INTERNATIONAL STANDARD

ISO/IEC 15444-17

First edition
2023-06

Information technology — JPEG 2000 image coding system —

Part 17: Extensions for coding of discontinuous media

Technologies de l'information — Système de codage d'images JPEG 2000 —

Partie 17: Extensions pour le codage des supports discontinus

Reference number
ISO/IEC 15444-17:2023(E)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by ITU-T (as ITU-T - T.816) and drafted in accordance with its editorial rules, in collaboration with Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO/IEC 15444 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

INTERNATIONAL STANDARD ISO/IEC 15444-17
RECOMMENDATION ITU-T T.816 (V1)

Information technology – JPEG 2000 image coding system: Extensions for coding of discontinuous media

Summary

Rec. ITU-T T.816 | ISO/IEC 15444-17 provides extensions of the scalable image coding tools described in Rec. ITU-T T.800 | ISO/IEC 15444-1 and Rec. ITU-T T.801 | ISO/IEC 15444-2, of two types. First, new wavelet-like image transforms known as "breakpoint-dependent" transforms are defined, whose underlying basis functions can be discontinuous at defined locations within the image component to which they are applied. Second, new scalable coding tools are described for a new type of image component known as a "breakpoint component", which provides a successively refinable and hierarchical description of the breakpoint locations used by the breakpoint-dependent transforms. Any non-initial component or components within a codestream conforming to this Recommendation | International Standard can be breakpoint components and any of the components in the codestream other than breakpoint components can use a breakpoint-dependent transform that depends upon one of the breakpoint components in the same codestream. These new tools together allow for the scalable coding of imagery that naturally exhibits strong discontinuities in the spatial domain. An important example of such imagery is depth maps.

This Recommendation was developed jointly with ISO/IEC JTC 1/SC 29/WG 1 (JPEG) and is common text with ISO/IEC 15444-17.

History

Edition	Recommendation	Approval	Study Group	Unique ID[*]
1.0	ITU-T T.816 (V1)	2023-02-13	16	11.1002/1000/15206

[*] To access the Recommendation, type the URL http://handle.itu.int/ in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at http://www.itu.int/ITU-T/ipr/.

CONTENTS

List of Tables

List of Figures

Page

Introduction

The JPEG 2000 core coding system specified in Rec. ITU-T T.800 | ISO/IEC 15444-1 and its extensions specified in Rec. ITU-T T.801 | ISO/IEC 15444-2 provide a suite of scalable coding technologies that are particularly suitable for photographic media, but less effective at coding media with hard discontinuities. An important example of such media is depth imagery, where each image sample is related to the length of the 3D line segment between the corresponding scene point and the camera. Depth imagery includes stereo disparity maps, where sample values are reciprocally related to depth. Another example of media with strong discontinuities is optical flow data, where each sample location is a two-dimensional vector. In these examples, discontinuities arise naturally at the boundaries of scene objects. Moreover, where this happens, intermediate values that might be obtained by bandlimited image resampling or interpolation operations have no physical meaning – i.e., they do not correspond to the depth or flow vector of any object in the original scene. The discrete wavelet transform (DWT) employed in JPEG 2000 is not well suited to the coding of such media, both from the perspective of coding efficiency and considering the nature of distortions that result when the wavelet sub-band samples are quantized.

To address these challenges, this Recommendation | International Standard introduces alternate "breakpoint-dependent" spatial wavelet transforms that are dependent on an auxiliary image component, known as a "breakpoint component." This Recommendation | International Standard also introduces scalable coding technologies for breakpoint components. Any non-initial component or components within the codestream can be designated as breakpoint components, allowing them to be used as the source of breakpoints for other components, or tiles thereof, which specify the use of breakpoint-dependent wavelet transforms.

This Recommendation | International Standard specifies two different types of breakpoint components, designated as "QuadBPT" and "TriBPT" components, with associated decoding and synthesis tools. Associated with the type of breakpoint component is a corresponding breakpoint-dependent wavelet transform, with its synthesis tools. The reconstruction procedures described in this Recommendation | International Standard produce individual sample values. In the TriBPT case, it is possible instead to directly reconstruct a deformable triangular mesh, whose complexity is related to the number of non-zero wavelet coefficients and the number of decoded breaks, which are identified here as "vertices." In each case, breakpoints introduce tears in the mesh. This feature can be valuable in computer graphics applications, where the mesh elements provide a more convenient description of the data than individual samples.

