| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [LibVsb/src](http://docs.google.com/index.html) - vsb\_patternCR.cpp (source / [functions](http://docs.google.com/vsb_patternCR.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 335 | 370 | 90.5 % | | Date: | 2012-03-23 |  | Functions: | 3 | 3 | 100.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 119 | 152 | 78.3 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : /\*--------------------------------------------------------------------------  2 : : \* Copyright (c) 2003, NuFlare Technology Inc., Digital GlobalSoft Ltd.  3 : : \* All Rights Reserved.  4 : : \* These technical data are subject to export control laws of Japan  5 : : \* regulations, and diversion contrary thereto is prohibited.  6 : : \*   7 : : \* Module : VSB Composite Representation Extraction Routines  8 : : \* Design Document : VSB Library - High Level Design (Rev 0.76)  9 : : \* Author : santhosh.kudva@digital.com  10 : : \* Date : 21st April, 2003  11 : : \*   12 : : \* File names prefixed with vsb\_ contain routines used privately by the library   13 : : \*   14 : : \* RCS Keywords  15 : : \* $Id: vsb\_patternCR.cpp,v 1.6 2012/02/10 05:07:19 santanu Exp $  16 : : \* $Log: vsb\_patternCR.cpp,v $  17 : : \* Revision 1.6 2012/02/10 05:07:19 santanu  18 : : \* Fix for E9K-80  19 : : \*  20 : : \* Revision 1.5 2011/08/18 11:07:00 abua  21 : : \* initial commit for SelfDiagnostics  22 : : \*  23 : : \* Revision 1.4 2011/07/05 11:20:39 asarkar  24 : : \* \*\*\* empty log message \*\*\*  25 : : \*  26 : : \* Revision 1.3 2011/05/23 10:47:40 asarkar  27 : : \* \*\*\* empty log message \*\*\*  28 : : \*  29 : : \* Revision 1.2 2011/05/19 09:23:45 asarkar  30 : : \* Updated  31 : : \*  32 : : \* Revision 1.1 2011/04/26 11:22:49 asarkar  33 : : \* VSB Library  34 : : \*  35 : : \* Revision 1.1 2007/07/05 06:47:42 santanu  36 : : \* Santanu::Second phase megre  37 : : \*  38 : : \* Revision 1.2 2007/02/05 11:46:12 santanu  39 : : \* EBM5K updated to version 1.67  40 : : \*  41 : : \* Revision 1.5 2005/03/15 00:30:15 ram  42 : : \* Added the changes for Self-Diagnostics macro. The SELF\_DIAG is changed to SELFDIAG\_PMD as a part of pattern shift detection.  43 : : \*  44 : : \* Revision 1.4 2004/09/20 21:57:29 ram  45 : : \* Changes related to Self Diagnostics  46 : : \*  47 : : \* Revision 1.3 2003/12/16 21:42:49 gopal  48 : : \* 1. Merged from main CVS after performance related changes  49 : : \*  50 : : \* Revision 1.16 2003/12/11 16:41:33 gopal  51 : : \* 1. Merged performance related changes from exp kit 6-b  52 : : \*  53 : : \* Revision 1.15 2003/11/10 15:58:32 raj  54 : : \* CR PG Keep bug fixes  55 : : \* 1. Termination conditions  56 : : \* 2. Flag resetting  57 : : \* 3. coordinate calculations  58 : : \* 4. counter updates  59 : : \*  60 : : \* Revision 1.14 2003/11/10 14:48:53 raj  61 : : \* CR PG Keep changes done by Santhosh Kudva San after CR PG Keep design  62 : : \*  63 : : \* 1. Complete changes for CR functions.  64 : : \* 2. related changes in fetching OR , NR and AR  65 : : \* 3. translate function changes  66 : : \* 4. Context data structure changes  67 : : \* ( this code release date 10 Oct 2003)  68 : : \* Revision 1.13 2003/10/06 23:58:27 kudva  69 : : \* Changes for Byte Alignment  70 : : \* 1. Added isAligned flag to context  71 : : \* 2. Added changes for APx and APy processing when isAligned is true  72 : : \* 3. Added changes to CR for APx and APy processing  73 : : \*  74 : : \* Checked in by - Santhosh Kudva  75 : : \*  76 : : \* Revision 1.12 2003/09/08 21:52:37 kudva  77 : : \* Bug ID #24 : Patterns have 5 AU Error (AR-Linear-Term, No Scale, AU Conversion Factor = 10)  78 : : \* Analysis : An Array Representation can have a non integral pitch. So to solve this,  79 : : \* 1. RoundINT was added to calculate pattern position (PP) from pitch.  80 : : \* 2. The PP was Translated. (This involves one more RoundINT)  81 : : \* Due to this the PP which was translated had a lower precision  82 : : \*  83 : : \* Eg. if pattern position was PP.X0 = 123.45 (after using pitch to calculate it)  84 : : \* RoundINT would result in PP.X0 = 123  85 : : \* AU Conversion = PP.X0 \* 10 = 1230  86 : : \*  87 : : \* However, if PP.X0 = 123.45  88 : : \* AU Conversion of this value would be 1234.5 and on round int. PP.X0 = 1235.  