| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibVsb/src](http://docs.google.com/index.html) - ChipCnfParser.cpp (source / [functions](http://docs.google.com/ChipCnfParser.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 198 | 322 | 61.5 % | | Date: | 2012-03-23 |  | Functions: | 17 | 18 | 94.4 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 133 | 294 | 45.2 % | |  | |
|  |

|  |
| --- |
| 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 13:21:56 IST 2007  6 : : \* File Name: ChipCnfParser.cpp  7 : : \* Purpose: A parser for chip.cnf  8 : : \* Author: Santanu Sinha  9 : : \*  10 : : \* Revision History:   11 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  12 : : #include <iostream>  13 : : #include <sstream>  14 : : #include <cmath>  15 : : #include <errno.h>  16 : : #include "ErrorHandler.h"  17 : : #include "Logger.h"  18 : : #include "ChipCnfParser.h"  19 : :   20 : : namespace nft{  21 : :   22 : : using namespace std;  23 : :   24 : : ChipCnfParser::ChipCnfParser()  25 : 1125 : : sChipCnfPath\_(),  26 : : uiNumLinesInIchipCnf\_(0),  27 : : errSet\_(0),  28 : : cnfFile\_(),  29 : : configBuf\_(NULL),  30 : : pMatrix\_(NULL),  31 : : bIsMatrixParsed\_(false)  32 : 1125 : {  33 : : //Hoge Hoge  34 : : }  35 : :   36 : : ChipCnfParser::~ChipCnfParser()  37 : 2250 : {  38 [ + + ][ # # ]: 1125 : delete pMatrix\_;  39 : 1125 : pMatrix\_ = NULL;  40 : 1125 : destroyConfigBuf();  41 : : }  42 : :   43 : :   44 : : void ChipCnfParser::createFrmMatrix(const uint32\_t rows, const uint32\_t cols)  45 : 1119 : {  46 : 1119 : pMatrix\_ = new \_frmMatrix(rows, cols);  47 : : }  48 : :   49 : : int ChipCnfParser::createConfigBuf()  50 : 2244 : {  51 : 2244 : FILE \*fp = NULL;  52 : 2244 : char \*cp = NULL;  53 : 2244 : int lineNo = -1;  54 : 2244 : char linebuf[BUFSIZ];  55 : 2244 : fp = fopen(sChipCnfPath\_.c\_str(), "r");  56 : : //#ifdef ERROR\_SEED //<<Test\_Id:ChipCnfParser\_fp\_Set\_To\_NULL>>  57 : : // fp = NULL;  58 : : //#endif  59 [ + + ]: 2244 : if( NULL == fp)  60 : : {  61 : 6 : ErrorHandler \*pErr = ErrorHandler::instance();  62 : 6 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  63 : : ,ErrorHandler::eErrFileOpen  64 : : ,pErr->getErrorMsg((ErrorHandler::eErrFileOpen)  65 : : ,sChipCnfPath\_.c\_str()));  66 : : }  67 : :   68 : : /\* Count total number of lines in chip.cnf \*/  69 [ + + ]: 53718 : for (lineNo = 0; (cp = fgets(linebuf, sizeof(linebuf), fp)); lineNo++);  70 : 2238 : uiNumLinesInIchipCnf\_ = lineNo;  71 : :   72 : 2238 : configBuf\_ = new(std::nothrow) char \*[lineNo];  73 : : //#ifdef ERROR\_SEED //<<Test\_Id:ChipCnfParser\_configBuf\_Set\_To\_false>>  74 : : // configBuf\_=false;  75 : : //#endif  76 [ - + ]: 2238 : if( !configBuf\_ )  77 : : {  78 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  79 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  80 : : ,ErrorHandler::eErrSystem  81 : : ,pErr->getErrorMsg(ErrorHandler::eErrSystem));  82 : : }  83 : :   84 : 2238 : rewind(fp) ;  85 [ + - ][ + + ]: 51480 : for (int i = 0 ; i < lineNo; i++)  86 : : {  87 : 51480 : configBuf\_[i] = 0;  88 : : }  89 [ + - ][ + + ]: 53718 : for (int i = 0 ; i < lineNo; i++)  90 : : {  91 : 51480 : cp = fgets(linebuf, sizeof(linebuf), fp);  