The normative material of this Recommendation | International Standard is contained within the main body together with Annex A. Additionally, Annex B describes ways of encapsulating breakpoint data within a linear file structure, that can be used as a source for encoding and a target for decoding procedures.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – JPEG 2000 image coding system: Extensions for coding of discontinuous media

1 Scope

This Recommendation | International Standard defines QuadBPT and TriBPT image components, collectively known as "breakpoint components", and specifies decoding and reconstruction procedures for recovering breakpoint component sample values from the codestream. This Recommendation | International Standard also specifies "breakpoint-dependent" spatial wavelet transforms that can be used in place of the transforms specified in Recommendation ITU-T T.800 | ISO/IEC 15444-1 for selected image components or tile-components. Extensions to the codestream syntax of Rec. ITU-T T.800 | ISO/IEC 15444-1 are specified to enable the identification of breakpoint components, of components that can use a breakpoint-dependent spatial wavelet transform, and the association of breakpoint components with such breakpoint-dependent wavelet transforms.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

– Recommendation ITU-T T.800 (2019) | ISO/IEC 15444-1:2019, *Information technology – JPEG 2000 image coding system: Core coding system*.

– Recommendation ITU-T T.801 (2021) | ISO/IEC 15444-2:2021, *Information technology – JPEG 2000 image coding system: Extensions*.

3 Definitions

3.1 Terms defined elsewhere

For the purposes of this Recommendation | International Standard, the terms and definitions given in Rec. ITU-T T.800 | ISO/IEC 15444-1 apply.

ITU, ISO and IEC maintain terminological databases for use in standardization at the following addresses:

– ITU Terms and definitions database: available at https://www.itu.int/go/terms

– ISO Online browsing platform: available at https://www.iso.org/obp

– IEC Electropedia: available at http://www.electropedia.org/

3.2 Terms defined in this Recommendation | International Standard

This Recommendation | International Standard defines the following terms:

3.2.1 2-span: Square configuration of width 2 and height 2, with 9 grid-points, such that the four corner grid-points all have coordinates that are divisible by 2.

3.2.2 4-span: 2×2 configuration of 2-spans (3.2.1), involving 25 grid-points, such that the four corner grid-points all have coordinates that are divisible by 4.

3.2.3 ambivalent break: Induced break (3.2.5) that has insufficient precision to determine whether the break occurs in the first or second half of the arc.

3.2.4 arc: Line segment connecting grid-points with even valued coordinates at any resolution of a breakpoint tile-component.

3.2.5 break: Explicitly decoded or inferred location within an arc (3.2.4) that indicates a boundary within an image component.

NOTE 1 – Breaks modify the behaviour of breakpoint-dependent transforms on that other image component.

NOTE 2 – An arc has at most one break.

3.2.6 breakpoint: Data structure consisting of break type, location and precision information for an arc (3.2.4).

NOTE – Type-0 breakpoints have no break at all.

3.2.7 breakpoint component: JPEG 2000 codestream image component that represents arc breakpoint (3.2.6) information.

3.2.8 breakpoint tile-component: All the breakpoints of a given tile, within a breakpoint component (3.2.7).

3.2.9 cell: 2×2 configuration of grid-points within a resolution of a breakpoint tile-component (3.2.8).

NOTE – Cells belong to a well-defined partition that is anchored at the global code-block anchor point.

3.2.10 CL band: The sole sub-band associated with each resolution of a breakpoint tile-component (3.2.8) other than the lowest resolution.

3.2.11 code-block anchor point: Origin of the coding partitions, which is one of the locations (0,0), (0,1), (1,0) or (1,1).

[SOURCE: Rec. ITU-T T.801 | ISO/IEC 15444-2]

3.2.12 directly induced break: Break (3.2.5) on an arc that is inferred from a break on a parent arc.

3.2.13 extrapolation qualifier: 2-bit quantity e_b which controls the way gradients are obtained for extrapolation within a TriBPT-dependent transformation.

3.2.14 indefinite break: Induced break (3.2.15) on an arc that is derived from one or more ambivalent breaks (3.2.3), such that there is insufficient precision to determine whether any break at all exists on the arc.

3.2.15 induced break: Break (3.2.5) on an arc that is inferred from breaks on other arcs.

3.2.16 induction block: Condition on an arc (3.2.4) that is explicitly recovered from the decoding of breakpoint code-blocks, indicating that no break (3.2.5) shall be induced on that arc.