89 : : \* This is a 5 AU difference.  90 : : \*  91 : : \* Fix : To remove the RoundINT, the PP must be kept as a double. The same problem was also anticipated for CR. Therefore, vsb now implements a new type vsb\_ep and vsb\_pp, whose members are of primitive type double. EP & PP are now maintained in vsb\_ep and vsb\_pp structures till they have to be returned to the caller. This results in at most 0.5 AU of error from VSB for Pattern positions.  92 : : \*  93 : : \* Reviewed by - Gopal Mor  94 : : \* Checked in by - Santhosh Kudva  95 : : \*  96 : : \* Revision 1.11 2003/08/21 16:00:08 kudva  97 : : \* Bug ID : #182  98 : : \* Problem Statement : CR Patterns display cascade errors  99 : : \*  100 : : \* Analysis : The CR Term has a non integral pitch. The original design assumption in CR was that, Pitch would always evaluate to integral quantities. However, thi  101 : : \* s assumption is disproven by bug1.txt. Therefore  102 : : \*  103 : : \* a) CR Pitch now has to be evaluated as a double and stored as such in context  104 : : \* b) The CR start position or PP0 has to be stored in the context. This enables us to process the subsequent matrix columns without back computing (removing the added pitch from the offset) to the original PP0.  105 : : \* c) RoundINT now has to be applied to N\*pitchX, to avoid a cascade error  106 : : \*  107 : : \* Reviewed by - Gopal Mor  108 : : \* Checked in by - Santhosh Kudva  109 : : \*  110 : : \* Revision 1.10 2003/07/25 20:31:58 kudva  111 : : \* When changing the context.hxx ARContext structure a bug was found in CR. Where CR would try and get PP for representations which did not have PP's.  112 : : \*  113 : : \* Analysis : The context lookahead which was done for Arrays and NR was causing the CRGetPP to succeed for certain cases where it should not succeed. This was changed by setting the context crctx->WhichSubContext to NoContext. This prevents CRGetPP from succeeding unless explicitly it is supposed to succeed  114 : : \*  115 : : \* Review by - Shiva  116 : : \*  117 : : \* Revision 1.9 2003/07/09 23:04:16 kudva  118 : : \* Bug ID : 171  119 : : \* Missing Patterns in Chip Test 6  120 : : \* Problem : NR with one pattern at the end of a CR which is the last representation in a cell was not handled  121 : : \* Solution : Code added to move the representation pointer back to the beginning of the CR  122 : : \*  123 : : \* Checked in by : Santhosh Kudva  124 : : \*  125 : : \* Revision 1.8 2003/06/11 15:50:31 kudva  126 : : \* Bug ID : bug\_015, bug\_016, pg\_0x11\_core.maf pg\_0x11\_CRmissingPattern.maf  127 : : \* Problem : CR Dumps Core, some patterns are missing  128 : : \* Solution : Context was not being reset properly, If NR is the last Representation inside the CR then the CR pgc->current pointer was not being reset.  129 : : \*  130 : : \* Checked in by : Santhosh Kudva  131 : : \* Reviewed by : Shivakumar TR, Atul Nagar and Gopal Mor  132 : : \*  133 : : \* Revision 1.7 2003/05/27 21:26:28 kudva  134 : : \* Bug ID : #154  135 : : \* Summary : Anachronism  136 : : \* Resolution : Previous check in was not tested with Forte 6.2  137 : : \* Forte 5.0 does not expect typedefines for function pointers to be qualified with an extern "C", however the Forte version 6.2 expects this qualification.  138 : : \* We use the \_\_SUNPRO\_CC macro to distinguish between the different compilers.  139 : : \*  140 : : \* Revision 1.6 2003/05/27 15:00:25 rmakhija  141 : : \* Change to fix compilation errors at NFT  142 : : \*  143 : : \* Revision 1.5 2003/05/22 17:25:17 kudva  144 : : \* Summary :  145 : : \*  146 : : \* 1. CR computation for PP extraction, did not have code to continue the CR extraction, if the last SubRepresentation was an OR, hence it would terminate early  147 : : \* 2. Pitch calculation for Linear was incorrect. PP.X0 was added instead of subtracted.  148 : : \* 3. Added Seperate Pitch calculation for CRLinear  149 : : \* 4. Matrix Pitch Calculation was incorrect because the if condition was checking for CurrRepy=NumRepY. This is incorrect and not needed since the X pitch calculation happens only if Y pitch calculation is completed.  