92 : : // PR 13650 :: do not delete comment lines .. will be   93 : : // handled in get\_tokens  94 : :   95 [ + - ]: 51480 : if(cp)  96 : : {  97 : 51480 : configBuf\_[i] = new(std::nothrow) char[strlen(cp) + 1];  98 : : //#ifdef ERROR\_SEED //<<Test\_Id:ChipCnfParser\_configBuf\_i\_Position\_Set\_To\_0>>  99 : : // configBuf\_[i] = 0;  100 : : //#endif  101 [ - + ]: 51480 : if( ! configBuf\_[i])  102 : : {  103 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  104 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  105 : : ,ErrorHandler::eErrSystem  106 : : ,pErr->getErrorMsg(ErrorHandler::eErrSystem));  107 : : }  108 : 51480 : strcpy(configBuf\_[i], cp);  109 : : }  110 : : }  111 : :   112 [ + - ]: 2238 : if(fp)  113 : 2238 : fclose(fp);  114 : :   115 : 2238 : return lineNo;  116 : :   117 : : }  118 : :   119 : : void ChipCnfParser::destroyConfigBuf()  120 : 3363 : {  121 : : // If the buffer was deleted normally, don't do anything  122 [ # # ][ # # ]: 3363 : if( NULL == configBuf\_ )   [ - + ][ # # ]  [ + - ][ + - ]  123 : 2238 : return;  124 : :   125 : : // Cleanup memory in exception case  126 [ # # ][ # # ]: 53718 : for(size\_t i = 0; i < uiNumLinesInIchipCnf\_; i++)  [ # # ][ # # ]  [ # # ][ # # ]  [ # # ][ # # ]  [ + - ][ + + ]  [ + - ][ + + ]  127 : : {  128 [ # # ][ # # ]: 51480 : if( configBuf\_[i] )  [ # # ][ # # ]  [ + - ][ + - ]  129 : : {   130 : 51480 : delete[] configBuf\_[i];  131 : : }  132 : : }  133 [ # # ][ # # ]: 2238 : delete[] configBuf\_;  [ # # ][ # # ]  [ + - ][ + - ]  134 : 2238 : configBuf\_ = NULL;  135 : :   136 : : }  137 : :   138 : : template<class T> void ChipCnfParser::readConfigNum(  139 : : const char \*keyword, T &value, bool bOptional )  140 : 24645 : {  141 : 24645 : char dlm[] = ":";  142 : 24645 : char token[ChipCnfParser::MAX\_VALUE\_TOKENS][LEN\_LINE] ;  143 : :   144 [ + - ][ + + ]: 450571 : for (size\_t i = 0; i < uiNumLinesInIchipCnf\_; i++)   [ + - ][ + + ]  [ + - ][ + + ]  [ + - ][ + - ]  145 : : {  146 [ + - ][ + - ]: 441594 : if ( NULL == configBuf\_[i] )  [ + - ][ + - ]  147 : 441594 : continue;  148 : :   149 : 441594 : int ntokens = get\_tokens(configBuf\_[i], dlm, token) ;  150 [ + - + - : 441594 : if( ntokens != ChipCnfParser::NUM\_REQUIRED\_TOKENS )  + - + - ]  151 : : {  152 : : // PR 13650:: In case there was a blank or comment line  153 : 441594 : continue;  154 : : }  155 : : //#ifdef ERROR\_SEED //<<Test\_Id:ChipCnfParser\_readConfigNum>>  156 : : // int str = strcmp( keyword, token[0] );  157 : : // str = 0;  158 [ + + ][ + + ]: 441594 : if( 0 == strcmp( keyword, token[0] ))  [ + + ][ + + ]  159 : : {  160 : : //#endif  161 : 15668 : char \*endPtr = 0;  162 : 15668 : long long tmpVal = strtoll(token[1], &endPtr, 10);  163 : :   164 : : // Check for out of range or invalid values  165 [ + - ][ + - ]: 15668 : if ( token[1] == endPtr || ERANGE == errno   [ + - + - ]  [ + - ]  [ + + + - ]  [ + - ]  [ + - + - ]  [ + - ][ + - ]  166 : : || (tmpVal > 0 && static\_cast<unsigned long long>(tmpVal) >   167 : : static\_cast<unsigned long long>(  168 : : std::numeric\_limits<T>::max()))   169 : : || tmpVal < static\_cast<long long>(  170 : : std::numeric\_limits<T>::min()) )  171 : : {  