| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [ArbitraryModeFlow/src](http://docs.google.com/index.html) - ArbitraryModeShotDistributor.cpp (source / [functions](http://docs.google.com/ArbitraryModeShotDistributor.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 126 | 166 | 75.9 % | | Date: | 2012-03-23 |  | Functions: | 8 | 8 | 100.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 27 | 52 | 51.9 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : #include "Sgs\_Define.h"  2 : :   3 : : #include "ArbitraryModeShotDistributor.h"  4 : : #include "ConverterParams.h"  5 : : #include "DXPCalculator.h"  6 : : #include "print.h"  7 : : #include "SFTable.h"  8 : : #include "SFTableAccessor.h"  9 : : #include "SimplePattern.h"  10 : : #include "SFLayer.h"  11 : : #include "TFInfo.h"  12 : : #include "UtilTypes.h"  13 : : #include "SDDB.h"  14 : :   15 : : namespace nft {  16 : :   17 : : ArbitraryModeShotDistributor::ArbitraryModeShotDistributor(  18 : : const ShotTimeCalculator & inShotTimeCalc)  19 : 46 : : mLastAddedTF(),  20 : : mTargetContainer(),  21 : : mShotTimeCalc(inShotTimeCalc),  22 : : mGMCCorrector(),  23 : : mSwingBackApplier(),  24 : 46 : mParams( ConverterParams::instance() ) {  25 : 46 : mLastAddedTF = new TF::Pointer[ mParams  26 : : ->getStartupMessage()->uiSFLCountInDpb ];  27 : : }  28 : :   29 : : void  30 : : ArbitraryModeShotDistributor::setTargetCellContainer(  31 : 63 : ArbitraryModeCellContainer::Pointer inContainer ) {  32 : 63 : mTargetContainer = inContainer;  33 : : }  34 : :   35 : 92 : ArbitraryModeShotDistributor::~ArbitraryModeShotDistributor() {  36 [ # # ][ + - ]: 46 : delete [] mLastAddedTF;  37 : : }  38 : :   39 : : void  40 : 63 : ArbitraryModeShotDistributor::reset() {  41 : 63 : size\_t numSFL = mParams->getStartupMessage()->uiSFLCountInDpb;  42 [ + - ][ + + ]: 83 : for( size\_t i = 0; i < numSFL; i++ )  43 : : {  44 : 83 : mLastAddedTF[ i ] = 0;  45 : : }  46 : : }  47 : :   48 : : //ADd's the shot after calculation into TF  49 : : void  50 : : ArbitraryModeShotDistributor::operator()(  51 : : const SimplePattern & inPat,  52 : : const PatternPosition & inPosition,  53 : : uint32\_t inSFLNo,  54 : : ThreadID inThreadID,  55 : 322 : uint32\_t inAI) throw(DPL::Error \*) {  56 : :   57 : 322 : SDDB \*pSDDB = SDDB::instance();  58 : : #ifndef KEEP\_ZERO\_SIZE\_SHOT  59 : 322 : if( 0 == inPat.getSizeX() || 0 == inPat.getSizeY() )  60 : : {  61 : 17 : try  62 : : {  63 : 17 : pSDDB->shotDiscarded();  64 : : }  65 [ # # ]: 0 : catch(DPL::Error \*pChild)  66 : : {  67 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  68 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_,  69 : : \_\_LINE\_\_, ErrorHandler::eErrFunction,  70 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  71 : : "SDDB::shotDiscarded"));  72 : : }  73 : : //Zero pattern removed  74 : 0 : return;  75 : : }  76 : : #endif //KEEP\_ZERO\_SIZE\_SHOT  77 : 305 : try  78 : : {  79 : 305 : pSDDB->SimplePatternAfterBBInclusion(inPat, inPosition);  80 : : }  81 [ # # ]: 0 : catch(DPL::Error \*pChild)  82 : : {  83 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  84 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_,  85 : : \_\_LINE\_\_, ErrorHandler::eErrFunction,  86 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  87 : : "SDDB::SimplePatternAfterBBInclusion"));  88 : : }  89 : :   90 : 305 : ConverterParams \*pParams = ConverterParams::instance();  91 : : const DPMANAGER\_MESSAGE\_SFL\_INFO \*pLayer  92 : 305 : =pParams->getLayerParams() + inSFLNo;  93 : : const DPMANAGER\_MESSAGE\_CONVERTER \*pStartMsg  94 : 305 : =pParams->getStartupMessage();  95 : 305 : const sgs\_column\_t \*pSecondParams = pParams->getColumnParams();  96 : : ProcessingMode mode = static\_cast<ProcessingMode>(  97 : 305 : pStartMsg->usProcessingMode & 0x03);  98 : 305 : TF::Pointer targetTF = 0;  99 : : //Check whether shot belongs to last TF  100 : : LongPosition patPosWrtCell(  101 : : inPosition.getX()  102 : : - mTargetContainer->getTargetCell()->startPos.getX(),  103 : : inPosition.getY()  104 : 305 : - mTargetContainer->getTargetCell()->startPos.getY());  105 : 305 : int64\_t tfSizeX = pStartMsg->iTFSizeX;  106 : 305 : int64\_t tfSizeY = pStartMsg->iTFSizeY;  107 : : int64\_t numCols = ( mTargetContainer->getTargetCell()->sc.dx  108 : 305 : - pLayer->llTFOriginX ) / tfSizeX;  109 : : int64\_t numRows = ( mTargetContainer->getTargetCell()->sc.dy  110 : 305 : - pLayer->llTFOriginY ) / tfSizeY;  111 : 305 : bool shotAddedToLastTF = false;  112 : 305 : TF::Pointer lastAddedTF = mLastAddedTF[ inSFLNo ];  113 [ + + ]: 305 : if( eProcessArbitraryTFLess != mode  114 : : && lastAddedTF  115 : : && inPat.getCode() == lastAddedTF->getPatCode()  116 : : && lastAddedTF->getParentSF()->getSFLNo() == inSFLNo )  117 : : {  118 : : //Check belonginness  119 : : int64\_t lastTFX = pLayer->llTFOriginX  120 : 145 : + ( lastAddedTF->getColumn() \* tfSizeX );  121 : : int64\_t lastTFY = pLayer->llTFOriginY  122 : 145 : + ( lastAddedTF->getRow() \* tfSizeY );  123 : 145 : int64\_t offsetX = patPosWrtCell.getX() - lastTFX;  124 : 145 : int64\_t offsetY = patPosWrtCell.getY() - lastTFY;  125 [ + + ][ + + ]: 145 : if( (offsetX >= 0 && offsetX < tfSizeX )  126 : : && (offsetY >= 0 && offsetY < tfSizeY ))  127 : : {  128 : 77 : e9k\_debug("Pattern will be added to last TF at %lld, %lld\n",  129 : : lastTFX, lastTFY);  130 : 77 : targetTF = lastAddedTF;  131 : 77 : shotAddedToLastTF = true;  132 : : }  133 : : }  134 : 305 : bool tfLessMode = pStartMsg->usProcessingMode & 0x02;  135 [ + + ]: 305 : if( ! shotAddedToLastTF )  136 : : {  137 : 228 : int64\_t tfPositionX = 0;  138 : 228 : int64\_t tfPositionY = 0;  139 : 228 : int64\_t normProcRow = 0;  140 : 228 : int64\_t normProcCol = 0;  141 [ + + ]: 228 : if( !tfLessMode )  142 : : {  143 : : //1. Calculate and check TF indices  144 : 196 : int64\_t col = (patPosWrtCell.getX() - pLayer->llTFOriginX)/tfSizeX;  145 : 196 : int64\_t row = (patPosWrtCell.getY() - pLayer->llTFOriginY)/tfSizeY;  146 : 196 : normProcRow = row;  147 : 196 : normProcCol = col;  148 : 196 : tfPositionX = pLayer->llTFOriginX + col \* tfSizeX + (tfSizeX >> 1);  149 : 196 : tfPositionY = pLayer->llTFOriginY + row \* tfSizeY + (tfSizeY >> 1);  150 : : }  151 : : else  152 : : {  153 : 32 : tfPositionX = patPosWrtCell.getX();  154 : 32 : tfPositionY = patPosWrtCell.getY();  155 : : }  156 : :   157 : 228 : LongPosition tfPosition( tfPositionX, tfPositionY );  158 : :   159 : : //2. Get SF where this is to be added  160 : 228 : SFTable::Pointer sfTable = mTargetContainer->getSFT();  161 : 228 : SFTableAccessor accessor( sfTable, pParams );  162 : 228 : SFGroup::Pointer sfg = accessor.getSFGroup( tfPosition, inSFLNo );  163 : 228 : SubField::Pointer sf = accessor.getSubField( tfPosition, inSFLNo );  164 [ - + ]: 228 : if( NULL == sfg )  165 : : {  166 : 0 : Logger::instance()->log(LogMessageTable::eTF\_OUSIDE\_SF\_TABLE,  167 : : tfPositionX, tfPositionY, inSFLNo );  168 : 0 : e9k\_debug("TF at %lld, %lld is outside SF Table for layer %d\n",  169 : : tfPositionX, tfPositionY, inSFLNo );  170 : 0 : return;  171 : : }  172 : 228 : if( !sf->getIsGmcDone() )  173 : : {  174 : 83 : if( sfg->getIsEmpty() )  175 : : {  176 : 83 : sfg->setIsEmpty( false );  177 : : }  178 : : LongPosition sfAbsPosition(  179 : : sf->getXCoordinate() + pStartMsg->llVchipOriginX  180 : : + mTargetContainer->getTargetCell()->startPos.getX()  181 : : + (pStartMsg->iSFSizeX >> 1),  182 : : sf->getYCoordinate() + pStartMsg->llVchipOriginY  183 : : + mTargetContainer->getTargetCell()->startPos.getY()  184 : 83 : + (pStartMsg->iSFSizeY >> 1));  185 : : //1. Calculate DXP value and store it for future reference  186 [ + - ]: 83 : if( pSecondParams->kbr.mode )  187 : : {  188 : : int16\_t dxpValue  189 : : = DXPCalculator::calculateDxpValue(  190 : : sfAbsPosition,  191 : : mParams->getKBRMap(),  192 : 83 : pSecondParams->kbr );  193 : 83 : size\_t tmp = dxpValue;  194 : 83 : Logger::instance()->log(LogMessageTable::eSHOT\_KBR\_VALUE,  195 : : dxpValue);  196 : 83 : sf->setDxpValue( dxpValue );  197 : : }  198 : : //2. IF GMC is enabled for SF, perform it  199 [ + - ]: 83 : if( !( pSecondParams->gmc.mode & 0x00000004 ) )  200 : : {  201 : 83 : const GMCMap \*pGMCMap = pParams->getGMCMap();  202 : 83 : DoublePosition corrPos;  203 : 83 : bool gmcDone = false;  204 [ + + ]: 83 : if( pSecondParams->gmc.mode & 0x00000001 )  205 : : {  206 : 2 : corrPos += mGMCCorrector.calculate( sfAbsPosition,  207 : : pGMCMap, pSecondParams->gmc,  208 : : GMCorrector::eGMCPolynomial );  209 : 2 : gmcDone = true;  210 : : }  211 [ + + ]: 83 : if( pSecondParams->gmc.mode & 0x00000002 )  212 : : {  213 : 2 : corrPos += mGMCCorrector.calculate( sfAbsPosition,  214 : : pGMCMap, pSecondParams->gmc,  215 : : GMCorrector::eGMCMap );  216 : 2 : gmcDone = true;  217 : : }  218 : 83 : sf->setXwrtVChip(  219 : : sfAbsPosition.getX() + SgsRoundINT(corrPos.getX()) );  220 : 83 : sf->setYwrtVChip(  221 : : sfAbsPosition.getY() + SgsRoundINT(corrPos.getY()) );  222 : 83 : try  223 : : {  224 [ + + ]: 83 : if( gmcDone )  225 : : {  226 : 2 : Logger::instance()->log(LogMessageTable::eSHOT\_INFO,  