| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [TFLayers/src](http://docs.google.com/index.html) - TFLayers.cpp (source / [functions](http://docs.google.com/TFLayers.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 100 | 139 | 71.9 % | | Date: | 2012-03-23 |  | Functions: | 19 | 22 | 86.4 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 30 | 50 | 60.0 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : #include <iostream>  2 : : #include <memory>  3 : :   4 : : #include <strings.h>  5 : :   6 : : #include "ConverterParams.h"  7 : : #include "SimplePattern.h"  8 : : #include "TFLayers.h"  9 : : #include "TFInfo.h"  10 : : #include "TF.h"  11 : : #include "TFRecycler.h"  12 : : #include "HealthCheckCounter.h"  13 : :   14 : : #include "print.h"  15 : :   16 : : namespace nft {  17 : :   18 : :   19 : : TFLayers::TFLayers()  20 : 1076 : : mNumberOfTFX(),  21 : : mNumberOfTFY(),  22 : : mTFSizeX(),  23 : : mTFSizeY(),  24 : : mLayerLocationX(),  25 : : mLayerLocationY(),  26 : : mMaxIndex(),  27 : : mMinIndex(),  28 : : mHandler(),  29 : : mMutex(),  30 : : mUnusedTFColumns(),  31 : : mUnusedTFs(),  32 : : mOldCol(),  33 : : mOldRow(),  34 : : mOldCode(),  35 : : mOldTF()  36 : 1076 : {  37 [ + + ][ # # ]: 6456 : for( size\_t i = 0; i <= 5; i ++ )  38 : : {  39 : 6456 : mLayers[ i ] = 0;  40 : : }  41 : : }  42 : :   43 : :   44 : 2152 : TFLayers::~TFLayers() {  45 : 0 : for( TFList::iterator it = mUnusedTFs.begin();  46 : : it != mUnusedTFs.end(); ++it )  47 : : {  48 : 0 : delete \*it;  49 : : }  50 : : }  51 : :   52 : :   53 : : TFLayers::TFLayers(const TFLayers & source)  54 : 0 : : mNumberOfTFX( source.mNumberOfTFX ),  55 : : mNumberOfTFY( source.mNumberOfTFY ),  56 : : mTFSizeX( source.mTFSizeX ),  57 : : mTFSizeY( source.mTFSizeY ),  58 : : mLayerLocationX( source.mLayerLocationX ),  59 : : mLayerLocationY( source.mLayerLocationY ),  60 : : mMaxIndex( source.mMaxIndex ),  61 : : mMinIndex( source.mMinIndex ),  62 : : mHandler( source.mHandler ),  63 : : mUnusedTFColumns( source.mUnusedTFColumns ),  64 : 0 : mUnusedTFs( source.mUnusedTFs ) {  65 [ # # ][ # # ]: 0 : for( size\_t i = 0; i <= 5; i ++ )  66 : : {  67 : 0 : mLayers[ i ] = source.mLayers[ i ];  68 : : }  69 : : }  70 : :   71 : : TFLayers &  72 : 0 : TFLayers::operator=(const TFLayers & source) {  73 [ # # ]: 0 : if( this != &source )  74 : : {  75 : 0 : mNumberOfTFX = source.mNumberOfTFX;  76 : 0 : mNumberOfTFY = source.mNumberOfTFY;  77 : 0 : mTFSizeX = source.mTFSizeX;  78 : 0 : mTFSizeY = source.mTFSizeY;  79 : 0 : mLayerLocationX = source.mLayerLocationX;  80 : 0 : mLayerLocationY = source.mLayerLocationY;  81 : 0 : mMaxIndex = source.mMaxIndex;  82 : 0 : mMinIndex = source.mMinIndex;  83 : 0 : mHandler = source.mHandler;  84 : 0 : mUnusedTFColumns = source.mUnusedTFColumns;  85 : 0 : mUnusedTFs = source.mUnusedTFs;  86 [ # # ]: 0 : for( size\_t i = 0; i <= 5; i ++ )  87 : : {  88 : 0 : mLayers[ i ] = source.mLayers[ i ];  89 : : }  90 : : }  91 : 0 : return \*this;  92 : : }  93 : :   94 : : void  95 : 1076 : TFLayers::setTFSizeX(int32\_t inTfSizeX) {  96 : 1076 : mTFSizeX = inTfSizeX;  97 : : }  98 : :   99 : : void  100 : 1076 : TFLayers::setTFSizeY(int32\_t inTfSizeY) {  101 : 1076 : mTFSizeY = inTfSizeY;  102 : : }  103 : :   104 : : void  105 : 1076 : TFLayers::setNumberOfTFX(uint16\_t inNumberOfTFX) {  106 : 1076 : mNumberOfTFX = inNumberOfTFX;  107 : : }  108 : :   109 : : void  110 : 1076 : TFLayers::setNumberOfTFY(uint16\_t inNumberOfTFY) {  111 : 1076 : mNumberOfTFY = inNumberOfTFY;  112 : : }  113 : :   114 : : void  115 : 1076 : TFLayers::setLayerLocationX(int64\_t inLocationX) {  116 : 1076 : mLayerLocationX = inLocationX;  117 : : }  118 : :   119 : : void  120 : 1076 : TFLayers::setLayerLocationY(int64\_t inLayerLocationY) {  121 : 1076 : mLayerLocationY = inLayerLocationY;  