172 : 0 : char msg[1024];  173 : 0 : sprintf(msg,"Value of %s is out of range in ichip.cnf",keyword);   174 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  175 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  176 : : ,ErrorHandler::eErrMessageString  177 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  178 : : ,msg));  179 : : }  180 : 15668 : value = static\_cast<T>(tmpVal);  181 : 15668 : return;  182 : : }  183 : :   184 : : }  185 : : //#ifdef ERROR\_SEED //<<Test\_Id:readConfigNum\_bOptional\_Set\_To\_false>>  186 : : // bOptional = false;  187 : : //#endif  188 [ - + ][ - + ]: 8977 : if( false == bOptional )  [ - + ][ # # ]  189 : : {  190 : :   191 : 0 : char msg[1024];  192 : 0 : sprintf(msg,"Value of %s is not given in ichip.cnf",keyword);   193 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  194 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  195 : : ,ErrorHandler::eErrMessageString  196 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  197 : : ,msg));  198 : : }  199 : :   200 : 8977 : value = 0;  201 : 24645 : return;  202 : :   203 : : }  204 : :   205 : : char \*ChipCnfParser::readConfigStr(const char \*keyword, bool bOptional )  206 : 11190 : {  207 : 11190 : char dlm[] = ":";  208 : 11190 : char token[MAX\_VALUE\_TOKENS][LEN\_LINE] ;  209 : :   210 [ + - ][ + + ]: 100722 : for (size\_t i = 0; i < uiNumLinesInIchipCnf\_; i++)   211 : : {  212 [ + - ]: 99603 : if ( NULL == configBuf\_[i] )  213 : 99603 : continue;  214 : :   215 : 99603 : int ntokens = get\_tokens(configBuf\_[i], dlm, token) ;  216 [ + - ]: 99603 : if( ntokens != ChipCnfParser::NUM\_REQUIRED\_TOKENS )  217 : : {  218 : : // PR 13650:: In case there was a blank or comment line  219 : 99603 : continue;   220 : : }  221 : :   222 [ + + ]: 99603 : if( 0 == strcmp( keyword, token[0] ))  223 : : {  224 : 10071 : removechar(token[1], '\"') ;  225 : 10071 : trim(token[1]) ;  226 [ + + ]: 20131 : char \*ptr = new char[strlen(token[1]) + 1];  227 : 10071 : strcpy(ptr, token[1]);  228 : 10071 : return ptr;  229 : : }  230 : : }  231 : : //#ifdef ERROR\_SEED //<<Test\_Id:readConfigStr\_bOptional\_Set\_To\_false>>  232 : : // bOptional = false;  233 : : //#endif  234 [ - + ]: 1119 : if( false == bOptional )  235 : : {  236 : 0 : char msg[1024];  237 : 0 : sprintf(msg,"Value of %s is not given in ichip.cnf",keyword);   238 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  239 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  240 : : ,ErrorHandler::eErrMessageString  241 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  242 : : ,msg));  243 : : }  244 : : // For optional string which has not been given  245 : 11190 : return NULL;  246 : : }  247 : :   248 : : void ChipCnfParser::errorMsg(std::string errorString) throw(DPL::Error \*)  249 : 0 : {  250 : 0 : destroyConfigBuf();  251 : 0 : char linebuf[BUFSIZ];  252 : 0 : sprintf(linebuf, "Value of %s is out of range in ichip.cnf", errorString.c\_str());  253 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  254 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  255 : : ,ErrorHandler::eErrMessageString  256 : : ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  257 : : ,linebuf));  258 : :   