConvMchnID) {  43 : 0 : m\_sConvMchnID = sConvMchnID;  44 : : }  45 : :   46 : 0 : void ChipCnfFileParams::set\_sConvMchnOS\_(const std::string sConvMchnOS) {  47 : 0 : m\_sConvMchnOS = sConvMchnOS;  48 : : }  49 : :   50 : 0 : void ChipCnfFileParams::set\_sChipName\_(const std::string sChipName) {  51 : 0 : m\_sChipName = sChipName;  52 : : }  53 : :   54 : 0 : void ChipCnfFileParams::set\_ulChipAuRef\_(const unsigned long ulChipAuRef) {  55 : 0 : m\_ulChipAuRef = ulChipAuRef;  56 : : }  57 : :   58 : 0 : void ChipCnfFileParams::set\_llChipSizeX\_(const long long llChipSizeX) {  59 : 0 : m\_llChipSizeX = llChipSizeX;  60 : : }  61 : :   62 : 0 : void ChipCnfFileParams::set\_llChipSizeY\_(const long long llChipSizeY) {  63 : 0 : m\_llChipSizeY = llChipSizeY;  64 : : }  65 : :   66 : 0 : void ChipCnfFileParams::set\_llChipBlkSizeX\_(const long long llChipBlkSizeX) {  67 : 0 : m\_llChipBlkSizeX = llChipBlkSizeX;  68 : : }  69 : :   70 : 0 : void ChipCnfFileParams::set\_llChipBlkSizeY\_(const long long llChipBlkSizeY) {  71 : 0 : m\_llChipBlkSizeY = llChipBlkSizeY;  72 : : }  73 : :   74 : 0 : void ChipCnfFileParams::set\_uiChipFrms\_(const unsigned int uiChipFrms) {  75 : 0 : m\_uiChipFrms = uiChipFrms;  76 : : }  77 : :   78 : 0 : void ChipCnfFileParams::set\_sFrmOrntn\_(const std::string sFrmOrntn) {  79 : 0 : m\_sFrmOrntn = sFrmOrntn;  80 : : }  81 : :   82 : 0 : void ChipCnfFileParams::set\_uiFrmHeight\_(const unsigned int uiFrmHeight) {  83 : 0 : m\_uiFrmHeight = uiFrmHeight;  84 : : }  85 : :   86 : 0 : void ChipCnfFileParams::set\_uiFrmWidth\_(const unsigned int uiFrmWidth) {  87 : 0 : m\_uiFrmWidth = uiFrmWidth;  88 : : }  89 : :   90 : 0 : void ChipCnfFileParams::set\_sCellPlcmntRule\_(const std::string sCellPlcmntRule) {  91 : 0 : m\_sCellPlcmntRule = sCellPlcmntRule;  92 : : }  93 : :   94 : 0 : void ChipCnfFileParams::set\_llCellMaxSizeX\_(const long long llCellMaxSizeX) {  95 : 0 : m\_llCellMaxSizeX = llCellMaxSizeX;  96 : : }  97 : :   98 : 0 : void ChipCnfFileParams::set\_llCellMaxSizeY\_(const long long llCellMaxSizeY) {  99 : 0 : m\_llCellMaxSizeY = llCellMaxSizeY;  100 : : }  101 : :   102 : 0 : void ChipCnfFileParams::set\_llCellArrMaxSizeX\_(const long long llCellArrMaxSizeX) {  103 : 0 : m\_llCellArrMaxSizeX = llCellArrMaxSizeX;  104 : : }  105 : :   106 : 0 : void ChipCnfFileParams::set\_llCellArrMaxSizeY\_(const long long llCellArrMaxSizeY) {  107 : 0 : m\_llCellArrMaxSizeY = llCellArrMaxSizeY;  108 : : }  109 : :   110 : 0 : void ChipCnfFileParams::set\_bPatArrCmprssn\_(const bool bPatArrCmprssn) {  111 : 0 : m\_bPatArrCmprssn = bPatArrCmprssn;  112 : : }  113 : :   114 : 0 : void ChipCnfFileParams::set\_bPatCmprssn\_(const bool bPatCmprssn) {  115 : 0 : m\_bPatCmprssn = bPatCmprssn;  116 : : }  117 : :   118 : 0 : void ChipCnfFileParams::set\_uiPatSet\_(const unsigned int uiPatSet) {  119 : 0 : m\_uiPatSet = uiPatSet;  120 : : }  121 : :   122 : 0 : void ChipCnfFileParams::set\_uiMaxDataRef\_(const unsigned