227 : : "GMC is enabled for SF");  228 : : LongPosition sfPos(  229 : 2 : sf->getXwrtVChip(), sf->getYwrtVChip());  230 : 2 : mGMCCorrector.check( pSecondParams->gmc, sfPos );  231 : : }  232 : : else  233 : : {  234 : 81 : Logger::instance()->log(LogMessageTable::eSHOT\_INFO,  235 : : "GMC is disabled for SF");  236 : : }  237 : : }  238 [ # # ]: 0 : catch(DPL::Error \*pChild)  239 : : {  240 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  241 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  242 : : ErrorHandler::eErrFunction,  243 : : pErr->getErrorMsg( ErrorHandler::eErrFunction,  244 : : "GMCorrector::check"));  245 : : }  246 : : }  247 : : else  248 : : {  249 : 0 : Logger::instance()->log(LogMessageTable::eSHOT\_INFO,  250 : : "GMC is disabled for SF");  251 : 0 : sf->setXwrtVChip( sfAbsPosition.getX() );  252 : 0 : sf->setYwrtVChip( sfAbsPosition.getY() );  253 : : }  254 : 83 : sf->setIsGmcDone( true );  255 : 83 : if( !sf->getIsSegmentCreated() )  256 : : {  257 : : //TODO::ERROR  258 : 83 : sf->createSegments(mParams);  259 : : }  260 : 83 : sf->setIsEmpty( false );  261 : : }  262 : 228 : Segment::Pointer segment = sf->getSegmentsForLocationPriority();  263 : 228 : segment->createBuffer( false );  264 : :   265 : : //TF needs to be created first  266 : : //TODO::OPTIMIZE FROM MEMORY MANAGER  267 : 228 : targetTF = new(std::nothrow) TF();  268 : : //#ifdef ERROR\_SEED //<<Test\_Id:targetTF\_Set\_To\_NULL>>  269 : : // targetTF = NULL;  270 : : //#endif  271 [ - + ]: 228 : if( NULL == targetTF )  272 : : {  273 : 0 : int iErrNo = errno; //System error no  274 : 0 : char systemErrorMsg[BUFSIZ];  275 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  276 : 0 : pErr->errorAdd(NULL, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  277 : : ErrorHandler::eErrMemoryAllocation,  278 : : pErr->getErrorMsg(  279 : : ErrorHandler::eErrMemoryAllocation,  280 : : strerror\_r( iErrNo, systemErrorMsg, BUFSIZ )));  281 : : }  282 : 228 : segment->getTFs()->addTF( targetTF );  283 : : //GMC && SWING BACK ON TF IS DONE AFTER TF DISTRIBUTION  284 : 456 : TFInfo info;  285 : 228 : info.mTF = targetTF;  286 : 228 : info.mTFPosition = TF::AbsolutePosition( tfPositionX, tfPositionY);  287 : 228 : info.mLayerNo = inSFLNo;  288 : 228 : info.mShotCode = inPat.getCode();  289 : : #ifdef SWINGBACK\_BEFORE\_GMC  290 : : info.mTFPosition = mSwingBackApplier( info );  291 : : #endif  292 : 228 : const GMCMap \*pGMCMap = pParams->getGMCMap();  293 : : //IF GMC is enabled for TF  294 [ - + ]: 228 : if( pSecondParams->gmc.mode & 0x00000004 )  295 : : {  296 : 0 : DoublePosition corrPos;  297 : 0 : bool gmcDone = false;  298 [ # # ]: 0 : if( pSecondParams->gmc.mode & 0x00000001 )  299 : : {  300 : 0 : corrPos += mGMCCorrector.calculate( info.mTFPosition,  301 : : pGMCMap, pSecondParams->gmc,  302 : : GMCorrector::eGMCPolynomial );  303 : 0 : gmcDone = true;  304 : : }  305 [ # # ]: 0 : if( pSecondParams->gmc.mode & 0x00000002 )  306 : : {  307 : 0 : corrPos += mGMCCorrector.calculate( info.mTFPosition,  