3.2.17 non-root arc: Arc (3.2.4) that is not a root arc.

3.2.18 parent arc: Arc (3.2.4) at depth $d + 1$ in the breakpoint decomposition that contains an arc at depth d.

NOTE – At most two arcs at depth d can have the same parent at depth $d + 1$.

3.2.19 pass-complete: Code-block within a breakpoint component for which one or more coding passes are found within the codestream packets and the last such coding pass is identified as completing the code-block's representation via the packet header signalling mechanisms.

3.2.20 QuadBPT: Breakpoint (3.2.6) arrangement involving only horizontal and vertical arcs (3.2.4).

3.2.21 root arc: Arc (3.2.4) that is not contained within any arc projected from the next lower resolution of a breakpoint tile-component (3.2.8), regardless of whether that next lower resolution actually exists within the tile-component's resolution hierarchy.

3.2.22 spatially induced break: Induced break (3.2.15) that is inferred from breaks on non-parent arcs.

3.2.23 tick-point: Possible break (3.2.5) location along an arc.

3.2.24 TriBPT: Breakpoint (3.2.6) arrangement involving horizontal, vertical and diagonal arcs (3.2.4).

3.2.25 TriBPT-LR: TriBPT (3.2.24) breakpoint arrangement involving diagonal arcs that run from the top-left to the bottom-right of a 2-span within any resolution of a breakpoint tile-component (3.2.8).

3.2.26 TriBPT-RL: TriBPT (3.2.24) breakpoint arrangement involving diagonal arcs that run from the top-right to the bottom-left of a 2-span within any resolution of a breakpoint tile-component (3.2.8).

3.2.27 vertex: Explicitly coded break (3.2.5) location.

3.2.28 zero-complete: Code-block within a breakpoint component (3.2.7) that makes no contribution to any codestream packet and is identified as complete by the first packet header of its precinct.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

BD-IDWT	Breakpoint-Dependent Inverse Discrete Wavelet Transformation

BGP	Base Grid Point
BIL	Break Inducing Line
BIP	Base Intersection Point
BLN	Base Line
BSA	Base Arc
CBAP	Code-Block Anchor Point
DIB	Directly Induced Breaks
HDT	Hierarchical Data Type
HGP	Horizontal Grid-Point
MPS	More Probable Symbol
RAC	Round Away from Centre
RTN	Round to Nearest
SBR	Source Breaks
SPB	Spatially induced Break
TPS	Tick-Points
VMR	Vertex Mapping Rule
VRT	Vertex
WDA	Wedge Area

5 Conventions

This Recommendation | International Standard uses the following conventions:

- **CBAP** (z_x, z_y) – code-block anchor point
- **MAX_WDG** – maximum search distance for the gradient extrapolation algorithm used during the TriBPT-dependent prediction step associated with a spatially induced arc.
- **BPT_INTER** – binary flag that is 1 if code-blocks of a breakpoint component use the inter-band coding mode and 0 if the code-blocks of a breakpoint component are coded without reference to any other code-block data.

6 Conformance

6.1 Codestream conformance

A codestream conforming to this Specification shall conform to Annex A.

6.2 Decoder

A decoder conforming to this Specification shall process a codestream that conforms to this Specification in the manner specified in Rec. ITU-T T.800 | ISO/IEC 15444-1 together with any additional signalled capability specified in this Specification, with the exception of breakpoint components and tile-components that identify the use of a breakpoint-dependent spatial wavelet transform, in which case the following shall apply:

- breakpoint components shall have the structure specified in clause 7;
- tile-components that use a breakpoint-dependent spatial wavelet transform shall be processed in accordance with clause 8; and
- breakpoint components shall be reconstructed from breakpoint code-blocks in accordance with clause 10, where breakpoint code-blocks are decoded in accordance with clause 9.

This Recommendation | International Standard is compatible with the coding technologies specified in both Rec. ITU-T T.800 | ISO/IEC 15444-1 and Rec. ITU-T T.801 | ISO/IEC 15444-2. However:

1. Breakpoint components, as specified herein, are not compatible with the multiple component transformations specified in Rec. ITU-T T.800 | ISO/IEC 15444-1 and Rec. ITU-T T.801 | ISO/IEC 15444-2, nor are they compatible with the non-linear transformations specified in Rec. ITU-T T.801 | ISO/IEC 15444-2; and

2. Breakpoint code-blocks, as specified herein, are not compatible with the region of interest coding and extraction techniques specified in Rec. ITU-T T.800 | ISO/IEC 15444-1 and Rec. ITU-T T.801 | ISO/IEC 15444-2.