122 : : }  123 : :   124 : : void  125 : 0 : TFLayers::setMaxIndex(uint32\_t inMaxInd) {  126 : 0 : mMaxIndex = inMaxInd;  127 : : }  128 : :   129 : : void  130 : 0 : TFLayers::setMinIndex(uint32\_t inMinIndex) {  131 : 0 : mMinIndex = inMinIndex;  132 : : }  133 : :   134 : : TF::Pointer  135 : : TFLayers::getTFForPattern(const SimplePattern &inShot,  136 : : const PatternPosition & inPosition,  137 : 15753 : ThreadID inThreadID, uint32\_t inLayerNo ) throw(DPL::Error \*) {  138 : 15753 : HealthCheckCounter::instance()->incrCounter(  139 : : HealthCheckCounter::eUpdateFrequent);   140 : :   141 : 15753 : int64\_t col = 0;  142 : 15753 : int64\_t row = 0;  143 : 15753 : calculateIndices( inPosition, col, row );  144 : 15753 : PatternCode patCode = inShot.getCode();  145 : 15753 : if( inPosition.getX() < getLayerLocationX()  146 : : || col >= getNumberOfTFX()  147 : : || inPosition.getY() < getLayerLocationY()  148 : : || row >= getNumberOfTFY() )  149 : : {  150 : : //TODO::LOG  151 : 99 : Logger::instance()->log(LogMessageTable::eTF\_OUSIDE\_TF\_LAYER,  152 : : col, getNumberOfTFX(), row, getNumberOfTFY());  153 : 99 : return NULL;  154 : : }  155 [ + + ][ + + ]: 15654 : if( col == mOldCol && row == mOldRow && mOldTF && patCode == mOldCode )  [ + + ][ + + ]  156 : : {  157 : : // std::cout<<"Match"<<std::endl;  158 : 3648 : return mOldTF;  159 : : }  160 : : // std::cout<<"No Match"<<std::endl;  161 : 12006 : boost::mutex::scoped\_lock lock(mMutex);  162 : 12006 : mOldCol = col;  163 : 12006 : mOldRow = row;  164 : 12006 : mOldCode = patCode;  165 : : //The following section of code accesses shared data  166 : : //If c++0x atomic types are not used, the following will have to  167 : : //be a critical section  168 : : //Another options is to construct everything during startup  169 : : //but that would be memory consuming to say the least...  170 : : //However, in that case the data structure can be simplified - Santanu  171 : 12006 : TFLayer layer = mLayers[ patCode ];  172 : : //#ifdef ERROR\_SEED //<<Test\_Id:TFLayers\_layer\_Set\_To\_0>>  173 : : // layer = 0;  174 : : //#endif  175 [ + + ]: 12006 : if( 0 == layer )  176 : : {  177 : 1622 : layer = new(std::nothrow) TFColumn[ mNumberOfTFX ];  178 : : //#ifdef ERROR\_SEED //<<Test\_Id:TFLayers\_layer\_Set\_To\_0>>  179 : : // layer = NULL;  180 : : //#endif  181 [ - + ]: 1622 : if(NULL == layer)  182 : : {  183 : 0 : int iErrNo = errno; //System error no  184 : 0 : char systemErrorMsg[BUFSIZ];  185 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  186 : 0 : pErr->errorAdd(NULL, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  187 : : ErrorHandler::eErrMemoryAllocation,  188 : : pErr->getErrorMsg(  189 : : ErrorHandler::eErrMemoryAllocation,  190 : : strerror\_r( iErrNo, systemErrorMsg, BUFSIZ )));  191 : : }  192 : 1622 : mLayers[ patCode ] = layer;  193 : 1622 : ::bzero( layer, mNumberOfTFX \* sizeof(TFColumn) );  194 : : }  195 : 12006 : TFColumn tfCol = layer[ col ];  196 : : //#ifdef ERROR\_SEED //<<Test\_Id:TFLayers\_tfCol\_Set\_To\_0>>  197 : : // tfCol = 0;  198 : : //#endif  199 [ + + ]: 12006 : if( 0 == tfCol )  200 : : {  201 : 4507 : tfCol = new(std::nothrow) TFRows[ mNumberOfTFY ];  202 : : //#ifdef ERROR\_SEED //<<Test\_Id:TFLayers\_tfCol\_Set\_To\_0>>  203 : : // tfCol = NULL;  204 : : //#endif  205 [ - + ]: 4507 : if(NULL == tfCol)  206 : : {  207 : 0 : int iErrNo = errno; //System error no  208 : 0 : char systemErrorMsg[BUFSIZ];  209 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  210 : 0 : pErr->errorAdd(NULL, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  211 : : ErrorHandler::eErrMemoryAllocation,  212 : : pErr->getErrorMsg(  213 : : ErrorHandler::eErrMemoryAllocation,  214 : : strerror\_r( iErrNo, systemErrorMsg, BUFSIZ )));  215 : : }  216 : 4507 : layer[ col ] = tfCol;  