int uiMaxDataRef) {  123 : 0 : m\_uiMaxDataRef = uiMaxDataRef;  124 : : }  125 : :   126 : 0 : void ChipCnfFileParams::set\_uiMaxDataCell\_(const unsigned int uiMaxDataCell) {  127 : 0 : m\_uiMaxDataCell = uiMaxDataCell;  128 : : }  129 : :   130 : 0 : void ChipCnfFileParams::set\_uiMaxDataCellComm\_(const unsigned int uiMaxDataCellComm) {  131 : 0 : m\_uiMaxDataCellComm = uiMaxDataCellComm;;  132 : : }  133 : :   134 : 0 : void ChipCnfFileParams::set\_iFrameTranslateY(const int iFrameTranslateY) {  135 : 0 : m\_iFrameTranslateY = iFrameTranslateY;  136 : : }  137 : :   138 : 0 : void ChipCnfFileParams::set\_dScale\_(const double dScale) {  139 : 0 : m\_dScale = dScale;  140 : : }  141 : 0 : void ChipCnfFileParams::set\_iMirror\_(const int iMirror) {  142 : 0 : m\_iMirror = iMirror;  143 : : }  144 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  145 : : \* Function Name: parseChipCnf()   146 : : \* Purpose: Parse the chip.cnf file  147 : : \* Parameters: std::string chipCnfDir  148 : : \* Return Values: void  149 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  150 : : bool ChipCnfFileParams::parseChipCnf(std::string chipCnfDir)  151 : 1125 : {  152 : :   153 : : // TODO Sourasis : AU should be propagated from PDGparams obj  154 : :   155 : 1125 : m\_Parser.set\_sChipCnfPath\_(chipCnfDir);  156 : 1125 : m\_Parser.parseNonFramePart();  157 : :   158 : 1119 : const \_chipCnfFile \*cnfFile = m\_Parser.get\_cnfFile\_();  159 : :   160 : 1119 : this->m\_sFrmtVer = cnfFile->frmtVersion;  161 : 1119 : this->m\_sConvSys = cnfFile->convSystem;  162 : 1119 : this->m\_sConvMchn = cnfFile->convMachine;  163 : 1119 : this->m\_sConvMchnID = cnfFile->convMachineId;  164 : 1119 : this->m\_sConvMchnOS = cnfFile->convMachineOS;  165 : 1119 : this->m\_sChipName = cnfFile->chipName;  166 : 1119 : this->m\_ulChipAuRef = cnfFile->chipAddrUnit;  167 : 1119 : this->m\_llChipSizeX = cnfFile->chipSizeX;  168 : 1119 : this->m\_llChipSizeY = cnfFile->chipSizeY;  169 : 1119 : this->m\_llChipBlkSizeX = cnfFile->chipBSizeX;  170 : 1119 : this->m\_llChipBlkSizeY = cnfFile->chipBSizeY;  171 : 1119 : this->m\_uiChipFrms = cnfFile->chipFrames;  172 : 1119 : this->m\_sFrmOrntn = cnfFile->frmOrientation;  173 : 1119 : this->m\_uiFrmHeight = cnfFile->frmHeight;  174 : 1119 : this->m\_uiFrmWidth = cnfFile->frmWid;  175 : : // this->m\_fiFrameStruct = cnfFile->frame;  176 : 1119 : this->m\_sCellPlcmntRule = cnfFile->multiPlacement;  177 : 1119 : this->m\_llCellMaxSizeX = cnfFile->cellMaxSizeX;  178 : 1119 : this->m\_llCellMaxSizeY = cnfFile->cellMaxSizeY;  179 : 1119 : this->m\_llCellArrMaxSizeX = cnfFile->cellArrMaxSizeX;  180 : 1119 : this->m\_llCellArrMaxSizeY = cnfFile->cellArrMaxSizeY;  181 : 1119 : this->m\_bPatArrCmprssn = cnfFile->patArrayCompress; //TODO char\*->bool  182 : 1119 : this->m\_bPatCmprssn = cnfFile->patCompress; //TODO -do-  183 : 1119 : this->m\_uiPatSet = cnfFile->patSet;  184 : 1119 : this->m\_uiMaxDataRef = cnfFile->maxDataRef;  