7 Breakpoint component structure

7.1 Breakpoint components and the reference grid

Breakpoint components have a "hierarchical data type", meaning that they are described at multiple resolutions, in a dyadic hierarchy. Breakpoint components are identified via the HDT marker segment defined in A.4. In particular, component c is a breakpoint component if Ihdt^c lies in the range 1 to 3.

Breakpoint component c is defined with respect to the same reference grid as other JPEG 2000 components, in terms of the global image dimensions (Xsiz, Ysiz), image offset (XOsiz, YOsiz) and sampling factors (XRsiz^c, YRsiz^c) that are recorded in the SIZ marker segment. The *grid-points* of component c consist of all integer coordinates in the rectangle with upper left hand corner at (x_o, y_o) and lower right hand corner at $(x_l - 1, y_l - 1)$, where

$$x_o = \left\lceil \frac{\text{XOsiz}}{X\text{Rsiz}^c} \right\rceil,\ x_l = \left\lceil \frac{\text{Xsiz}}{X\text{Rsiz}^c} \right\rceil,\ y_o = \left\lceil \frac{\text{YOsiz}}{Y\text{Rsiz}^c} \right\rceil,\ y_l = \left\lceil \frac{\text{Ysiz}}{Y\text{Rsiz}^c} \right\rceil. \tag{1}$$

These formulae are identical to the equations for x_o, x_l, y_o and y_l in Rec. ITU-T T.800 | ISO/IEC 15444-1, but note that the term *grid-points* is used here for the locations of a breakpoint component, rather than *samples*.

As with other components, breakpoint components are partitioned into tile-components, following the partitioning of the reference grid into tiles, so that the grid-points of a breakpoint tile-component having component index c belong to the rectangle with upper left corner at (tcx_o, tcy_o) and lower right hand corner at $(tcx_l - 1, tcy_l - 1)$, where

$$tcx_o = \left\lceil \frac{tx_o}{X\text{Rsiz}^c} \right\rceil,\ tcx_l = \left\lceil \frac{tx_l}{X\text{Rsiz}^c} \right\rceil,\ tcy_o = \left\lceil \frac{ty_o}{Y\text{Rsiz}^c} \right\rceil,\ tcy_l = \left\lceil \frac{ty_l}{Y\text{Rsiz}^c} \right\rceil, \tag{2}$$

where the tile in question occies the rectangle with top left corner at (tx_o, ty_o) and lower right corner at $(tx_l - 1, ty_l - 1)$ on the reference grid. These formulae are identical to the equations for tcx_o, tcx_l, tcy_o and tcy_l in Rec. ITU-T T.800 | ISO/IEC 15444-1, except that the breakpoint tile-component's locations are identified as *grid-points* rather than *samples*.

Like other JPEG 2000 components, each breakpoint tile-component has an associated number of *decomposition levels* N_L, that is identified via the applicable COD or COC marker segment, imputing the component with $N_L + 1$ distinct resolution levels, denoted $r = 0,1, \dots, N_L$; these are the resolutions of the hierarchical breakpoint representation. Level $r = N_L$ is the full resolution of the tile-component, while $r = 0$ is the lowest resolution, also known as the tile-component's LL band. The *grid-points* associated with the resolution r of a breakpoint tile-component correspond to the integer-valued coordinates within the rectangle with upper left corner at (trx_o, try_o) and lower right hand corner at $(trx_l - 1, try_l - 1)$, where

$$trx_o = \left\lceil \frac{tcx_o}{2^{N_L - r}} \right\rceil,\ trx_l = \left\lceil \frac{tcx_l}{2^{N_L - r}} \right\rceil,\ try_o = \left\lceil \frac{tcy_o}{2^{N_L - r}} \right\rceil,\ try_l = \left\lceil \frac{tcy_l}{2^{N_L - r}} \right\rceil. \tag{3}$$

Each resolution r has a corresponding depth d within the hierarchical representation. The highest resolution $r = N_L$ corresponds to depth $d = 1$, while the resolution $r = 1$ corresponds to depth $d = N_L$. Unlike other JPEG 2000 components, for which the LL band at resolution $r = 0$ has the same depth $d = N_L$ as resolution 1, the LL band of a breakpoint component has depth $d = N_L + 1$.