259 : : }  260 : : void ChipCnfParser::parseNonFramePart()  261 : 1125 : {  262 : 1125 : std::string errorString;  263 : 1125 : if( sChipCnfPath\_.empty() )  264 : 0 : return;   265 : :   266 : 1125 : uiNumLinesInIchipCnf\_ = createConfigBuf();  267 : :   268 : 1119 : cnfFile\_.frmtVersion = readConfigStr("Format.Version");  269 : : //#ifdef ERROR\_SEED //<<Test\_Id:ChipCnfParser\_cnfFile\_frmtVersion\_Set\_To\_false>>  270 : : // cnfFile\_.frmtVersion = false;  271 : : //#endif  272 [ - + ]: 1119 : if (!cnfFile\_.frmtVersion)  273 : : {  274 : 0 : errorString = "Format.Version";  275 : 0 : errorMsg(errorString);  276 : : }  277 : :   278 : 1119 : cnfFile\_.convSystem = readConfigStr("Conversion.System");  279 [ - + ]: 1119 : if (!cnfFile\_.convSystem)  280 : : {  281 : 0 : errorString = "Conversion.System";  282 : 0 : errorMsg(errorString);  283 : : }  284 : :   285 : 1119 : cnfFile\_.convMachine = readConfigStr("Conversion.Machine");  286 [ - + ]: 1119 : if (!cnfFile\_.convMachine)  287 : : {  288 : 0 : errorString = "Conversion.Machine";  289 : 0 : errorMsg(errorString);  290 : : }  291 : :   292 : 1119 : cnfFile\_.convMachineId = readConfigStr("Conversion.Machine.ID");  293 [ - + ]: 1119 : if (!cnfFile\_.convMachineId)  294 : : {  295 : 0 : errorString = "Conversion.Machine.ID";  296 : 0 : errorMsg(errorString);  297 : : }  298 : :   299 : 1119 : cnfFile\_.convMachineOS = readConfigStr("Conversion.Machine.OS");  300 [ - + ]: 1119 : if (!cnfFile\_.convMachineOS)  301 : : {  302 : 0 : errorString = "Conversion.Machine.OS";  303 : 0 : errorMsg(errorString);  304 : : }  305 : :   306 : 1119 : cnfFile\_.chipName = readConfigStr("Chip.Name");  307 [ - + ]: 1119 : if (!cnfFile\_.chipName)  308 : : {  309 : 0 : errorString = "Chip.Name";  310 : 0 : errorMsg(errorString);  311 : : }  312 : :   313 : 1119 : readConfigNum( "Chip.addressUnit.ref", cnfFile\_.chipAddrUnit);  314 [ - + ]: 1119 : if (0 >= cnfFile\_.chipAddrUnit)  315 : : {  316 : 0 : errorString = "Chip.addressUnit.ref";  317 : 0 : errorMsg(errorString);  318 : : }  319 : :   320 : 1119 : readConfigNum( "Chip.Size.x", cnfFile\_.chipSizeX);  321 [ - + ]: 1119 : if (0 >= cnfFile\_.chipSizeX)  322 : : {  323 : 0 : errorString = "Chip.Size.x";  324 : 0 : errorMsg(errorString);  325 : : }  326 : :   327 : 1119 : readConfigNum( "Chip.Size.y", cnfFile\_.chipSizeY);  328 [ - + ]: 1119 : if (0 >= cnfFile\_.chipSizeY)  329 : : {  330 : 0 : errorString = "Chip.Size.y";  331 : 0 : errorMsg(errorString);  332 : : }  333 : :   334 : 1119 : readConfigNum( "Chip.Block.Size.x", cnfFile\_.chipBSizeX);  335 [ - + ]: 1119 : if (0 >= cnfFile\_.chipBSizeX)  336 : : {  337 : 0 : errorString = "Chip.Block.Size.x";  338 : 0 : errorMsg(errorString);  339 : : }  340 : :   341 : 1119 : readConfigNum( "Chip.Block.Size.y", cnfFile\_.chipBSizeY);  342 [ - + ]: 1119 : if (0 >= cnfFile\_.chipBSizeY)  343 : : {  344 : 0 : errorString = "Chip.Block.Size.y";  345 : 0 : errorMsg(errorString);  346 : : }  347 : :   348 : 1119 : cnfFile\_.frmOrientation = readConfigStr("Frame.Orientation", true);  349 [ - + ]: 1119 : if ( !cnfFile\_.frmOrientation )  350 : : {  351 : 0 : delete [] cnfFile\_.frmOrientation;  352 : 0 : cnfFile\_.frmOrientation = new