| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [PatternProcessor/src](http://docs.google.com/index.html) - ShotShiftCalculator.cpp (source / [functions](http://docs.google.com/ShotShiftCalculator.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 49 | 49 | 100.0 % | | Date: | 2012-03-23 |  | Functions: | 8 | 8 | 100.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 10 | 12 | 83.3 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : :   2 : : #include "ShotShiftCalculator.h"  3 : : #include "ConverterParams.h"  4 : :   5 : : namespace nft {  6 : : ShotShiftCalculator::ShotShiftCalculator()  7 : 7468 : :mShiftFlag(true)  8 : 7468 : ,mLayers( ConverterParams::instance()->getLayerParams()) {  9 : : }  10 : 7468 : ShotShiftCalculator::~ShotShiftCalculator(){  11 : : }  12 : : unsigned int   13 : : ShotShiftCalculator::calculateShiftValue(  14 : : unsigned int MaxShiftNum,   15 : : int length,  16 : : unsigned int MaxShotSize,  17 : : unsigned int inSFLNo)   18 : 4638 : {  19 : 4638 : uint32\_t sflId = (mLayers + inSFLNo)->uiSFLID;  20 : 4638 : int Ns = (length-1)/MaxShotSize+1;  21 : 4638 : int Lm = length/Ns;  22 : 4638 : unsigned int shiftValue = ((sflId - 1) % MaxShiftNum)\*(Lm/MaxShiftNum); //TODO  23 : 4638 : return shiftValue;  24 : : }  25 : :   26 : : void   27 : : ShotShiftCalculator::adjustShotShiftAfterHorDivision(  28 : : std::vector<SimplePattern> &inPat,   29 : : std::vector<PatternPosition> &inPos,  30 : : std::vector<SimplePattern> &outPat,   31 : : std::vector<PatternPosition> &outPos,  32 : : unsigned int noOfPats,   33 : : unsigned int shiftSize)  34 : 813 : {  35 [ + + ]: 813 : if(shiftSize > 0){  36 : : //Initial inPattern  37 : 27 : outPat[0].setCode(inPat[noOfPats-1].getCode());  38 : 27 : outPos[0].setX(inPos[0].getX());   39 : 27 : outPos[0].setY(inPos[0].getY());  40 : 27 : outPat[0].setSizeX(shiftSize);  41 : 27 : outPat[0].setSizeY(inPat[noOfPats-1].getSizeY());  42 : :   43 : : //Middle inPatterns  44 [ + - ][ + + ]: 75 : for(unsigned int counter=1;counter<noOfPats;counter++){  45 : 48 : outPat[counter].setCode(outPat[counter-1].getCode());  46 : 48 : outPos[counter].setX(outPos[counter-1].getX() + outPat[counter-1].getSizeX());   47 : 48 : outPos[counter].setY(outPos[counter-1].getY()); //Position depends on previous pattern pos  48 : 48 : outPat[counter].setSizeX(inPat[counter-1].getSizeX());  49 : 48 : outPat[counter].setSizeY(inPat[counter-1].getSizeY());  50 : : }  51 : :   52 : : //Last inPattern   53 : 27 : outPat[noOfPats].setCode(inPat[noOfPats-1].getCode());  54 : 27 : outPat[noOfPats].setSizeX(inPat[noOfPats-1].getSizeX() - shiftSize);  55 : 27 : outPat[noOfPats].setSizeY(inPat[noOfPats-1].getSizeY());  56 : 27 : outPos[noOfPats].setX(outPos[noOfPats-1].getX() + outPat[noOfPats-1].getSizeX());  57 : 27 : outPos[noOfPats].setY(outPos[noOfPats-1].getY());  58 : : }else{  59 : 813 : mShiftFlag = false;  60 : : }  61 : : }  62 : :   63 : : void   64 : : ShotShiftCalculator::adjustShotShiftAfterVerDivision(  65 : : std::vector<SimplePattern> &inPat,   66 : : std::vector<PatternPosition> &inPos,  67 : : std::vector<SimplePattern> &outPat,   68 : : std::vector<PatternPosition> &outPos,  69 : : unsigned int noOfPats,   70 : : unsigned int size)  71 : 216 : {  72 [ + + ]: 216 : if(size > 0){  73 : : //Initial inPattern  74 : 3 : outPat[0].setCode(inPat[noOfPats-1].getCode());  75 : 3 : outPos[0].setX(inPos[0].getX());   76 : 3 : outPos[0].setY(inPos[0].getY());  77 : 3 : outPat[0].setSizeX(inPat[noOfPats-1].getSizeX());  78 : 3 : outPat[0].setSizeY(size);  79 : : // outPat[0].setAI(inPat[noOfPats-1].getAI());  80 : :   81 : : //Middle inPatterns  82 [ + - ][ + + ]: 8 : for(unsigned int counter=1;counter<noOfPats;counter++){  83 : 5 : outPat[counter].setCode(inPat[counter-1].getCode());  84 : 5 : outPos[counter].setX(outPos[counter-1].getX());  85 : 5 : outPos[counter].setY(outPos[counter-1].getY() + outPat[counter-1].getSizeY());  86 : 5 : outPat[counter].setSizeX(inPat[counter-1].getSizeX());  87 : 5 : outPat[counter].setSizeY(inPat[counter-1].getSizeY());  88 : : // outPat[counter].setAI(inPat[counter-1].getAI());  89 : : }  90 : :   91 : : //Last inPattern   92 : 3 : outPat[noOfPats].setCode(inPat[noOfPats-1].getCode());  93 : 3 : outPat[noOfPats].setSizeX(inPat[noOfPats-1].getSizeX());  94 : 3 : outPat[noOfPats].setSizeY(inPat[noOfPats-1].getSizeY() - size);  95 : 3 : outPos[noOfPats].setX(outPos[noOfPats-1].getX());  96 : 3 : outPos[noOfPats].setY(outPos[noOfPats-1].getY() + outPat[noOfPats-1].getSizeY());  97 : : // outPat[noOfPats].setAI(inPat[noOfPats-1].getAI());  98 : : }else{  99 : 216 : mShiftFlag = false;  100 : : }  101 : : }  102 : :   103 : 1215 : } // namespace nft |

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