7.2 Division of breakpoint resolutions into cells, arcs and the CL band

Breakpoint components, their tile-components and their resolutions, do not have samples, but their grid-points may correspond to the sample locations of other components. Instead of samples, the coded information associated with a breakpoint component belongs to *arcs*.

There are two fundamental arrangements for these arcs, known as QuadBPT and TriBPT, and two mirror-image variants for the TriBPT arrangement, known as TriBPT-LR and TriBPT-RL. All three arrangements are defined in terms of 2×2 cells within each resolution of a breakpoint tile-component. To understand these arrangements, it is helpful to start by introducing the concept of a *2-span*, which consists of 9 grid-points. As shown in Figure 1, each arc runs between grid-points on the boundary of its 2-span. The central grid-point of a 2-span always has odd-valued coordinates in the resolution to which it belongs, while the corners of each 2-span have even-valued coordinates. Within any given resolution arcs have length 2, if we measure the length of diagonal arcs in the vertical direction only. This is equivalent to a length of 2^d at the component's full resolution.

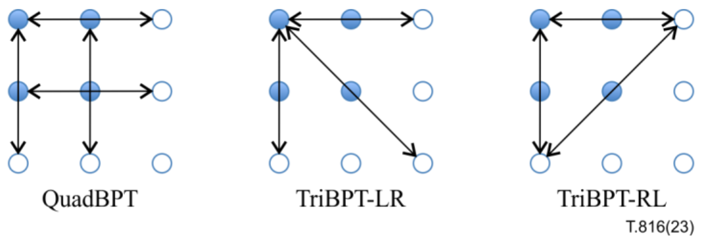


Figure 1 – Arc arrangements within a single 2-span. A 2 × 2 cell is shown here with solid grid-points. See Figure 2 for other geometric relationships between cells and 2-spans

Each 2-span is associated with a cell that contains only 4 unique grid-points. The grid-points of each resolution are partitioned into cells, such that the upper left corner of each cell has coordinates of the form $(2c_x + z_x, 2c_y + z_y)$ and (z_x, z_y) corresponds to the code-block anchor point (CBAP) identified by bits 3 and 4 of the Scod parameter in the extended COD syntax specified in Rec. ITU-T T.801 | ISO/IEC 15444-2. The four possible combinations of CBAP coordinates, $(z_x, z_y) = (0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$, lead to four different cell partitions, that enable geometric manipulation (flipping and rotation) in the compressed domain. Figure 2 illustrates the four different relationships between cells and their 2-spans that result from the different anchor-points.

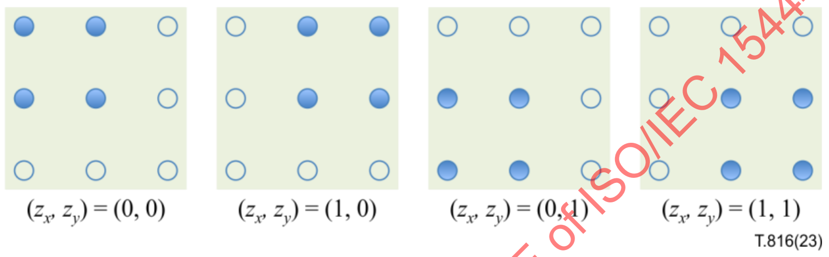


Figure 2 – Cell-span geometry for each of the four possible code-block anchor points

NOTE 1 – The code-block anchor point is also the anchor point for the precinct partitions in JPEG 2000. This means that the cell partition always aligns with the precinct partition for the same resolution of a tile-component.

The cells associated with a resolution of a breakpoint tile-component have coordinates (c_x, c_y), covering the rectangle with upper left corner at (clx_o, cly_o) and lower right hand corner at $(clx_l - 1, cly_l - 1)$, where

$$clx_o = \left\lfloor \frac{trx_o - z_x}{2} \right\rfloor,\ clx_l = \left\lceil \frac{trx_l - z_x}{2} \right\rceil,\ cly_o = \left\lfloor \frac{try_o - z_y}{2} \right\rfloor,\ cly_l = \left\lceil \frac{try_l - z_y}{2} \right\rceil. \qquad (4)$$

That is, the resolution's cells consist of all 2 × 2 elements from the partition that intersect with the resolution's region of support, regardless of whether the intersection involves 1, 2 or all 4 grid-points.