308 : : pGMCMap, pSecondParams->gmc,  309 : : GMCorrector::eGMCMap );  310 : 0 : gmcDone = true;  311 : : }  312 : 0 : info.mTFPosition += TF::AbsolutePosition(   313 : : SgsRoundINT8(corrPos.getX()),  314 : : SgsRoundINT8(corrPos.getY()) );  315 : 0 : try  316 : : {  317 [ # # ]: 0 : if( gmcDone )  318 : : {  319 : 0 : Logger::instance()->log(LogMessageTable::eSHOT\_INFO,  320 : : "GMC is enabled for TF");  321 : 0 : mGMCCorrector.check( pSecondParams->gmc, info.mTFPosition);  322 : : }  323 : : else  324 : : {  325 : 0 : Logger::instance()->log(LogMessageTable::eSHOT\_INFO,  326 : : "GMC is disabled for TF");  327 : : }  328 : : }  329 [ # # ]: 0 : catch(DPL::Error \*pChild)  330 : : {  331 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  332 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  333 : : ErrorHandler::eErrFunction,  334 : : pErr->getErrorMsg( ErrorHandler::eErrFunction,  335 : : "GMCorrector::check"));  336 : : }  337 : : }  338 : : else  339 : : {  340 : 228 : Logger::instance()->log(LogMessageTable::eSHOT\_INFO,  341 : : "GMC is disabled for TF");  342 : : }  343 : : #ifndef SWINGBACK\_BEFORE\_GMC  344 : 228 : info.mTFPosition = mSwingBackApplier( info );  345 : : #endif  346 : 228 : targetTF->setParentSF( sf );  347 : 228 : targetTF->setLocX( info.mTFPosition.getX() - sf->getXCoordinate() );  348 : 228 : targetTF->setLocY( info.mTFPosition.getY() - sf->getYCoordinate() );  349 : 228 : targetTF->setPatCode( inPat.getCode() );  350 [ + + ]: 228 : if( !tfLessMode )  351 : : {  352 : 196 : targetTF->setRow( normProcRow );  353 : 196 : targetTF->setColumn( normProcCol );  354 : : }  355 : :   356 : 228 : try  357 : : {  358 : 228 : pSDDB->TFDistributed(\*targetTF, info, sf);  359 : : }  360 [ # # ]: 0 : catch(DPL::Error \*pChild)  361 : : {  362 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  363 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_,  364 : : \_\_LINE\_\_, ErrorHandler::eErrFunction,  365 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  366 : : "SDDB::TFDistributed"));  367 : : }  368 : :   369 : : }  370 : :   371 : 305 : SimplePattern tmpPat = inPat;  372 : 305 : mShotTimeCalc.calculateShotTime(  373 : : tmpPat, inPosition, targetTF->getParentSF()->getDxpValue(), inSFLNo, inAI );  374 : : int64\_t tfOrigPosX = pLayer->llTFOriginX  375 : 305 : + ( targetTF->getColumn() \* tfSizeX );  376 : : int64\_t tfOrigPosY = pLayer->llTFOriginY  377 : 305 : + ( targetTF->getRow() \* tfSizeY );  378 : 305 : tmpPat.setPositionX( patPosWrtCell.getX() - (tfOrigPosX + (tfSizeX>>1)));  379 : 305 : tmpPat.setPositionY( patPosWrtCell.getY() - (tfOrigPosY + (tfSizeY>>1)));  380 : 305 : targetTF->addShot( tmpPat, inThreadID );  381 : 305 : mLastAddedTF[ inSFLNo ] = targetTF;  382 : 305 : e9k\_debug( "Shot at position (%lld, %lld) added to TF(%d,%d)\n",  383 : : inPosition.getX(), inPosition.getY(),  384 : : targetTF->getRow(), targetTF->getColumn());  385 : : }  386 : :   387 : 1215 : } // namespace nft |

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