char[strlen("horizontal") + 1];  353 : 0 : strcpy(cnfFile\_.frmOrientation, "horizontal");  354 : : }  355 [ + + ][ - + ]: 1119 : else if( strcasecmp(cnfFile\_.frmOrientation, "horizontal")   356 : : && strcasecmp(cnfFile\_.frmOrientation, "vertical") )  357 : : {  358 : 0 : errorString = "Frame.Orientation";  359 : 0 : errorMsg(errorString);  360 : : }  361 : :   362 : 1119 : readConfigNum( "Chip.Frames", cnfFile\_.chipFrames);  363 [ - + ]: 1119 : if (0 >= cnfFile\_.chipFrames)  364 : : {  365 : 0 : errorString = "Chip.Frames";  366 : 0 : errorMsg(errorString);  367 : : }  368 : :   369 : 1119 : readConfigNum( "Frame.Height", cnfFile\_.frmHeight);  370 [ - + ]: 1119 : if (0 >= cnfFile\_.frmHeight)  371 : : {  372 : 0 : errorString = "Frame.Height";  373 : 0 : errorMsg(errorString);  374 : : }  375 : :   376 : 1119 : readConfigNum( "Frame.Width", cnfFile\_.frmWid );  377 [ - + ]: 1119 : if (0 >= cnfFile\_.frmWid)  378 : : {  379 : 0 : errorString = "Frame.Width";  380 : 0 : errorMsg(errorString);  381 : : }  382 : :   383 : 1119 : cnfFile\_.multiPlacement = readConfigStr("Cell.placementRule", true);  384 [ + - ]: 1119 : if ( !cnfFile\_.multiPlacement )  385 : : {  386 : 1119 : delete []cnfFile\_.multiPlacement;  387 : 1119 : cnfFile\_.multiPlacement = new char[strlen("noMultiple") + 1];  388 : 1119 : strcpy(cnfFile\_.multiPlacement, "noMultiple");  389 : : }  390 [ # # ][ # # ]: 0 : else if( strcasecmp(cnfFile\_.multiPlacement, "noMultiple")   391 : : && strcasecmp(cnfFile\_.multiPlacement, "multipleIDX") )  392 : : {  393 : 0 : errorString = "Cell.placementRule";  394 : 0 : errorMsg(errorString);  395 : : }  396 : :   397 : 1119 : readConfigNum( "Cell.maxSize.x", cnfFile\_.cellMaxSizeX);  398 [ - + ]: 1119 : if (0 >= cnfFile\_.cellMaxSizeX)  399 : : {  400 : 0 : errorString = "Cell.maxSize.x";  401 : 0 : errorMsg(errorString);  402 : : }  403 : :   404 : 1119 : readConfigNum( "Cell.maxSize.y", cnfFile\_.cellMaxSizeY);  405 [ - + ]: 1119 : if (0 >= cnfFile\_.cellMaxSizeY)  406 : : {  407 : 0 : errorString = "Cell.maxSize.y";  408 : 0 : errorMsg(errorString);  409 : : }  410 : :   411 : 1119 : readConfigNum( "Cell.Array.maxSize.x", cnfFile\_.cellArrMaxSizeX, 'Y');  412 [ - + ]: 1119 : if (0 == cnfFile\_.cellArrMaxSizeX)  413 : : {  414 : : }  415 [ # # ]: 0 : else if (0 > cnfFile\_.cellArrMaxSizeX)  416 : : {  417 : 0 : errorString = "Cell.Array.maxSize.x";  418 : 0 : errorMsg(errorString);  419 : : }  420 : :   421 : 1119 : readConfigNum( "Cell.Array.maxSize.y", cnfFile\_.cellArrMaxSizeY, 'Y');  422 [ - + ]: 1119 : if (0 == cnfFile\_.cellArrMaxSizeY)  423 : : {  424 : : }  425 [ # # ]: 0 : else if (0 > cnfFile\_.cellArrMaxSizeY)  426 : : {  427 : 0 : errorString = "Cell.Array.maxSize.y";  428 : 0 : errorMsg(errorString);  429 : : }  430 : :   431 : 1119 : cnfFile\_.patArrayCompress = readConfigStr("Pattern.arrayCompression");  432 [ - + ]: 1119 : if (!cnfFile\_.patArrayCompress)  433 : : {  434 : 0 : errorString = "Pattern.arrayCompression";  435 : 0 : errorMsg(errorString);  436 : : }  437 [ + - ][ + - ]: 1119 : else if (!strcmp(cnfFile\_.patArrayCompress, "on") ||  [ + - ]  438 : : !strcmp(cnfFile\_.patArrayCompress, "true") ||  439 : : !strcmp(cnfFile\_.patArrayCompress, "1"))  