Ignoring the LL band (resolution 0) for the moment, each other resolution $r = 1, \dots, N_L$ of a breakpoint tile-component is assigned a single "detail band", similar to the LH, HL and HH sub-bands described in Rec. ITU-T T.800 | ISO/IEC 15444-1. This single detail band is denoted the CL band. Each element of a CL band is a 2 × 2 cell (or a piece thereof, if the tile-component boundaries dissect the cell). The hierarchical representation of the breakpoint tile-component then consists of one LL band and N_L CL bands. Each CL band is $clx_l - clx_o$ cells wide by $cly_l - cly_o$ cells high, where these quantities are derived from the corresponding resolution's dimensions using Formula (4). The set of cells that constitute a CL band determine the properties of its code-blocks and precincts, as explained next.

The dimensions of the LL band are expressed in grid-points rather than cells, and are identical to the dimensions of resolution 0, as derived from Formula (3) with $r = 0$. The LL band of a breakpoint tile-component is used to record breakpoints that would have been found in a CL band at depth $N_L + 1$, except that they are encoded with extra information to reveal whether or not the breakpoints should be considered to have been induced from even lower (uncoded) levels in the hierarchy. This is explained further in clause 9.

NOTE 2 – Unlike regular image components, the LL bands of breakpoint tile-components are commonly devoid of any significant information – i.e., they need signal no breaks at all – but this might not always be the case.

7.3 Division of breakpoint resolutions into precincts and code-blocks

Like all JPEG 2000 components, each resolution of a breakpoint component is partitioned into precincts, and the associated bands are partitioned into code-blocks, such that each code-block belongs to exactly one precinct. The number of precincts which span a breakpoint tile-component at resolution r is given by

$$precinctswide = \begin{cases} \left\lceil \frac{trx_l - z_x}{2^{\mathrm{PPx}}} \right\rceil - \left\lfloor \frac{trx_o - z_x}{2^{\mathrm{PPx}}} \right\rfloor & trx_l > trx_o \\ 0 & trx_l = trx_o \end{cases},$$

$$precinctshigh = \begin{cases} \left\lceil \frac{try_l - z_y}{2^{\mathrm{PPy}}} \right\rceil - \left\lfloor \frac{try_o - z_y}{2^{\mathrm{PPy}}} \right\rfloor & try_l > try_o \\ 0 & try_l = try_o \end{cases},$$

where PPx and PPy are signalled in COD and COC marker segments. These formulae for *precinctswide* and *precinctshigh* are the same as those found in Rec. ITU-T T.800 | ISO/IEC 15444-1, modified to account for the possibility of non-zero CBAP coordinates (z_x, z_y). However, the values of PPx and PPy associated with breakpoint components shall be no smaller than 1. Moreover, for resolutions $r > 0$, the values of PPx and PPy for breakpoint components with TriBPT arrangements, shall be no smaller than 2. These constraints are summarized in Table 1.

Table 1 – Constraints on PPx and PPy precinct size parameters breakpoint components

Breakpoint Arrangement	Resolution	PPx	PPy
QuadBPT	$r = 0$	≥ 1	≥ 1
QuadBPT	$r > 0$	≥ 1	≥ 1
TriLR or TriRL	$r = 0$	≥ 1	≥ 1
TriLR or TriRL	$r > 0$	≥ 2	≥ 2

Each CL band is divided into code-blocks which are organized into precincts in exactly the same way as regular sub-bands, except that each precinct for a breakpoint tile component at resolution $r > 0$ contains code-blocks from just the one CL band at that resolution, as opposed to HL, LH and HH sub-bands. Each nominal code-block at resolution r contains a rectangular array of cells that is $2^{\mathrm{xcb'}}$ cells wide and $2^{\mathrm{ycb'}}$ cells high, except at the boundaries of the tile-component, where code-blocks contain only those cells that intersect with the CL band region, as given by Formula (4). The code-block partition is anchored at the CBAP location (z_x, z_y) introduced above (code-block anchor point), and consists of all code-blocks that have a non-empty intersection with the CL band region. This means that each CL code-block's cell indices occupy a rectangle whose upper-left corner has coordinates (bx_o, by_o) and whose lower-right corner has coordinates $(bx_l - 1, by_l - 1)$, where

$$bx_o = \max\{k_x 2^{\mathrm{xcb'}}, clx_o\}, \quad bx_l = \min\{(k_x + 1)2^{\mathrm{xcb'}}, clx_l\},$$

$$by_o = \max\{k_y 2^{\mathrm{ycb'}}, cly_o\}, \quad by_l = \min\{(k_y + 1)2^{\mathrm{ycb'}}, cly_l\},$$

