| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibVsb/src](http://docs.google.com/index.html) - VSBDataProcessor.cpp (source / [functions](http://docs.google.com/VSBDataProcessor.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 84 | 100 | 84.0 % | | Date: | 2012-03-23 |  | Functions: | 9 | 9 | 100.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 16 | 26 | 61.5 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : #include "Sgs\_Define.h"  2 : : #include "dec\_math.h"  3 : : #include "Logger.h"  4 : : #include "ErrorHandler.h"  5 : : #include "VSBDataProcessor.h"  6 : : #include "vchip.h"  7 : : #include "VSBGenericCellReader.h"  8 : : #include "HealthCheckCounter.h"  9 : :   10 : : namespace nft{  11 : :   12 : 4376 : VSBDataProcessor::VSBDataProcessor(){  13 : 1094 : mRefReader = NULL;  14 : 1094 : mCellCommonReader = NULL;  15 : : }  16 : :   17 : 2188 : VSBDataProcessor::~VSBDataProcessor(){  18 : 1094 : delete mRefReader;  19 : 1094 : delete mCellCommonReader;  20 : 1094 : delete mLinkReader;  21 [ + - ][ # # ]: 1094 : delete mGenericCellReader;   22 : : }  23 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  24 : : \* Function Name: initialize  25 : : \* Purpose: Initialize the different readers  26 : : \* Parameters: const MemInfo &cellCommonBuf  27 : : , const MemInfo &cellBuf  28 : : , const MemInfo &refBuf  29 : : , const MemInfo &linkBuf  30 : : \* Return Values: void  31 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  32 : : void   33 : : VSBDataProcessor::initialize(  34 : : const MemInfo &cellCommonBuf  35 : : , const MemInfo &cellBuf  36 : : , const MemInfo &refBuf  37 : 1094 : , const MemInfo &linkBuf){  38 : :   39 : 1094 : char cellCommonFileName[1024];  40 : :   41 : : //Initialize the cell common reader  42 : 1094 : if (cellCommonBuf.getMem()) //cell.common exists  43 : : {  44 : 0 : mCellCommonReader = new VSBCellReader(cellCommonBuf.getMem());  45 : : }  46 : :   47 : : const DPMANAGER\_MESSAGE\_CONVERTER \*startMsg  48 : 1094 : = ConverterParams::instance()->getStartupMessage();  49 : :   50 : : //Get VSB AU in femtometer  51 : : //TODO::NFT\_CHECK uint64\_t vsbAU\_inFm = pChipCnf->get\_ulChipAuRef\_();  52 : 1094 : uint32\_t blkSizeX = startMsg->ullVchipBSizeX;  53 : 1094 : uint32\_t blkSizeY = startMsg->ullVchipBSizeY;  54 : :   55 : 1094 : uint32\_t blkCountX = startMsg->ullDpbWidth/blkSizeX;  56 : 1094 : uint32\_t blkCountY = startMsg->ullDpbHeight/blkSizeY;  57 : :   58 : 1094 : UIntPosition frameLoc;  59 : :   60 : 1094 : unsigned short orientation = ConverterParams::instance()->getStartupMessage()->usFrameOrientation;  61 : : // orientation = 0; VERTICAL   62 : : // orientation = 1; HORIZONTAL   63 : :   64 : : //Process a frame  65 : : //Get the different frame processors  66 : 1094 : VSBCellRRIDTracker cellTracker;  67 : : //#ifdef ERROR\_SEED //<<Test\_Id:VSBDataProcessor\_initialize\_ref>>  68 : : // char\* value\_ref = refBuf.getMem();  69 : : // value\_ref = NULL;  70 : 1094 : if (NULL == refBuf.getMem())  71 : : {  72 : : //#endif  73 : 0 : Logger::instance()->log(LogMessageTable::eDATA\_READER\_NO\_FILE,  74 : : "ref");  75 : :   76 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  77 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  78 : : ,ErrorHandler::eErrFileNotFound  79 : : ,pErr->getErrorMsg((ErrorHandler::eErrFileNotFound),"ref"));  80 : : }   81 : : //#ifdef ERROR\_SEED //<<Test\_Id:VSBDataProcessor\_initialize\_link>>  82 : : // char\* value\_link = linkBuf.getMem();  83 : : // value\_link = NULL;  84 : 1094 : if (NULL == linkBuf.getMem())  85 : : {  86 : : //#endif  87 : 0 : Logger::instance()->log(LogMessageTable::eDATA\_READER\_NO\_FILE,  88 : : "link");  89 : :   90 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  91 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  92 : : ,ErrorHandler::eErrFileNotFound  93 : : ,pErr->getErrorMsg((ErrorHandler::eErrFileNotFound),"link"));  94 : : }  95 : :   96 : : //At least one of cell.n and cell.common must exist  97 : : //#ifdef ERROR\_SEED //<<Test\_Id:VSBDataProcessor\_initialize\_cell>>  98 : : // char\* value\_cell = cellBuf.getMem();  99 : : // char\* value\_cell\_common = cellCommonBuf.getMem();  100 : : // value\_cell = NULL;  101 : : // value\_cell\_common = NULL;  102 : 1094 : if (NULL == cellBuf.getMem() && NULL == cellCommonBuf.getMem())  103 : : {  104 : : //#endif  105 : 0 : Logger::instance()->log(LogMessageTable::eDATA\_READER\_NO\_FILE,  106 : : "cell");  107 : :   108 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  109 : 0 : pErr->errorAdd(NULL, (char\*)\_\_FILE\_\_, (char\*)\_\_FUNCTION\_\_, (int)\_\_LINE\_\_  110 : : ,ErrorHandler::eErrFileNotFound  111 : : ,pErr->getErrorMsg((ErrorHandler::eErrFileNotFound),"cell"));  112 : : }   113 : :   114 : : //Now initialize the Ref Reader  115 : 1094 : mRefReader = new VSBRefReader(refBuf.getMem(), blkCountX, blkCountY, blkSizeX,  116 : : blkSizeY, orientation);  117 : :   118 : : //Get the link readers and the cell  119 : 1094 : mLinkReader = new VSBLinkReader(linkBuf.getMem());  120 : :   121 : : //Now initialize the Generic cell Reader  122 : 1094 : mGenericCellReader = new VSBGenericCellReader(cellBuf.getMem()  123 : : , mCellCommonReader  124 : : , mLinkReader  125 : : , &mCellTracker  126 : : , ConverterParams::instance()->getStartupMessage()->uiAU);  127 : : //, static\_cast<uint32\_t>(1000000000 \* 0.001)); //input in um output converted in fm  128 : :   129 : : //Prepare the cell tracker that tracks the reference cells  130 : : //TODO Not required I think  131 : 1094 : mCellTracker.reInit();  132 : :   133 : : //Identify the reference cells  134 : 1094 : mRefReader->refNearReferenceTableCreate(mCellTracker);  135 : : }  136 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  137 : : \* Function Name: setBlockIDs  138 : : \* Purpose: Return the cell context for the given VSB block  139 : : \* Parameters:   140 : : , uint32\_t bid\_x  141 : : , uint32\_t bid\_y  142 : : \* Return Values: void  143 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  144 : : void   145 : : VSBDataProcessor::setBlockIDs(  146 : : uint32\_t bid\_x,   147 : : uint32\_t bid\_y)  148 : 1127 : {  149 : 1127 : mBlockID\_x = bid\_x;  150 : 1127 : mBlockID\_y = bid\_y;  151 : 1127 : mCellContext = mRefReader->refGetBlock(&mCurrentContext, mBlockID\_x, mBlockID\_y, &mCurrentBlock);  152 : : }  153 : :   154 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  155 : : \* Function Name: getNextCell  156 : : \* Purpose: Return next VSB cell  157 : : \* If false then no more blocks  158 : : \* Parameters:   159 : : Cell \*cell  160 : : , uint32\_t bid\_x  161 : : , uint32\_t bid\_y  162 : : \* Return Values: bool  163 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  164 : : bool  165 : : VSBDataProcessor::getNextCell(  166 : : Cell \*cell)throw (DPL::Error \*)  167 : 2386 : {  168 : 2386 : HealthCheckCounter::instance()->incrCounter(  169 : : HealthCheckCounter::eUpdateFrequent |  170 : : HealthCheckCounter::eUpdateInfrequent);  171 : 2386 : bool status = false;  172 : :   173 [ + + ]: 2386 : if (NULL == mCellContext)  174 : : {  175 : : //No more blocks  176 : 14 : return false;  177 : : }  178 : :   179 : : IntPosition blockLoc(mRefReader->refGetBlockSizeX() \* mBlockID\_x,  180 : 2372 : mRefReader->refGetBlockSizeY() \* mBlockID\_y);  181 : :   182 : 2372 : try{  183 : 2372 : mGenericCellReader->cellGetNext(mCellContext, cell);  184 : : }  185 [ # # ]: 0 : catch(DPL::Error \*pChild)  186 : : {  187 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  188 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  189 : : ErrorHandler::eErrFunction,  190 : : pErr->getErrorMsg( ErrorHandler::eErrFunction,  191 : : "VSBDataProcessor::getNextCell"));  192 : : }  193 : :   194 [ + + ]: 2372 : if( !cell->pgc )   195 : 1111 : return false; //No more cell left in the block  196 : :   197 : : //Cell is defined wrt block, pattern is defined wrt to cell.  198 : : //Translate them  199 : 1261 : cell->startPos = cell->getCellStartPosition() + blockLoc;  200 : : #if 0  201 : : //TODO Address unit convertion  202 : : cellLoc.setX(vsbToPDGAU(cellLoc.getX(), vsbAU\_inFm));  203 : : cellLoc.setY(vsbToPDGAU(cellLoc.getY(), vsbAU\_inFm));  204 : : //Handling for mirroring   205 : : #endif  206 [ # # ]: 2386 : return true;  207 : : }  208 : :   209 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  210 : : \* Function Name: getNextFlattenedCell  211 : : \* Purpose: Flatten array cell into single cells of "Cell" type  212 : : \* Parameters: Cell inCell  213 : : , Cell \*outCell  214 : : , LongPosition inBlock, LongPosition \*outBlock  215 : : , FlattenedContext \*flattenContext  216 : : \* Return Values: bool  217 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  218 : : bool  219 : : VSBDataProcessor::getNextFlattenedCell(  220 : : Cell &inCell  221 : : , Cell \*outCell  222 : : , IntPosition &inBlockPos, IntPosition &outBlockPos  223 : : , FlattenedContext \*flattenContext)  224 : 215 : {  225 : : //Returns false if there is no cell to flatten.  226 : 215 : uint32\_t rep\_x=0, rep\_y=0, flattenedCellX=0, flattenedCellY=0;  227 : 215 : double pitchX ; // pitch of a cell in the array along x  228 : 215 : double pitchY ; // pitch of a cell in the array along y  229 : 215 : int32\_t span\_x, span\_y;  230 : 215 : unsigned long long blockSizeX = ConverterParams::instance()->getStartupMessage()->ullVchipBSizeX;  231 : 215 : unsigned long long blockSizeY = ConverterParams::instance()->getStartupMessage()->ullVchipBSizeY;  232 : :   233 : 215 : if(!(flattenContext->getIsMoreCell())){  234 : 54 : return false;  235 : : }  236 : : // The input cell is a single cell.  237 [ - + ]: 161 : else if ((inCell.RefType == SINGLE\_VALUE) || (inCell.RefType == SINGLE\_NEAR )){  238 : : // The input and out Cells are the same  239 : 0 : \*outCell = inCell;  240 : 0 : flattenContext->setIsMoreCell(false);  241 : 0 : return true;  242 : : }  243 : : else{ //Array Cell  244 : 161 : rep\_x = inCell.ac.xr;  245 : 161 : rep\_y = inCell.ac.yr;  246 : :   247 [ + + ][ + - ]: 161 : pitchX = (rep\_x -1) ? (double)(inCell.ac.xe - inCell.ac.xb)/(double)(rep\_x-1) : 0;  248 [ + + ][ + - ]: 161 : pitchY = (rep\_y -1) ? (double)(inCell.ac.ye - inCell.ac.yb)/(double)(rep\_y-1) : 0;  249 : :   250 : 161 : flattenedCellX = flattenContext->getNumCellX();  251 : 161 : flattenedCellY = flattenContext->getNumCellY();  252 : :   253 : 161 : span\_x = (inBlockPos.getX() \* blockSizeX + inCell.ac.xb +  254 : : + SgsRoundINT(flattenedCellX \* pitchX));  255 : 161 : outBlockPos.setX(span\_x /((int)blockSizeX));  256 : :   257 : 161 : span\_y = (inBlockPos.getY() \* blockSizeY + inCell.ac.yb   258 : : + SgsRoundINT(flattenedCellY \* pitchY));  259 : 161 : outBlockPos.setY(span\_y /((int) blockSizeY));  260 : :   261 : 161 : outCell->sc.dx = inCell.ac.dx;  262 : 161 : outCell->sc.dy = inCell.ac.dy;  263 : 161 : outCell->sc.xb = span\_x % ((int)blockSizeX);  264 : 161 : outCell->sc.yb = span\_y % ((int)blockSizeY);  265 : 161 : outCell->pgc = new GroupContext();  266 : 161 : \*(outCell->pgc) = \*(inCell.pgc);  267 : 161 : outCell->PlacementType = inCell.PlacementType;  268 : 161 : outCell->RefType = inCell.RefType;  269 : 161 : outCell->refid = inCell.refid;  270 : 161 : outCell->mSegmentDataSize = inCell.mSegmentDataSize;  271 : 161 : outCell->mTargetThreadID = inCell.mTargetThreadID;  272 : 161 : outCell->startPos  273 : : = IntPosition(  274 : : outCell->sc.xb + outBlockPos.getX() \* blockSizeX,  275 : : outCell->sc.yb + outBlockPos.getY() \* blockSizeY);  276 : 161 : flattenContext->setNumCellY(++flattenedCellY);  277 [ + + ]: 161 : if(flattenedCellY == rep\_y){  278 : 108 : flattenContext->setNumCellX(++flattenedCellX);  279 : 108 : flattenContext->setNumCellY(0);  280 : : }  281 [ + + ]: 161 : if(flattenedCellX == rep\_x)  282 : 54 : flattenContext->setIsMoreCell(false);  283 : : }  284 : 215 : return true;  285 : : }  286 : :   287 : 1215 : } //namespace nft |

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