185 : 1119 : this->m\_uiMaxDataCell = cnfFile->maxDataCell;  186 : 1119 : this->m\_uiMaxDataCellComm = cnfFile->maxDataCommon;  187 : 1119 : this->m\_iFrameTranslateY = cnfFile->frameTranslateY;  188 : :   189 : 1119 : return true;  190 : : }  191 : :   192 : : void ChipCnfFileParams::print() const  193 : 0 : {  194 : : #if 0   195 : : std::cout<<" VSB-12 version: "<<m\_sFrmtVer<<std::endl;  196 : : std::cout<<" Conversion system: "<<m\_sConvSys<<std::endl;  197 : : std::cout<<" Conversion machine: "<<m\_sConvMchn<<std::endl;  198 : :   199 : : \_frameInfo\*\* pFrame = NULL;//get\_fiFrameStruct\_();  200 : : uint32\_t frames = get\_uiChipFrms\_();  201 : :   202 : : for (uint32\_t i=0; i<frames; i++) {  203 : : std::cout<<std::endl;  204 : : std::cout<<" Frame No.: "<<i+1<<std::endl;  205 : : std::cout<<" Height (in blocks): "<<pFrame[0][i].height<<std::endl;  206 : : std::cout<<" Height (in VSB AU)"<<pFrame[0][i].vsbHeight<<std::endl;  207 : : std::cout<<" Width (in blocks): "<<pFrame[0][i].wid<<std::endl;  208 : : std::cout<<" Width (in VSB AU): "<<pFrame[0][i].vsbWid<<std::endl;  209 : : std::cout<<" Sub-frames: "<<pFrame[0][i].subFrms<<std::endl;  210 : : std::cout<<" X location: "<<pFrame[0][i].frmLocX<<std::endl;  211 : : std::cout<<" Y Location: "<<pFrame[0][i].frmLocY<<std::endl;  212 : : }  213 : : std::cout<<std::endl;  214 : : #endif   215 : : }  216 : :   217 : : uint32\_t ChipCnfFileParams::get\_frameHeight(uint32\_t i, uint32\_t j) const  218 : 2247 : {  219 : 2247 : uint32\_t frames = get\_uiChipFrms\_();  220 [ - + ]: 2247 : if (j>=frames)  221 : : {  222 : 0 : return 0;  223 : : }  224 : : else  225 : : {   226 [ + - ]: 2247 : if (0 == i)   227 : : {  228 : 2247 : try  229 : : {  230 : 2247 : m\_Parser.parseFrameInfo();  231 : : }  232 [ # # ]: 0 : catch( DPL::Error \*pChild )  233 : : {  234 : 0 : e9k\_error("ChipCnfFileParams::parseFrameInfo failed\n");  235 : : }  236 : :   237 : :   238 : 2247 : return m\_Parser.get\_pMatrix\_()->getFrameInfo(i,j)->height;  239 : : }  240 [ # # ]: 0 : else if ( get\_frameSubframeCount(j)> i)  241 : : {  242 : 0 : try  243 : : {  244 : 0 : m\_Parser.parseFrameInfo();  245 : : }  246 [ # # ]: 0 : catch( DPL::Error \*pChild )  247 : : {  248 : 0 : e9k\_error("ChipCnfFileParams::parseFrameInfo failed\n");  249 : : }  250 : :   251 : 0 : return m\_Parser.get\_pMatrix\_()->getFrameInfo(i,j)->height;  252 : : }  253 : : else  254 : : {   255 : : }  256 : : }  257 : 2247 : return 0; //FIXME:: exception  258 : : }  259 : :   260 : :   261 : : uint32\_t ChipCnfFileParams::get\_frameWidth(uint32\_t i, uint32\_t j) const  262 : 2247 : {  263 : 2247 : uint32\_t frames = get\_uiChipFrms\_();  264 [ - + ]: 2247 : if (j>=frames)  265 : : {  266 : 0 : return 0;  267 : : }  268 : : else  269 : : {  270 [ + - ]: 2247 : if (0 == i)  271 : : {  272 : 2247 : try  273 : : {  274 : 2247 : m\_Parser.parseFrameInfo();  275 : : }  276 [ # # ]: 0 : catch( DPL::Error \*pChild )  277 : : {  278 : 0 : e9k\_error("ChipCnfFileParams::parseFrameInfo failed\n");  279 : : }  280 : 2247 : return m\_Parser.get\_pMatrix\_()->getFrameInfo(i,j)->wid;  281 : : }  282 [ # # ]: 0 : else if ( get\_frameSubframeCount(j)> i)  283 : : {  284 : 0 : try  285 : : {  286 : 0 : m\_Parser.parseFrameInfo();  287 : : }  288 [ # # ]: 0 : catch( DPL::Error \*pChild )  289 : : {  290 : 0 : e9k\_error("ChipCnfFileParams::parseFrameInfo failed\n");  291 : : }  292 : 0 : return m\_Parser.get\_pMatrix\_()->getFrameInfo(i,j)->wid;  293 : : }  294 : : else  295 : : {  296 : : }  297 : : }  298 : 2247 : return 0; //FIXME:: exception  299 : : }  300 : :   301 : : uint32\_t ChipCnfFileParams::get\_frameSubframeCount(uint32\_t j) const  302 : 2247 : {  303 : 2247 : uint32\_t frames = get\_uiChipFrms\_();  304 [ - + ]: 2247 : if (j >= frames)  305 : : {  306 : 0 : return 0;  307 : : }  308 : : else  309 : : {  310 : 2247 : try  311 : : {  312 : 2247 : m\_Parser.parseFrameInfo();  313 : : }  314 [ # # ]: 0 : catch( DPL::Error \*pChild )  315 : : {  316 : 0 : e9k\_error("ChipCnfFileParams::parseFrameInfo failed\n");  317 : : }  318 : 2247 : return m\_Parser.get\_pMatrix\_()->getFrameInfo(0,j)->subFrms;  319 : : }  320 : 2247 : return 0;  321 : : }  322 : :   323 : : uint32\_t ChipCnfFileParams::get\_frameLocX(uint32\_t i, uint32\_t j) const  324 : 1119 : {  325 : 1119 : uint32\_t frames = get\_uiChipFrms\_();  326 [ - + ]: 1119 : if (j>=frames)  327 : : {  328 : 0 : return 0;  329 : : }  330 : : else  331 : : {  332 [ + - ]: 1119 : if (0 == i)  333 : : {  334 : 1119 : try  335 : : {  336 : 1119 : m\_Parser.parseFrameInfo();  337 : : }  338 [ # # ]: 0 : catch( DPL::Error \*pChild )  339 : : {  340 : 0 : e9k\_error("ChipCnfFileParams::parseFrameInfo failed\n");  341 : : }  342 : 1119 : return m\_Parser.get\_pMatrix\_()->getFrameInfo(i,j)->frmLocX;  343 : : }  344 [ # # ]: 0 : else if ( get\_frameSubframeCount(j)> i)  345 : : {  346 : 0 : try  347 : : {  348 : 0 : m\_Parser.parseFrameInfo();  349 : : }  350 [ # # ]: 0 : catch( DPL::Error \*pChild )  351 : : {  352 : 0 : e9k\_error("ChipCnfFileParams::parseFrameInfo failed\n");  353 : : }  354 : 0 : return m\_Parser.get\_pMatrix\_()->getFrameInfo(i,j)->frmLocX;  355 : : }  356 : : else  357 : : {  358 : : }  359 : : }  360 : 1119 : return 0; //FIXME:: exception  361 : : }  362 : :   363 : :   364 : : uint32\_t ChipCnfFileParams::get\_frameLocY(uint32\_t i, uint32\_t j) const  365 : 1119 : {  366 : 1119 : uint32\_t frames = get\_uiChipFrms\_();  367 [ - + ]: 1119 : if (j>=frames)  368 : : {  369 : 0 : return 0;  370 : : }  371 : : else  372 : : {  373 [ + - ]: 1119 : if (0 == i)  374 : : {  375 : 1119 : try  376 : : {  377 : 1119 : m\_Parser.parseFrameInfo();  378 : : }  379 [ # # ]: 0 : catch( DPL::Error \*pChild )  380 : : {  381 : 0 : e9k\_error("ChipCnfFileParams::parseFrameInfo failed\n");  382 : : }  383 : 1119 : return m\_Parser.get\_pMatrix\_()->getFrameInfo(i,j)->frmLocY;  384 : : }  385 [ # # ]: 0 : else if ( get\_frameSubframeCount(j)> i)  386 : : {  387 : 0 : try  388 : : {  389 : 0 : m\_Parser.parseFrameInfo();  390 : : }  391 [ # # ]: 0 : catch( DPL::Error \*pChild )  392 : : {  393 : 0 : e9k\_error("ChipCnfFileParams::parseFrameInfo failed\n");  394 : : }  395 : 0 : return m\_Parser.get\_pMatrix\_()->getFrameInfo(i,j)->frmLocY;  396 : : }  397 : : else  398 : : {  399 : : }  400 : : }  401 : 1119 : return 0; //FIXME:: exception  402 : : }  403 : :   404 : : void ChipCnfFileParams::deleteFrameMatrix()  405 : 0 : {  406 : 0 : m\_Parser.cleanUpFrameMatrix();  407 : : }  408 : : #if 0  409 : : void CRealChip::set\_sChipID\_(const std::string sChipID) {  410 : : m\_sChipID = sChipID;  411 : : }  412 : :   413 : :   414 : : void CRealChip::set\_lPosX\_(const long long lPosX) {  415 : : m\_lPosX = lPosX;  416 : : }  417 : :   418 : : void CRealChip::set\_lPosY\_(const long long lPosY) {  419 : : m\_lPosY = lPosY;  420 : : }  421 : :   422 : : void CRealChip::set\_lPitchX\_(const long long lPitchX) {  423 : : m\_lPitchX = lPitchX;  424 : : }  425 : :   426 : : void CRealChip::set\_lPitchY\_(const long long lPitchY) {  427 : : m\_lPitchY = lPitchY;  428 : : }  429 : :   430 : : void CRealChip::set\_iRepX\_(const int iRepX) {  431 : : m\_iRepX = iRepX;  432 : : }  433 : :   434 : : void CRealChip::set\_iRepY\_(const int iRepY) {  435 : : m\_iRepY = iRepY;  436 : : }  437 : :   438 : : void CRealChip::set\_sChipDir\_(const std::string sChipDir) {  439 : : m\_sChipDir = sChipDir;  440 : : }  441 : :   442 : : void CRealChip::set\_dScale\_(const double dScale) {  443 : : m\_dScale = dScale;  444 : : }  445 : :   446 : : void CRealChip::set\_iMirror\_(const int iMirror) {  447 : : m\_iMirror = iMirror;  448 : : }  449 : :   450 : : void CRealChip::print()  451 : : {  452 : : printf (" Chip ID: %s\n", m\_sChipID.c\_str());  453 : : printf (" Position: (%lld fm, %lld fm)\n", m\_lPosX, m\_lPosY);  454 : : printf (" Pitch: (X = %lld fm; Y = %lld fm)\n", m\_lPitchX, m\_lPitchY);  455 : : printf (" Repetition: (Nx = %d; Ny = %d)\n", m\_iRepX, m\_iRepY);  456 : : printf (" Chip directory path: %s\n", m\_sChipDir.c\_str());  457 : : printf (" Scaling: %f\n", m\_dScale);  458 : : printf (" Mirror: %d\n", m\_iMirror);  459 : :   460 : : const ChipCnfFileParams\* pCnfFileParams = get\_m\_chipCnf\_();  461 : : std::cout<<std::endl<<" From chip.cnf:"<<std::endl;  462 : : pCnfFileParams->print();  463 : : }  464 : : #endif  465 : : #if 0  466 : : CVChip::~CVChip()  467 : : {  468 : : reset();  469 : : }  470 : : void CVChip::reset()  471 : : {  472 : : for(std::vector<const CRealChip \*>::iterator it = m\_cChip.begin();  473 : : it != m\_cChip.end(); it++)  474 : : {  475 : : CRealChip \*pRealChip = const\_cast<CRealChip \*>(\*it);  476 : : delete pRealChip;  477 : : }  478 : : m\_cChip.clear();  479 : : }  480 : :   481 : : void CVChip::add(const CRealChip \*pRealChip)  482 : : {  483 : : if (pRealChip == NULL)  484 : : return;  485 : :   486 : : m\_cChip.push\_back (pRealChip);  487 : : }  488 : :   489 : : void CVChip::set\_lPosX\_(const long long lPosX) {  490 : : m\_lPosX = lPosX;  491 : : }  492 : :   493 : : void CVChip::set\_lPosY\_(const long long lPosY) {  494 : : m\_lPosY = lPosY;  495 : : }  496 : :   497 : : void CVChip::set\_lSizeX\_(const long long lSizeX) {  498 : : m\_lSizeX = lSizeX;  499 : : }  500 : :   501 : : void CVChip::set\_lSizeY\_(const long long lSizeY) {  502 : : m\_lSizeY = lSizeY;  503 : : }  504 : :   505 : : void CVChip::print()  506 : : {  507 : : printf (" Position: (%lld fm, %lld fm)\n", m\_lPosX, m\_lPosY);  508 : : printf (" Size: (%lld fm x %lld fm)\n\n", m\_lSizeX, m\_lSizeY);  509 : :   510 : : printf (" Parameters:\n");  511 : : printf (" SDG Block Size X: %f um\n", m\_cParam.dSDGBlockSizeX\_);  512 : : printf (" Stripe Height: %f um\n", m\_cParam.dStripeHeight\_);  513 : : printf (" PDG AU: %f um\n", m\_cParam.dPDGAU\_);  514 : : printf (" Cluster Size: %f um\n", m\_cParam.dClusterSize\_);  515 : : printf (" SDG Block Count along Y dir: %d\n\n", m\_cParam.iSDGBlockCountY\_);  516 : :   517 : : for (int i = 0; i < this->getChipNum(); i++)  518 : : {  519 : : printf (" Real Chip No: %d\n", i+1);  520 : : //const CRealChip \*pChip = this->getChip(i);  521 : : //if (pChip != NULL)  522 : : // pChip->print();  523 : : }  524 : : }  525 : :   526 : : void CLayout::add(const char\* sID, const CVChip \*pVChip)  527 : : {  528 : : if (sID == NULL || pVChip == NULL)  529 : : return;  530 : :   531 : : m\_cVChip[sID] = const\_cast<CVChip \*>(pVChip);  532 : : }  533 : :   534 : : void CLayout::getVChipIndentifierList(std::vector<std::string>& cVChipIndentifier)  535 : : {  536 : : cVChipIndentifier.clear();  537 : :   538 : : std::map<std::string, CVChip \*>::iterator it;  539 : :   540 : : for (it=m\_cVChip.begin();it!=m\_cVChip.end();it++)  541 : : {  542 : : cVChipIndentifier.push\_back (it->first.c\_str());  543 : : }  544 : : }  545 : :   546 : : CVChip\* CLayout::getVChip (const char\* sID)  547 : : {  548 : : if ((sID == NULL) || (0 >= strlen(sID)))  549 : : {  550 : : return NULL;  551 : : }  552 : :   553 : : std::map<std::string, CVChip \*>::iterator it = m\_cVChip.find (sID);  554 : :   555 : : if (it == m\_cVChip.end())  556 : : {  557 : : return NULL;  558 : : }  559 : :   560 : : return (\*it).second;  561 : : }  562 : :   563 : : void CLayout::print()  564 : : {  565 : : std::vector<std::string> cVChipIDList;  566 : : this->getVChipIndentifierList(cVChipIDList);  567 : :   568 : : for (unsigned int i = 0; i < cVChipIDList.size(); i++)  569 : : {  570 : : printf ("Virtual Chip No: %d\n", i+1);  571 : : const char \*sVChipID = cVChipIDList[i].c\_str();  572 : : CVChip \*pVChip = this->getVChip(sVChipID);  573 : : if (pVChip != NULL)  574 : : pVChip->print();  575 : : }  576 : : }  577 : :   578 : : #endif  579 : :   580 : 1215 : } // namespace nft |

|  |
| --- |
| Generated by: [LCOV version 1.9](http://ltp.sourceforge.net/coverage/lcov.php) |