440 : : {  441 [ + - ]: 1119 : delete[] cnfFile\_.patArrayCompress;  442 : 1119 : cnfFile\_.patArrayCompress = new char[strlen("on") + 1];  443 : 1119 : strcpy(cnfFile\_.patArrayCompress, "on");   444 : : }  445 [ # # ][ # # ]: 0 : else if (!strcmp(cnfFile\_.patArrayCompress, "off") ||  [ # # ]  446 : : !strcmp(cnfFile\_.patArrayCompress, "false") ||  447 : : !strcmp(cnfFile\_.patArrayCompress, "0"))  448 : : {  449 [ # # ]: 0 : delete[] cnfFile\_.patArrayCompress;  450 : 0 : cnfFile\_.patArrayCompress = new char[strlen("off") + 1];  451 : 0 : strcpy(cnfFile\_.patArrayCompress, "off");   452 : : }  453 : : else  454 : : {  455 : 0 : errorString = "Pattern.arrayCompression";  456 : 0 : errorMsg(errorString);  457 : : }  458 : :   459 : : //Pattern Compression  460 : 1119 : cnfFile\_.patCompress = readConfigStr("Pattern.Compression");  461 [ - + ]: 1119 : if (!cnfFile\_.patCompress)  462 : : {  463 : 0 : errorString = "Pattern.Compression";  464 : 0 : errorMsg(errorString);  465 : : }  466 [ + - ][ + - ]: 1119 : else if (!strcmp(cnfFile\_.patCompress, "on") ||  [ + - ]  467 : : !strcmp(cnfFile\_.patCompress, "true") ||  468 : : !strcmp(cnfFile\_.patCompress, "1"))  469 : : {  470 [ + - ]: 1119 : delete[] cnfFile\_.patCompress;  471 : 1119 : cnfFile\_.patCompress = new char[strlen("on") + 1];  472 : 1119 : strcpy(cnfFile\_.patCompress, "on");  473 : : }  474 [ # # ][ # # ]: 0 : else if (!strcmp(cnfFile\_.patCompress, "off") ||  [ # # ]  475 : : !strcmp(cnfFile\_.patCompress, "false") ||  476 : : !strcmp(cnfFile\_.patCompress, "0"))  477 : : {  478 [ # # ]: 0 : delete[] cnfFile\_.patCompress;  479 : 0 : cnfFile\_.patCompress = new char[strlen("off") + 1];  480 : 0 : strcpy(cnfFile\_.patCompress, "off");  481 : : }  482 : : else  483 : : {  484 : 0 : errorString = "Pattern.Compression";  485 : 0 : errorMsg(errorString);  486 : : }  487 : :   488 : :   489 : 1119 : readConfigNum( "Pattern.Set", cnfFile\_.patSet);  490 [ - + ]: 1119 : if (0 > cnfFile\_.patSet || 1 < cnfFile\_.patSet)  491 : : {  492 : 0 : errorString = "Pattern.Set";  493 : 0 : errorMsg(errorString);  494 : : }  495 : :   496 : 1119 : readConfigNum( "MaxData.ref", cnfFile\_.maxDataRef , true);  497 [ - + ]: 1119 : if (0 > cnfFile\_.maxDataRef)  498 : : {  499 : 0 : errorString = "MaxData.ref";  500 : 0 : errorMsg(errorString);  501 : : }  502 : :   503 : 1119 : readConfigNum( "MaxData.cell", cnfFile\_.maxDataCell, true);  504 [ - + ]: 1119 : if (0 > cnfFile\_.maxDataCell)  505 : : {  506 : 0 : errorString = "MaxData.cell";  507 : 0 : errorMsg(errorString);  508 : : }  509 : :   510 : 1119 : readConfigNum( "MaxData.cell.common", cnfFile\_.maxDataCommon);  511 [ - + ]: 1119 : if (0 > cnfFile\_.maxDataCommon)  512 : : {  513 : 0 : errorString = "MaxData.cell.common";  514 : 0 : errorMsg(errorString);  515 : : }  516 : :   517 : 1119 : readConfigNum( "MaxData.Frame", cnfFile\_.maxDataFrame, true);  518 [ - + ]: 1119 : if (0 > cnfFile\_.maxDataFrame)  519 : : {  520 : 0 : errorString = "MaxData.Frame";  521 : 0 : errorMsg(errorString);  522 : : }  523 : :   524 : 1119 : readConfigNum( "MaxNum.cellLocation", cnfFile\_.maxNumCellLoc, true);  525 : 1119 : readConfigNum( "Frame.Translate.y", cnfFile\_.frameTranslateY, true);  