150 : : \*  151 : : \* Reviewed by - Narayana Reddy  152 : : \* Checked in by - Santhosh Kudva  153 : : \*  154 : : \* Revision 1.4 2003/04/30 00:32:07 narayan  155 : : \* Changes  156 : : \* 1. When exiting CR Representation the pgc->current was being updated with  157 : : \* pgc->current += Num Fields in PP \* NumPP \* sizeof(int)  158 : : \* instead of  159 : : \* pgc->current += Num Fields in PP \* NumPP \* X0Y0WordLength  160 : : \* because PP size depends on X0Y0WordLength size defined per segment  161 : : \*  162 : : \* Checked in by - Santhosh Kudva  163 : : \* Reviewed by - Narayana Reddy  164 : : \*  165 : : \* Revision 1.3 2003/04/29 22:55:03 narayan  166 : : \* Changes  167 : : \*  168 : : \* 1. Change due to change in ordinal value of NoContext in ContextType enumeration  169 : : \* 2. Initialization of CurrRepresentations, CurPP and NumPP to 1,1 and 0, due to the PPCount being PP0 + Num PP (for Optimize type in CR)  170 : : \* 3. WhichContext used instead of WhichSubContext  171 : : \* 4. Return values not checked correctly (checked for 1 instead of -1)  172 : : \* 5. X0Offset and Y0Offset reset to 0 when exiting a Composite Representation  173 : : \* 6. Switch on WhichContext instead of CRType  174 : : \*  175 : : \* Checked in by - Santhosh Kudva  176 : : \* Reviewed by - Narayana Reddy  177 : : \*  178 : : \* Revision 1.2 2003/04/25 17:30:06 narayan  179 : : \* Updated private declerations of function prototypes with extern "C" to  180 : : \* prevent Name Mangling  181 : : \*  182 : : \* Reviewed By - Narayana Reddy  183 : : \* Checked in By - Santhosh Kudva  184 : : \*  185 : : \* Revision 1.1 2003/04/23 21:43:42 kudva  186 : : \* VSB First Code Checkin - First Baseline - Santhosh Kudva  187 : : \*  188 : : \* -------------------------------------------------------------------------\*/  189 : :   190 : : #include <stdio.h>  191 : : #include "dec\_types.h"  192 : : #include "dec\_math.h"  193 : : #include "vsb\_types.h"  194 : : #include "vsb\_utils.h"  195 : : #include "vsb\_translate.h"  196 : : #include "context.h"  197 : : #include "PH4Flag.h"  198 : : #include "SDDB.h"  199 : : namespace nft {  200 : : extern "C" int NRGetEP(GroupContext \*pgc, EP \*ep, int \*Nx, int \*Ny, int \*Xe, int \*Ye) ;  201 : : extern "C" int ORGetEP(GroupContext \*pgc, EP \*ep, int \*Nx, int \*Ny, int \*Xe, int \*Ye) ;  202 : : extern "C" int ARGetEP(GroupContext \*pgc, EP \*ep, int \*Nx, int \*Ny, int \*Xe, int \*Ye) ;  203 : :   204 : : extern "C" int ORGetPP(GroupContext \*pgc, PP \*pp) ;  205 : : extern "C" int ARGetPP(GroupContext \*pgc, PP \*pp) ;  206 : : extern "C" int CRGetPP(GroupContext \*pgc, PP \*pp) ;   207 : :   208 : : /\* Private Declerations \*/  209 : :   210 : : extern "C" int CRGetEP(GroupContext \*pgc, EP \*ep, int \*Nx, int \*Ny, int \*Xe, int \*Ye) ;  211 : : extern "C" int CRGetPP(GroupContext \*pgc, PP \*pp) ;  212 : :   213 : : #if (!defined(\_\_SUNPRO\_CC) || \_\_SUNPRO\_CC < 0x530)  214 : : typedef int (\* EPFunction)(GroupContext \*, EP \*, int \*, int \*, int \*, int \*) ;  215 : : typedef int (\* PPFunction)(GroupContext \*, PP \*) ;   216 : : typedef int (\* SizeFunction)(Byte \*, const int, const int) ;   217 : : #else  218 : : extern "C" typedef int (\* EPFunction)(GroupContext \*, EP \*, int \*, int \*, int \*, int \*) ;  219 : : extern "C" typedef int (\* PPFunction)(GroupContext \*, PP \*) ;   220 : : extern "C" typedef int (\* SizeFunction)(Byte \*, const int, const int) ;   221 : : #endif  222 : :   223 : : static EPFunction WhichEPFunction[4]={NULL, NRGetEP, ORGetEP, ARGetEP} ;  224 : : //Interra commented this to suppress warning of unused variable  225 : : //static PPFunction WhichPPFunction[4]={NULL, NULL, ORGetPP, ARGetPP} ;  226 : : static SizeFunction WhichSizeFunction[4]={NULL, sizeofNR, sizeofOR, sizeofAR} ;  227 : :   228 : :   229 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  230 : : \* Name: CRGetEP  231 : : \* Functionality: Extracts a CR Representation (OR2:PH1(NPEPiPP+)+)  232 : : \* Input: Pointers to GroupContext, EP, Number of X Repetitions, Number of Y   233 : : \* Repetitions, Array end position X direction, Array end position Y direction  234 : : \* Output: Returns an EP as a Array Representation  235 : : \* Return value: Success or failure of the operation  236 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  237 : : int CRGetEP(GroupContext \*pgc, EP \*ep, int \*Nx, int \*Ny, int \*Xe, int \*Ye)  238 : 480 : {  239 : 480 : CRContext \*crctx = NULL ;  240 : : //vsbPH3 \*ph3 = NULL ;  241 : 480 : vsbPH3 ph3 ;  242 : : // int SkipSize = 0 ;  243 : : //int NumFieldsInEP = 0 ;  244 : 480 : int Value = 0 ;  245 : 480 : int status = 0 ;  246 : 480 : PatternArrangement parrange ;  247 : 480 : ArrayType arrtype ;  248 : 480 : PP pp ;  249 : 480 : vsbPP gPPi ;  250 : 480 : ContextType ctype ;  251 : 480 : vsbEP vep ;  252 : 480 : double vXe, vYe ;  253 : 480 : alias\_PH4\_1 alias\_PH4\_1Val;  254 : 480 : alias\_PH4\_2 alias\_PH4\_2Val;  255 : 480 : alias\_PH3 alias\_PH3Val;  256 : :   257 : :   258 : :   259 : 480 : crctx = &(pgc->ppc.crctx) ;  260 : :   261 [ + + ]: 480 : if(crctx->isEmpty)  262 : : {  263 : 59 : uint temp ;   264 : : //ph3 = (vsbPH3 \*)pgc->current ;  265 : : //temp = vsbGetInt(pgc->current) ;  266 : 59 : alias\_PH4\_1Val.value = alias\_PH4\_2Val.value = vsbGetInt(pgc->current) ;  267 : : //PH4 handling  268 : 59 : if(1 == PH4Flag::instance()->getValue())  269 : : {  270 : 0 : vsbPH4\_1 ph4;  271 : 0 : ph4 = alias\_PH4\_1Val.ph4 ;  272 [ # # ][ # # ]: 0 : switch(ph4.b)  273 : : {  274 : : case 0xb1:  275 : : {  276 : 0 : vsbPH4\_1 ph4\_1;  277 : : //ph4\_1= \*(vsbPH4\_1 \*)(&temp);  278 : 0 : ph4\_1 = alias\_PH4\_1Val.ph4 ;  279 : 0 : pgc->attributeInfo = ph4\_1.AI1;  280 : 0 : pgc->current = pgc->current + sizeof(vsbPH4\_1);  281 : 0 : break;  282 : : }  283 : : case 0xb2:  284 : : {  285 : 0 : vsbPH4\_2 ph4\_2;  286 : : //ph4\_2 = \*(vsbPH4\_2 \*)(&temp);  287 : 0 : ph4\_2 = alias\_PH4\_2Val.ph4 ;  288 : 0 : pgc->attributeInfo = ph4\_2.AI2;  289 : 0 : pgc->current = pgc->current + sizeof(vsbPH4\_2);  290 : 0 : break;  291 : : }  292 : : }  293 : : }  294 : : else  295 : : {  296 : 59 : pgc->attributeInfo=0;  297 : : }  298 : :   299 : : //temp = vsbGetInt(pgc->current) ;  300 : 59 : alias\_PH3Val.value = vsbGetInt(pgc->current) ;  301 : : //ph3 = \*(vsbPH3 \*)(&temp) ;  302 : 59 : ph3 = alias\_PH3Val.ph3;  303 : 59 : pgc->current += sizeof(vsbPH3) ;  304 : :   305 : 59 : pgc->current += vsbBufferToPP(pgc->current, XYDirxn, pgc->X0Y0WordLength, &pp) ;   306 : :   307 : 59 : crctx->gPP0 = pp ;  308 : :   309 : 59 : \*Nx = \*Ny = 1 ;  310 : 59 : \*Xe = \*Ye = 0 ;  311 : :   312 : 59 : crctx->NumRepX = crctx->NumRepY = 0 ;  313 : 59 : crctx->CurrRepX = crctx->CurrRepY = 0 ;  314 : 59 : crctx->CurrRepresentations = 1 ;  315 : 59 : crctx->CurrPP = 1 ;  316 : 59 : crctx->NumPP = 0 ;  317 : 59 : crctx->PitchX = crctx->PitchY = 0 ;  318 : :   319 : 59 : crctx->isPatternOver = true ;  320 : 59 : crctx->isReprEPOver = false ;  321 : 59 : crctx->WhichSubContext = NoContext ;  322 : 59 : crctx->Xe = pp.X0 ;  323 : 59 : crctx->Ye = pp.Y0 ;  324 : 59 : SDDB \*pSDDB = SDDB::instance();  325 : 59 : try  326 : : {  327 : 59 : pSDDB->updateInPatCR();   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 : : "SDDB::updateInPatCR"));  336 : : }  337 : :   338 [ + - ][ + + ]: 59 : switch(ph3.codes)  [ + - ]  339 : : {  340 : : case 0x91 :  341 : : case 0x92 :  342 : 32 : crctx->CRType = CRArray ;  343 : 32 : crctx->NumRepresentations = ph3.Nv ;  344 : 32 : crctx->PatArrangement = parrange = WhichArrangement[ph3.codea] ;  345 : 32 : arrtype = WhichArrayType[ph3.codes % 0x91] ;  346 : :   347 [ + + ][ + + ]: 32 : switch(parrange)  [ - + ][ + + ]  [ - + ]  348 : : {  349 : : case Horizontal :   350 : : // crctx->NumRepX = \*(short int \*)pgc->current ;   351 : 5 : crctx->NumRepX = vsbGetShort(pgc->current) ;   352 : 5 : crctx->NumRepY = 1 ;   353 : 5 : pgc->current+= sizeof(short int) ;   354 : 5 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, pgc->isAligned, &Value) ;  355 : 5 : crctx->PitchX = Value ;  356 : 5 : crctx->Xe = Value ;  357 [ + + ]: 5 : if(Term == arrtype)  358 : : {  359 [ + - ]: 3 : crctx->PitchX = (crctx->NumRepX > 1) ? (crctx->PitchX - pp.X0)/(crctx->NumRepX - 1) : 0 ;  360 : : }   361 : : else  362 : : {  363 : 2 : crctx->Xe = (crctx->Xe \* (crctx->NumRepX-1)) + pp.X0 ;  364 : : }  365 : 2 : break ;  366 : : case Vertical :  367 : 6 : crctx->NumRepX = 1 ;   368 : : // crctx->NumRepY = \*(short int \*)pgc->current ;   369 : 6 : crctx->NumRepY = vsbGetShort(pgc->current) ;   370 : 6 : pgc->current+= sizeof(short int) ;   371 : 6 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, pgc->isAligned, &Value) ;  372 : 6 : crctx->PitchY = Value ;  373 : 6 : crctx->Ye = Value ;  374 [ + + ]: 6 : if(Term == arrtype)  375 : : {  376 [ + - ]: 3 : crctx->PitchY = (crctx->NumRepY > 1) ? (crctx->PitchY - pp.Y0)/(crctx->NumRepY - 1) : 0 ;  377 : : }   378 : : else  379 : : {  380 : 3 : crctx->Ye = (crctx->Ye \* (crctx->NumRepY-1)) + pp.Y0 ;   381 : : }  382 : 3 : break ;  383 : : case Matrix :  384 : 13 : crctx->NumRepX = vsbGetShort(pgc->current) ;   385 : 13 : pgc->current+= sizeof(short int) ;   386 : 13 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, pgc->isAligned, &Value) ;  387 : 13 : crctx->PitchX = Value ;  388 : 13 : crctx->Xe = Value ;  389 : :   390 : 13 : crctx->NumRepY = vsbGetShort(pgc->current) ;   391 : 13 : pgc->current+= sizeof(short int) ;   392 : 13 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, pgc->isAligned, &Value) ;  393 : 13 : crctx->PitchY = Value ;  394 : 13 : crctx->Ye = Value ;  395 [ + + ]: 13 : if(Term == arrtype)  396 : : {  397 [ + - ]: 11 : crctx->PitchX = (crctx->NumRepX > 1) ? (crctx->PitchX - pp.X0)/(crctx->NumRepX - 1) : 0 ;  398 [ + - ]: 11 : crctx->PitchY = (crctx->NumRepY > 1) ? (crctx->PitchY - pp.Y0)/(crctx->NumRepY - 1) : 0 ;  399 : : }  400 : : else  401 : : {  402 : 2 : crctx->Xe = (crctx->Xe \* (crctx->NumRepX-1)) + pp.X0 ;  403 : 2 : crctx->Ye = (crctx->Ye \* (crctx->NumRepY-1)) + pp.Y0 ;   404 : : }  405 : 2 : break ;  406 : : case Linear :  407 : 8 : crctx->CRType = CRLinear ;  408 : 8 : crctx->NumRepX = crctx->NumRepY = vsbGetShort(pgc->current) ;   409 : 8 : pgc->current += sizeof(short int) ;  410 : 8 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, 0, &Value) ;  411 : 8 : crctx->PitchX = Value ;  412 : 8 : pgc->current += vsbBufferToValue(pgc->current, pgc->X0Y0WordLength, 0, &Value) ;  413 : 8 : crctx->PitchY = Value ;  414 : :   415 [ + + ]: 8 : if(Term == arrtype)  416 : : {  417 [ + - ]: 5 : crctx->PitchX = (crctx->NumRepX > 1) ? (crctx->PitchX - pp.X0)/(crctx->NumRepX - 1) : 0 ;  418 [ + - ]: 5 : crctx->PitchY = (crctx->NumRepY > 1) ? (crctx->PitchY - pp.Y0)/(crctx->NumRepY - 1) : 0 ;  419 : : }  420 : 5 : break ;  421 : 0 : default: return -1; //Invalid context  422 : : }  423 : 27 : break ;  424 : : case 0xa0 :  425 : : case 0xa1 :  426 : : case 0xa2 :   427 : 27 : crctx->CRType = CROptimize ;  428 : 27 : crctx->NumRepresentations = ph3.codea ;  429 : 27 : crctx->NumPP = ph3.Nv ;  430 : 27 : crctx->PPDirection = WhichDirection[ph3.codes % 0xa0] ;  431 : 3 : break ;  432 : : }  433 : :   434 : 59 : crctx->StartReprPtr = pgc->current ;  435 : 59 : crctx->CurrReprPtr = pgc->current ;  436 : 59 : crctx->isEmpty = false ;  437 : :   438 : 59 : crctx->CurrPPptr = crctx->StartPPptr = NULL ;  439 : : // Compute Start of PPptr  440 [ + - ][ + + ]: 284 : for(int i=0; i < crctx->NumRepresentations; i++)  441 : : {  442 : 225 : ctype = GetContextType(crctx->CurrReprPtr) ;  443 : 225 : crctx->CurrReprPtr += WhichSizeFunction[ctype](crctx->CurrReprPtr, pgc->X0Y0WordLength, pgc->isAligned) ;  444 : : }  445 : :   446 [ + + ]: 59 : if(CROptimize == crctx->CRType)  447 : : {  448 : 27 : int SkipSize ;  449 : 27 : crctx->StartPPptr = crctx->CurrReprPtr ;  450 : 27 : crctx->CurrPPptr = crctx->CurrReprPtr ;  451 [ + + ]: 27 : if(XYDirxn == crctx->PPDirection)  452 : : {  453 : 19 : SkipSize = (2 \* pgc->X0Y0WordLength) \* crctx->NumPP ;  454 : : }  455 : : else  456 : : {  457 : 8 : SkipSize = pgc->X0Y0WordLength \* crctx->NumPP ;  458 : : }  459 : 27 : crctx->EndCRptr = crctx->CurrReprPtr + SkipSize ;  460 : : }   461 : : else  462 : : {  463 : 32 : crctx->EndCRptr = crctx->CurrReprPtr ;  464 : : }  465 : 480 : crctx->CurrReprPtr = crctx->StartReprPtr ;  466 : : }  467 : :   468 : :   469 : 480 : pgc->current = crctx->CurrReprPtr ;  470 : : ExtractEP:  471 [ + + ]: 561 : if(crctx->isPatternOver)  472 : : {  473 [ + + ]: 284 : if(NoContext == crctx->WhichSubContext)  474 : : {  475 : 59 : ctype = GetContextType(pgc->current) ;  476 : 59 : crctx->WhichSubContext = ctype ;  477 : 59 : pgc->ppc.arctx.isEmpty = true ;  478 : : }  479 : : else  480 : : {  481 : 225 : ctype = crctx->WhichSubContext ;  482 : : }  483 : 284 : status = WhichEPFunction[ctype](pgc, ep, Nx, Ny, Xe, Ye) ;  484 : :   485 [ + + ]: 284 : if(-1 == status)  486 : : {  487 : :   488 : 225 : SDDB \*pSDDB = SDDB::instance();  489 : 225 : try  490 : : {  491 : 225 : pSDDB->CRPatHandling(crctx);  492 : : }  493 [ # # ]: 0 : catch( DPL::Error \*pChild )  494 : : {  495 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  496 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  497 : : ErrorHandler::eErrFunction,  498 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  499 : : "SDDB::CRPatHandling"));  500 : : }  501 : :   502 : :   503 : 225 : crctx->CurrRepresentations++ ;  504 : 225 : crctx->WhichSubContext = NoContext ;  505 [ + + ]: 225 : if(crctx->CurrRepresentations <= crctx->NumRepresentations)  506 : : {  507 : 166 : if(NoContext == crctx->WhichSubContext)  508 : : {  509 : 166 : ctype = GetContextType(pgc->current) ;  510 : 166 : crctx->WhichSubContext = ctype ;  511 : 166 : pgc->ppc.arctx.isEmpty = true ;  512 : : }  513 : : else  514 : : {  515 : 166 : ctype = crctx->WhichSubContext ;  516 : : }  517 : 166 : status = WhichEPFunction[ctype](pgc, ep, Nx, Ny, Xe, Ye) ;  518 : : }  519 : : else  520 : : {  521 [ + + ]: 59 : if(CROptimize == crctx->CRType)  522 : : {  523 : : // Set pgc->current to the end of the CR  524 : 27 : pgc->current = crctx->EndPPptr ;  525 : : }  526 : :   527 : 59 : return -1 ;  528 : : }  529 : : }   530 : : }   531 : :   532 : 502 : crctx->CurrReprPtr = pgc->current ;   533 : 502 : crctx->isPatternOver = false ;   534 : :   535 : 502 : itod(&(crctx->ReprEP), &vep) ;   536 : :   537 : 502 : crctx->ReprPP.X0 = vep.X0;  538 : 502 : crctx->ReprPP.Y0 = vep.Y0;  539 : :   540 : 502 : vXe = crctx->ReprXe ;  541 : 502 : vYe = crctx->ReprYe ;  542 : :   543 [ + + ]: 502 : if(CROptimize == crctx->CRType || CRLinear == crctx->CRType)  544 : : {  545 [ + + ][ + + ]: 211 : if(Array == crctx->WhichSubContext && Linear != pgc->ppc.arctx.PatArrangement)  546 : : {  547 : : // Handle Optimize-Array  548 [ + + ]: 96 : if(CROptimize == crctx->CRType)  549 : : {  550 [ + + ]: 69 : if(crctx->isReprEPOver)  551 : : {  552 [ + + ]: 51 : if(crctx->CurrPP < crctx->NumPP)  553 : : {  554 : 33 : crctx->CurrPP++ ;  555 : :   556 : : /\* For cr.or.x / cr.or.y \*/  557 : 33 : pp = crctx->gPP0 ;  558 : :   559 : 33 : crctx->CurrPPptr += vsbBufferToPP(crctx->CurrPPptr, crctx->PPDirection, pgc->X0Y0WordLength, &pp) ;  560 : : // Should this gPPi go to the context ?  561 : 33 : gPPi.X0 = pp.X0 ;  562 : 33 : gPPi.Y0 = pp.Y0 ;  563 : :   564 : 33 : crctx->ReprPP.X0 = gPPi.X0 ;  565 : 33 : crctx->ReprPP.Y0 = gPPi.Y0 ;  566 : :   567 : 33 : vep.X0 += gPPi.X0 ;  568 : 33 : vep.Y0 += gPPi.Y0 ;  569 : 33 : vXe += gPPi.X0 ;  570 : 33 : vYe += gPPi.Y0 ;  571 : : }  572 : : else  573 : : {  574 : 18 : crctx->isPatternOver = true ;  575 : 18 : crctx->isReprEPOver = false ;  576 : 18 : crctx->CurrPP = 1 ;  577 : 18 : crctx->EndPPptr = crctx->CurrPPptr;  578 : 18 : crctx->CurrPPptr = crctx->StartPPptr ;  579 : 18 : goto ExtractEP ;  580 : : }   581 : : }   582 : : else  583 : : {  584 : 18 : vep.X0 += crctx->gPP0.X0 ;  585 : 18 : vep.Y0 += crctx->gPP0.Y0 ;  586 : 18 : vXe += crctx->gPP0.X0 ;  587 : 18 : vYe += crctx->gPP0.Y0 ;  588 : 18 : crctx->CurrPP = 1 ;  589 : 18 : crctx->isReprEPOver = true ;  590 : : }  591 : : }   592 : : else // CRLinear => Compute gPPi   593 : : {  594 [ + + ]: 27 : if(crctx->isReprEPOver)  595 : : {  596 [ + + ]: 19 : if(crctx->CurrRepY < crctx->NumRepY-1)  597 : : {  598 : 11 : crctx->CurrRepY++ ;  599 : 11 : gPPi.Y0 = crctx->gPP0.Y0 + (crctx->CurrRepY \* crctx->PitchY) ;  600 : 11 : gPPi.X0 = crctx->gPP0.X0 + (crctx->CurrRepY \* crctx->PitchX) ;  601 : :   602 : 11 : crctx->ReprPP.X0 = gPPi.X0 ;  603 : 11 : crctx->ReprPP.Y0 = gPPi.Y0 ;  604 : :   605 : 11 : vep.X0 += gPPi.X0 ;  606 : 11 : vep.Y0 += gPPi.Y0 ;  607 : 11 : vXe += gPPi.X0 ;  608 : 11 : vYe += gPPi.Y0 ;  609 : : }  610 : : else  611 : : {  612 : 8 : crctx->isPatternOver = true ;  613 : 8 : crctx->isReprEPOver = false ;  614 : 8 : crctx->EndPPptr = crctx->CurrReprPtr;  615 : 8 : crctx->CurrPPptr = crctx->StartPPptr ;  616 : 8 : crctx->CurrRepY = 0 ;  617 : 8 : goto ExtractEP ;  618 : : }   619 : : }  620 : : else  621 : : {  622 : 8 : vep.X0 += crctx->gPP0.X0 ;  623 : 8 : vep.Y0 += crctx->gPP0.Y0 ;  624 : 8 : vXe += crctx->gPP0.X0 ;  625 : 8 : vYe += crctx->gPP0.Y0 ;  626 : 8 : crctx->CurrRepY = 0 ;  627 : 8 : crctx->isReprEPOver = true ;  628 : : }  629 : : }  630 : : }  631 : : else  632 : : {  633 : : // Handle Optimize-Normal, Optimize-Optimize, Optimize-Linear  634 : 115 : vep.X0 += crctx->gPP0.X0 ;  635 : 115 : vep.Y0 += crctx->gPP0.Y0 ;  636 : 115 : vXe += crctx->gPP0.X0 ;  637 : 115 : vYe += crctx->gPP0.Y0 ;  638 : 115 : crctx->CurrRepY = 0 ;  639 : 115 : crctx->CurrPP = 1 ;  640 : : }  641 : : }  642 [ + - ]: 291 : else if(CRArray == crctx->CRType)  643 : : {  644 [ + + ][ + + ]: 291 : if(Optimize == crctx->WhichSubContext || (Array == crctx->WhichSubContext && Linear == pgc->ppc.arctx.PatArrangement))  [ + + ]  645 : : {  646 : : // Handle Array-Optimize  647 [ + + ]: 102 : if(crctx->isReprEPOver)  648 : : {  649 [ + + ][ - + ]: 69 : switch(crctx->WhichSubContext)  650 : : {  651 : 54 : case Optimize : status = ORGetPP(pgc, &pp) ;  652 : 54 : break ;  653 : 15 : case Array : status = ARGetPP(pgc, &pp);   654 : 15 : break ;  655 : 0 : default : status = -1 ;   656 : 69 : break ;  657 : : }  658 : :   659 : 69 : crctx->CurrReprPtr = pgc->current ;  660 : :   661 [ + + ]: 69 : if(-1 == status)  662 : : {  663 : 33 : crctx->isPatternOver = true ;  664 : 33 : crctx->isReprEPOver = false ;  665 : 33 : goto ExtractEP ;  666 : : }  667 : 36 : vep.X0 = crctx->ReprPP.X0 + crctx->gPP0.X0 ;  668 : 36 : vep.Y0 = crctx->ReprPP.Y0 + crctx->gPP0.Y0 ;  669 : 36 : vXe = crctx->ReprPP.X0 + crctx->Xe ;  670 : 36 : vYe = crctx->ReprPP.Y0 + crctx->Ye ;  671 : : }  672 : : else  673 : : {  674 : : // Sub Representations EP  675 : 33 : vep.X0 += crctx->gPP0.X0 ;  676 : 33 : vep.Y0 += crctx->gPP0.Y0 ;  677 : 33 : vXe += crctx->Xe ;  678 : 33 : vYe += crctx->Ye ;  679 : 33 : crctx->isReprEPOver = true ;  680 : : }  681 : 69 : \*Nx = crctx->NumRepX ;  682 : 69 : \*Ny = crctx->NumRepY ;  683 : : }  684 [ + + ][ - + ]: 163 : else if(Array == crctx->WhichSubContext && (Horizontal == crctx->PatArrangement && Vertical == pgc->ppc.arctx.PatArrangement))   685 : : {  686 : : // Handle Array-Horizontal-Vertical=>Matrix  687 : 0 : vep.X0 += crctx->gPP0.X0 ;  688 : 0 : vep.Y0 += crctx->gPP0.Y0 ;  689 : 0 : vXe += crctx->Xe ;  690 : 0 : vYe += crctx->Ye ;  691 : 0 : \*Nx = crctx->NumRepX ;  692 : 0 : crctx->isPatternOver = true ;  693 : : }  694 [ + + ][ + + ]: 163 : else if(Array == crctx->WhichSubContext && (Vertical == crctx->PatArrangement && Horizontal == pgc->ppc.arctx.PatArrangement))  695 : : {  696 : : // Handle Array-Vertical-Horizontal=>Matrix  697 : 3 : vep.X0 += crctx->gPP0.X0 ;  698 : 3 : vep.Y0 += crctx->gPP0.Y0 ;  699 : 3 : vXe += crctx->Xe ;  700 : 3 : vYe += crctx->Ye ;  701 : 3 : \*Ny = crctx->NumRepY ;  702 : 3 : crctx->isPatternOver = true ;  703 : : }  704 : 156 : else if(Array == crctx->WhichSubContext)  705 : : {  706 : : // Handle Array-Array  707 [ + + ]: 160 : if(crctx->isReprEPOver)  708 : : {  709 [ + + ]: 138 : if(crctx->CurrRepY < crctx->NumRepY-1)  710 : : {  711 : 62 : crctx->CurrRepY++ ;  712 : : }  713 [ + + ]: 76 : else if(crctx->CurrRepX < crctx->NumRepX-1)  714 : : {  715 : 54 : crctx->CurrRepX++ ;  716 : 54 : crctx->CurrRepY = 0 ;  717 : : }  718 : : else  719 : : {  720 : 22 : crctx->isPatternOver = true ;  721 : 22 : crctx->CurrRepX = 0 ;  722 : 22 : crctx->CurrRepY = 0 ;  723 : 22 : crctx->isReprEPOver = false ;  724 : 22 : goto ExtractEP ;  725 : : }   726 : 116 : gPPi.X0 = crctx->gPP0.X0 + (crctx->CurrRepX \* crctx->PitchX) ;  727 : 116 : gPPi.Y0 = crctx->gPP0.Y0 + (crctx->CurrRepY \* crctx->PitchY) ;  728 : 116 : vep.X0 += gPPi.X0 ;   729 : 116 : vep.Y0 += gPPi.Y0 ;   730 : 116 : vXe += gPPi.X0 ;   731 : 116 : vYe += gPPi.Y0 ;   732 : : }  733 : : else  734 : : {  735 : 22 : vep.X0 += crctx->gPP0.X0 ;   736 : 22 : vep.Y0 += crctx->gPP0.Y0 ;   737 : 22 : vXe += crctx->gPP0.X0 ;   738 : 22 : vYe += crctx->gPP0.Y0 ;   739 : 22 : crctx->isReprEPOver = true ;  740 : : }  741 : : }  742 : : else  743 : : {  744 : : // Handle Array-Normal  745 : 26 : vep.X0 += crctx->gPP0.X0 ;  746 : 26 : vep.Y0 += crctx->gPP0.Y0 ;  747 : 26 : vXe += crctx->Xe ;  748 : 26 : vYe += crctx->Ye ;  749 : 26 : \*Nx = crctx->NumRepX ;  750 : 26 : \*Ny = crctx->NumRepY ;  751 : 26 : crctx->isPatternOver = true ;  752 : : }  753 : : }  754 : :   755 [ + - ][ - + ]: 421 : if ( 1.0F != pgc->TranslateFactor || pgc->mirror)  756 : : {  757 [ # # ]: 0 : if( 0 > TranslateEP(pgc, &vep, &vXe, &vYe) )  758 : 0 : return -1 ;  759 : : }  760 : 421 : dtoi(&vep, ep) ;  761 : 421 : \*Xe = RoundINT(vXe) ;  762 : 421 : \*Ye = RoundINT(vYe) ;  763 : 421 : pgc->current = crctx->StartReprPtr ;  764 : 480 : return 0 ;  765 : : }  766 : :   767 : : /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  768 : : \* Name: CRGetPP  769 : : \* Functionality: Returns one PP from a CR Representation   770 : : \* Input: Pointers to GroupContext and PP  771 : : \* Output: Returns a PP   772 : : \* Return value: Success or failure of the operation  773 : : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  774 : :   775 : : int CRGetPP(GroupContext \*pgc, PP \*pp)  776 : 518 : {  777 : 518 : CRContext \*crctx = NULL ;  778 : : //ORContext \*orctx = NULL ;  779 : 518 : int status = 0 ;  780 : 518 : vsbPP gPPi ;  781 : 518 : vsbPP vpp ;  782 : :   783 : 518 : crctx = &(pgc->ppc.crctx) ;  784 : :   785 [ + - ]: 518 : if(CROptimize == crctx->CRType || CRLinear == crctx->CRType)  786 : : {  787 [ + + ][ + - ]: 518 : if(Array != crctx->WhichSubContext || (Array == crctx->WhichSubContext && Linear == pgc->ppc.arctx.PatArrangement ))  788 : : {  789 [ + + ][ + + ]: 518 : if((crctx->CurrPP == crctx->NumPP && CROptimize == crctx->CRType) || (crctx->CurrRepY == crctx->NumRepY - 1 && CRLinear == crctx->CRType))  [ + + ][ + + ]  790 : : {  791 : 190 : crctx->CurrPP = 0 ;  792 : 190 : crctx->CurrRepY = -1 ;  793 : 190 : crctx->EndPPptr = crctx->CurrPPptr ;  794 : 190 : crctx->CurrPPptr = crctx->StartPPptr ;  795 : :   796 [ + + ]: 190 : if(Normal == crctx->WhichSubContext)  797 : : {  798 : 53 : crctx->isPatternOver = true ;  799 : 53 : return -1 ;  800 : : }  801 : :   802 : 137 : pgc->current = crctx->CurrReprPtr ;  803 [ + + ][ - + ]: 137 : switch(crctx->WhichSubContext)  804 : : {  805 : 90 : case Optimize : status = ORGetPP(pgc, pp) ;  806 : 90 : break ;  807 : 47 : case Array : status = ARGetPP(pgc, pp);   808 : 47 : break ;  809 : 0 : default : status = 0 ;   810 : 137 : break ;  811 : : }  812 : :   813 : 137 : crctx->ReprPP.X0 = pp->X0 ;  814 : 137 : crctx->ReprPP.Y0 = pp->Y0 ;  815 : 137 : crctx->CurrReprPtr = pgc->current ;  816 : :   817 [ + + ]: 137 : if(-1 == status)  818 : : {  819 : 62 : crctx->isPatternOver = true ;  820 : 62 : pgc->current = crctx->StartReprPtr ;// Added by Ram on Aug 20 while coding SD  821 : 137 : return -1 ;  822 : : }  823 : :   824 : : } /\* If NumPP \*/  825 : :   826 [ + + ]: 403 : if(Normal != crctx->WhichSubContext)  827 : : {  828 : 323 : vpp = crctx->ReprPP ;  829 : : }   830 : : else  831 : : {  832 : 80 : vpp.X0 = crctx->ReprEP.X0 ;  833 : 80 : vpp.Y0 = crctx->ReprEP.Y0 ;  834 : : }  835 : :   836 [ + + ]: 403 : if(CROptimize == crctx->CRType)  837 : : {  838 : : // Fetch gPPi  839 [ + - ]: 326 : if(crctx->CurrPP <= crctx->NumPP)  840 : : {  841 [ + + ]: 326 : if(0 == crctx->CurrPP )  842 : : {  843 : 59 : vpp.X0 += crctx->gPP0.X0 ;  844 : 59 : vpp.Y0 += crctx->gPP0.Y0 ;  845 : : }  846 : : else   847 : : {  848 : : /\* For cr.or.x / cr.or.y \*/  849 : 267 : pp->X0 = crctx->gPP0.X0 ;  850 : 267 : pp->Y0 = crctx->gPP0.Y0 ;  851 : :   852 : 267 : crctx->CurrPPptr += vsbBufferToPP(crctx->CurrPPptr, crctx->PPDirection, pgc->X0Y0WordLength, pp) ;  853 : : // Should this gPPi go to the context ?  854 : 267 : gPPi.X0 = pp->X0 ;  855 : 267 : gPPi.Y0 = pp->Y0 ;  856 : :   857 : 267 : vpp.X0 += gPPi.X0 ;  858 : 267 : vpp.Y0 += gPPi.Y0 ;  859 : : }  860 : 326 : crctx->CurrPP++ ;  861 : : }  862 : : else  863 : : {  864 : 0 : crctx->isPatternOver = true ;  865 : 0 : return -1;  866 : : }   867 : : }  868 : : else // CRLinear  869 : : {  870 : : // Compute gPPi  871 [ + - ]: 77 : if(crctx->CurrRepY < crctx->NumRepY-1)  872 : : {  873 : 77 : crctx->CurrRepY++ ;  874 : :   875 : 77 : gPPi.X0 = crctx->gPP0.X0 + (crctx->CurrRepY \* crctx->PitchX) ;  876 : 77 : gPPi.Y0 = crctx->gPP0.Y0 + (crctx->CurrRepY \* crctx->PitchY) ;  877 : :   878 : :   879 : 77 : vpp.X0 += gPPi.X0 ;  880 : 77 : vpp.Y0 += gPPi.Y0 ;  881 : : }  882 : : else  883 : : {  884 : 0 : crctx->isPatternOver = true ;  885 : 0 : return -1 ;  886 : : }   887 : : }  888 : :   889 [ + - ][ - + ]: 403 : if ( 1.0F != pgc->TranslateFactor || pgc->mirror)  890 : : {  891 : 0 : if( 0 > TranslatePP(pgc, &vpp) )  892 : 403 : return -1 ;  893 : : }  894 : :   895 : 403 : dtoi(&vpp, pp) ;  896 : 403 : pgc->current = crctx->StartReprPtr ;  897 : : }  898 : :   899 : : }   900 : 518 : return status ;  901 : : }  902 : :   903 : 1215 : }// namespace nft |

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