217 : 4507 : ::bzero( tfCol, mNumberOfTFY \* sizeof(TFRows) );  218 : : }  219 : 12006 : TF::Pointer tf = tfCol[ row ];  220 [ + + ]: 12006 : if( 0 == tf )  221 : : {  222 : 10240 : try  223 : : {  224 : 10240 : tf = TFRecycler::instance()->getTF();  225 : : }  226 [ # # ]: 0 : catch( DPL::Error \*pChild )  227 : : {  228 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  229 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  230 : : ErrorHandler::eErrFunction,  231 : : pErr->getErrorMsg(  232 : : ErrorHandler::eErrFunction,  233 : : "TFRecycler::getTF"));  234 : : }  235 : 10240 : tfCol[ row ] = tf;  236 : 10240 : tf->setPatCode( patCode );  237 : 10240 : tf->setRow( row );  238 : 10240 : tf->setColumn( col );  239 : 20480 : TFInfo info;  240 : 10240 : TF::AbsolutePosition pos;  241 : 10240 : pos.setX( getLayerLocationX() + col \* mTFSizeX + (mTFSizeX >> 1));  242 : 10240 : pos.setY( getLayerLocationY() + row \* mTFSizeY + (mTFSizeY >> 1));  243 : : //TFSize/2 is added above, since TF origin is at center of TF  244 : 10240 : info.mTFPosition = pos;  245 : 10240 : info.mTF = tf;  246 : 10240 : info.mShotCode = patCode;  247 : 10240 : info.mLayerNo = inLayerNo;  248 : 10240 : try  249 : : {  250 : 10240 : mHandler( info );  251 : : }  252 [ + - ]: 12 : catch( DPL::Error \*pChild )  253 : : {  254 : 12 : ErrorHandler \*pErr = ErrorHandler::instance();  255 : 12 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  256 : : ErrorHandler::eErrFunction,  257 : : pErr->getErrorMsg(  258 : : ErrorHandler::eErrFunction,  259 : : "TFRecycler::getTF"));  260 : : }  261 : :   262 : : }  263 : 11994 : mOldTF = tf;  264 : 11994 : return tf;  265 : : }  266 : :   267 : : void  268 : 0 : TFLayers::addColumnToUnusedList(TFLayers::TFColumn inMemInfo) {  269 : : //UNUSED  270 : : }  271 : :   272 : : void  273 : : TFLayers::registerTFDistributionHandler(  274 : 1076 : const TFLayers::TFDistributionHandler & inHandler) {  275 : 1076 : mHandler = inHandler;  276 : : }  277 : :   278 : : void  279 : 10151 : TFLayers::releaseTF(TF::Pointer inTF, bool inCache) {  280 [ + - ]: 10151 : if( inCache )  281 : : {  282 : : {  283 : 10151 : boost::mutex::scoped\_lock lock( mMutex );  284 : 10151 : TFLayer layer = mLayers[ inTF->getPatCode() ];  285 : 10151 : TFColumn col = layer[inTF->getColumn()];  286 : 10151 : col[ inTF->getRow() ] = 0;  287 : 10151 : inTF->resetTF();  288 : : }  289 : 10151 : TFRecycler::instance()->saveTF( inTF );  290 : : }  291 : : else  292 : : {  293 [ # # ]: 10151 : delete inTF;  294 : : }  295 : : }  296 : :   297 : : void  298 : : TFLayers::releaseTFLayers()  299 : 1076 : {  300 [ + + ]: 7532 : for( uint32\_t i =0; i <= 5; i++ )  301 : : {  302 : 6456 : TFLayer layer = mLayers[i];  303 [ + + ]: 6456 : if( !layer )  304 : : {  305 : 1622 : continue;  306 : : }  307 : 0 : for( uint32\_t j = 0; j < getNumberOfTFX(); j++ )  308 : : {  309 : 113294 : TFColumn col = layer[ j ];  310 [ + + ]: 113294 : if( !col )  311 : : {  312 : 4507 : continue;  313 : : }  314 : 790559 : for( uint32\_t k = 0; k < getNumberOfTFY(); k++ )  315 : : {  316 : 393026 : TF::Pointer tf = col[ k ];  317 [ + + ]: 393026 : if( !tf )  318 : : {  319 : 89 : continue;  320 : : }  321 : 89 : tf->releaseShotMem();  322 : 89 : delete tf;  323 : 89 : col[ k ] = 0;  324 : : }  325 [ + - ]: 4507 : delete[] col;  326 : : }  327 [ + - ]: 1622 : delete[] layer;  328 : : }  329 : : }  330 : :   331 : : void  332 : : TFLayers::calculateIndices(const PatternPosition &inPos,  333 : 15753 : int64\_t & outIndexX, int64\_t & outIndexY) {  334 : 15753 : outIndexX = ( inPos.getX() - getLayerLocationX() ) / getTFSizeX();  335 : 15753 : outIndexY = ( inPos.getY() - getLayerLocationY() ) / getTFSizeY();  336 : : }  337 : :   338 : :   339 : 1215 : } // namespace nft |

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