526 [ - + ]: 1119 : if (0 < cnfFile\_.frameTranslateY)  527 : : {  528 : 0 : errorString = "Frame.Translate.y";  529 : 0 : errorMsg(errorString);  530 : : }  531 : :   532 : 1119 : destroyConfigBuf();  533 : :   534 : : uint32\_t maxFrmCols = static\_cast<uint32\_t>(   535 : : ceil( static\_cast<double>(cnfFile\_.chipSizeX)  536 [ + - ]: 1119 : / static\_cast<double>(cnfFile\_.chipBSizeX)));  537 : :   538 : 1119 : createFrmMatrix(cnfFile\_.chipFrames, maxFrmCols);  539 : :   540 : 1125 : return;  541 : : }  542 : :   543 : : void ChipCnfParser::parseFrameInfo()  544 : 8979 : {  545 : 8979 : if( sChipCnfPath\_.empty() )  546 : 8979 : return; // FIXME::EXCEPTION(ASSERTION)  547 : 8979 : std::string errorString;  548 : :   549 : 8979 : char linebuf[BUFSIZ];  550 [ + + ]: 8979 : if( cnfFile\_.pMatrix\_ )  551 : 7860 : return;  552 : :   553 : 1119 : uiNumLinesInIchipCnf\_ = 0;  554 : 1119 : try {  555 : 1119 : uiNumLinesInIchipCnf\_ = createConfigBuf();  556 : : }  557 [ # # ]: 0 : catch( DPL::Error \*pChild )  558 : : {  559 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  560 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  561 : : ErrorHandler::eErrFunction,  562 : : pErr->getErrorMsg(  563 : : ErrorHandler::eErrFunction,  564 : : "ChipCnfParser::createConfigBuf"));  565 : : }  566 : :   567 [ + - ][ + + ]: 2247 : for (int j = 0; j < cnfFile\_.chipFrames; j++)  568 : : {  569 : 1128 : sprintf(pMatrix\_->getFrameInfo(0,j)->name, "%d", j+1) ;  570 : :   571 : 1128 : pMatrix\_->getFrameInfo(0,j)->isNull = false ;  572 : 1128 : pMatrix\_->getFrameInfo(0,j)->isSelected = false ;  573 : :   574 : 1128 : sprintf(linebuf, "Frame.%d.Height", j+1) ;  575 : 1128 : readConfigNum( linebuf, pMatrix\_->getFrameInfo(0,j)->height, true);  576 : :   577 [ + - ]: 1128 : if (0 == pMatrix\_->getFrameInfo(0,j)->height)  578 : 1128 : pMatrix\_->getFrameInfo(0,j)->height = cnfFile\_.frmHeight;  579 : :   580 : 1128 : pMatrix\_->getFrameInfo(0,j)->vsbHeight   581 : : = pMatrix\_->getFrameInfo(0,j)->height \* cnfFile\_.chipBSizeY ;  582 : :   583 : 1128 : pMatrix\_->getFrameInfo(0,j)->wid = cnfFile\_.frmWid;  584 : 1128 : pMatrix\_->getFrameInfo(0,j)->vsbWid   585 : : = pMatrix\_->getFrameInfo(0,j)->wid \* cnfFile\_.chipBSizeX ;  586 : :   587 : 1128 : pMatrix\_->getFrameInfo(0,j)->frmLocX = 0 ;  588 : 1128 : pMatrix\_->getFrameInfo(0,j)->vbgIdx = 0 ;  589 : :   590 [ + + ]: 1128 : if( 0 == j)  591 : : {  592 : 1119 : pMatrix\_->getFrameInfo(0,j)->frmLocY = 0 ;  593 : 1119 : pMatrix\_->getFrameInfo(0,j)->vbgIdy = 0 ;  594 : : }  595 : : else  596 : : {  597 : 9 : pMatrix\_->getFrameInfo(0,j)->frmLocY   598 : : = pMatrix\_->getFrameInfo(0,j-1)->frmLocY   599 : : + pMatrix\_->getFrameInfo(0,j-1)->vsbHeight ;  600 : 9 : pMatrix\_->getFrameInfo(0,j)->vbgIdy   601 : : = pMatrix\_->getFrameInfo(0,j-1)->vbgIdy   602 : : + pMatrix\_->getFrameInfo(0,j-1)->height ;  603 : : }  604 : :   605 : 1128 : sprintf(linebuf, "Frame.%d.subFrames", j+1);  606 : 1128 : readConfigNum( linebuf, pMatrix\_->getFrameInfo(0,j)->subFrms,true);  607 : 1128 : if (pMatrix\_->getFrameInfo(0,j)->subFrms >   608 : : pMatrix\_->get\_numCols\_())  609 : : {  610 : 0 : errorString = linebuf;  611 : 0 : errorMsg(errorString);  612 : : }  613 : 1128 : int32\_t subFrmWid = 0;   614 : 1128 : sprintf(linebuf, "Frame.