(k_x, k_y) are the integer-valued code-block indices, and the set of code-blocks corresponds to all (k_x, k_y) such that $bx_l > bx_o$ and $by_l > by_o$. In the above,

$$\mathrm{xcb}' = \begin{cases} \min\{\mathrm{xcb}, \mathrm{PPx} - 1\} & r > 0 \\ \min\{\mathrm{xcb}, \mathrm{PPx}\} & r = 0 \end{cases} \quad \mathrm{ycb}' = \begin{cases} \min\{\mathrm{ycb}, \mathrm{PPy} - 1\} & r > 0 \\ \min\{\mathrm{ycb}, \mathrm{PPy}\} & r = 0 \end{cases}, \tag{5}$$

where xcb and ycb are signalled in COD and COC marker segments. These formulae are identical to the formulae for xcb′ and ycb′ in Rec. ITU-T T.800 | ISO/IEC 15444-1.

NOTE 1 – The CL band region, given by Formula (4), can differ slightly from the regions associated with LH, HL or HH bands of a non-breakpoint tile-component at the same resolution, as given by determined in accordance with Rec. ITU-T T.800 | ISO/IEC 15444-1. It turns out that every precinct of a breakpoint tile-component is non-empty, in the sense that it contains at least one code-block that contains at least one cell. By contrast, it is possible for a precinct of a non-breakpoint tile-component to be empty, having no code-block from any LH, HL or HH sub-band, as explained in Rec. ITU-T T.800 | ISO/IEC 15444-1.

The LL band of a breakpoint tile-component is also partitioned into code-blocks, where the partition is anchored at the CBAP location (z_x, z_y), with code-blocks of width $2^{\mathrm{xcb'}}$ and height $2^{\mathrm{ycb'}}$, where xcb' and ycb' are still found using Formula (5). However, LL code-blocks are measured in grid-points rather than cells, and code-blocks on the boundary of the tile-component contain only those grid-points that belong to the region associated with resolution 0, as given by Formula (3) with $r = 0$. Specifically, each LL code-block's grid-points have indices that lie within a rectangle whose upper-left corner has coordinates (Bx_o, By_o) and whose lower-right corner has coordinates $(Bx_l - 1, By_l - 1)$, where

$$Bx_o = \max\{z_x + k_x 2^{\mathrm{xcb'}}, trx_o\}, \quad Bx_l = \min\{z_x + (k_x + 1)2^{\mathrm{xcb'}}, trx_l\},$$

$$By_o = \max\{z_y + k_y 2^{\mathrm{ycb'}}, try_o\}, \quad By_l = \min\{z_y + (k_y + 1)2^{\mathrm{ycb'}}, try_l\},$$

(k_x, k_y) are the integer-valued code-block indices, and the set of LL code-blocks corresponds to all (k_x, k_y) such that $Bx_l > Bx_o$ and $By_l > By_o$.

NOTE 2 – The number and sizes of LL code-blocks from a breakpoint tile-component are identical to those of an identically dimensioned non-breakpoint component with the same N_L, PPx, PPy, xcb and ycb parameters.

Even though LL code-blocks of a breakpoint tile-component are dimensioned in terms of grid-points, their coded information is based on 2 × 2 cells. Each LL code-block's grid-points are separately partitioned into cells, such that the upper-left corner of the top-left cell in the code-block, and the lower-right corner of the bottom-right cell in the code-block have coordinates

$$\left(2LLx_o + z_x, 2LLy_o + z_y\right) \text{ and } \left(2LLx_l + z_x - 1, 2LLy_l + z_y - 1\right),$$

respectively, where the inclusive lower and exclusive upper bounds on the block's cell coordinates are

$$LLx_o = \left\lfloor \frac{Bx_o - z_x}{2} \right\rfloor, LLy_o = \left\lfloor \frac{By_o - z_y}{2} \right\rfloor, LLx_l = \left\lceil \frac{Bx_l - z_x}{2} \right\rceil \text{ and } LLy_l = \left\lceil \frac{By_l - z_y}{2} \right\rceil$$

This means that the left half of each cell on the left boundary of the LL code-block lies outside the block if $Bx_o \bmod 2 \neq z_x$, a condition that can only occur for code-blocks on the left edge of the LL band. Similarly, the upper half of each cell on the top boundary of the code-block lies outside the block if $By_o \bmod 2 \neq z_y$, a condition that can only occur for code-blocks on the upper edge of the LL band. The right half of cells on the right edge of a code-block will lie outside the block if $Bx_l \bmod 2 \neq z_x$, while the lower half of cells on the bottom edge of a code-block will lie outside if $By_l \bmod 2 \neq z_y$; these conditions can only occur for code-blocks on the right and bottom edges of the LL band, respectively. Regardless of whether all grid-points of a cell lie within the code-block to which it belongs, the block decoding procedures described in 9 regard all arcs associated with such grid-points as if they belong to the block.