%d.subFrame.Width", j+1);  615 : 1128 : readConfigNum( linebuf, subFrmWid, true);  616 [ - + ]: 1128 : if ( 0 > subFrmWid )  617 : : {  618 : 0 : errorString = linebuf;  619 : 0 : errorMsg(errorString);  620 : : }  621 : :   622 [ + - ]: 1128 : if ( 0 == subFrmWid )   623 : 1128 : subFrmWid = cnfFile\_.frmWid;  624 : :   625 [ - + ]: 1128 : for ( int i = 0; i < (int)pMatrix\_->getFrameInfo(0,j)->subFrms ; i++)   626 : : {  627 : 0 : sprintf(pMatrix\_->getFrameInfo(i,j)->name, "%d.%d", j+1, i+1) ;  628 : :   629 : 0 : pMatrix\_->getFrameInfo(i,j)->isNull = false ;  630 : 0 : pMatrix\_->getFrameInfo(i,j)->isSelected = false ;  631 : :   632 : 0 : sprintf(linebuf, "Frame.%d.subFrame.%d.Width", j+1, i+1);  633 : 0 : readConfigNum( linebuf, pMatrix\_->getFrameInfo(i,j)->wid, true);  634 [ # # ]: 0 : if (0 == pMatrix\_->getFrameInfo(i,j)->wid)  635 : 0 : pMatrix\_->getFrameInfo(i,j)->wid = subFrmWid ;  636 : :   637 : 0 : pMatrix\_->getFrameInfo(i,j)->vsbWid   638 : : = pMatrix\_->getFrameInfo(i,j)->wid \* cnfFile\_.chipBSizeX ;  639 : :   640 : 0 : pMatrix\_->getFrameInfo(i,j)->height   641 : : = pMatrix\_->getFrameInfo(0,j)->height;  642 : :   643 : 0 : pMatrix\_->getFrameInfo(i,j)->vsbHeight   644 : : = pMatrix\_->getFrameInfo(0,j)->vsbHeight;  645 : :   646 [ # # ]: 0 : if( 0 == i)  647 : : {  648 : 0 : pMatrix\_->getFrameInfo(i,j)->frmLocX = 0 ;  649 : 0 : pMatrix\_->getFrameInfo(i,j)->vbgIdx = 0 ;  650 : : }  651 : : else  652 : : {  653 : 0 : pMatrix\_->getFrameInfo(i,j)->frmLocX   654 : : = pMatrix\_->getFrameInfo(i-1,j)->frmLocX   655 : : + pMatrix\_->getFrameInfo(i-1,j)->vsbWid ;  656 : 0 : pMatrix\_->getFrameInfo(i,j)->vbgIdx   657 : : = pMatrix\_->getFrameInfo(i-1,j)->vbgIdx   658 : : + pMatrix\_->getFrameInfo(i-1,j)->wid ;  659 : : }  660 : :   661 : 0 : pMatrix\_->getFrameInfo(i,j)->frmLocY   662 : : = pMatrix\_->getFrameInfo(0,j)->frmLocY ;  663 : 0 : pMatrix\_->getFrameInfo(i,j)->vbgIdy   664 : : = pMatrix\_->getFrameInfo(0,j)->vbgIdy ;  665 : : }  666 : : }  667 : 1119 : destroyConfigBuf();  668 : 1119 : cnfFile\_.pMatrix\_ = pMatrix\_;  669 : 1119 : return;  670 : :   671 : : //error\_out:   672 : : // destroyConfigBuf();  673 : : // ostringstream sout;  674 : : // sout<<"[ERROR] Value of \""<<errorString  675 : : // <<"\" is out of range in ichip.cnf"  676 : : // <<std::endl;  677 : : // sprintf(linebuf, "Value of %s is out of range in ichip.cnf", errorString.c\_str());  678 : : // ErrorHandler \*pErr = ErrorHandler::instance();  679 : : // pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  680 : : // ,ErrorHandler::eErrMessageString  681 : : // ,pErr->getErrorMsg((ErrorHandler::eErrMessageString)  682 : : // ,linebuf));  683 : : }  684 : :   685 : :   686 : : void ChipCnfParser::cleanUpFrameMatrix()  687 : 0 : {  688 : 0 : pMatrix\_->releaseMatrixMemory();  689 : 0 : cnfFile\_.pMatrix\_ = NULL;  690 : : }  691 : :   692 : : void ChipCnfParser::set\_sChipCnfPath\_(const std::string &value)  693 : 1125 : {  694 : 1125 : sChipCnfPath\_ = value;  695 : : }  696 : :   697 : 1215 : } //namespace nft |

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