The number of cells that are found within any code-block is given by

$$W_{\text{blk}} = bx_l - bx_o \text{ and } H_{\text{blk}} = by_l - by_o,$$

where bx_o, by_o, bx_l and by_l become LLx_o, LLy_o, LLx_l and LLy_l for LL band code-blocks.

7.4 Root and non-root arc breakpoint associations

Breakpoints are associated with the grid-points of a 2 × 2 cell, and serve to describe possible breaks along one of the arcs that span the corresponding 2-span.

For the QuadBPT arrangement shown in Figure 1, there are four breakpoints per cell, as follows:

- Horizontal *non-root arc* breakpoints are associated with each of a resolution's grid-points (p_x, p_y) for which p_x is odd and p_y is even, serving to describe possible breaks along the horizontal arc that is centred at (p_x, p_y).
- Vertical *non-root arc* breakpoints are associated with each of a resolution's grid-points (p_x, p_y) for which p_x is even and p_y is odd, serving to describe possible breaks along the vertical arc that is centred at (p_x, p_y).
- Horizontal *root arc* breakpoints and vertical *root arc* breakpoints are both associated with each of a resolution's grid-points (p_x, p_y) for which p_x and p_y are both odd, serving to describe possible breaks along the horizontal and vertical arcs that are centred at (p_x, p_y).

The term *non-root arc* here is used to describe an arc that shares an end-point with an arc of the same orientation at the next lower resolution (whether it exists or not); each non-root arc thus corresponds to one half of an arc at the lower resolution. By contrast, the *root arcs* are those that are not subsets of any arc at lower resolutions.

For the TriBPT arrangements shown in Figure 1, there are three breakpoints per cell, as follows:

- Horizontal arc breakpoints are associated with each of a resolution's grid-points (p_x, p_y) for which p_x is odd and p_y is even, serving to describe possible breaks along the horizontal arc that is centred at (p_x, p_y).
- Vertical arc breakpoints are associated with each of a resolution's grid-points (p_x, p_y) for which p_x is even and p_y is odd, serving to describe possible breaks along the vertical arc that is centred at (p_x, p_y).
- Diagonal arc breakpoints are associated with each of a resolution's grid-points (p_x, p_y) for which p_x and p_y are both odd, serving to describe possible breaks along the diagonal arc that is centred at (p_x, p_y). These diagonal arcs run from the top-left to bottom-right corners of the 2-span in the TriBPT-LR case, and from the top-right to the bottom-left corners of the 2-span in the TriBPT-RL case.

The TriBPT arrangements also have root and non-root arcs that can be observed across four 2-spans, known collectively as a *4-span*. The corners of a 4-span correspond to grid-points whose coordinates are divisible by 4. Figure 3 identifies the root and non-root arcs for all configurations.

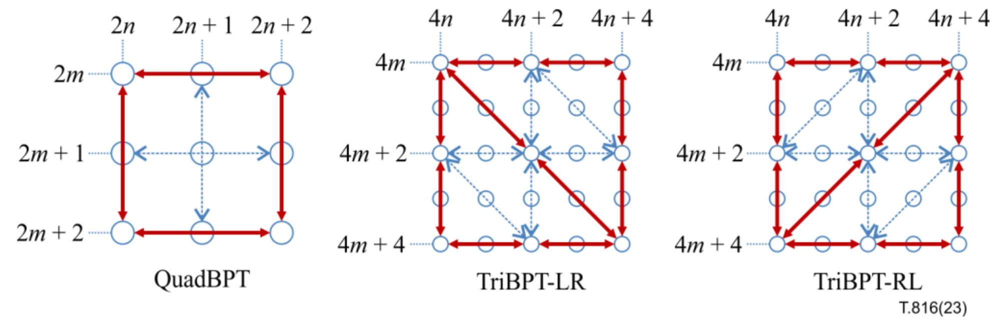


Figure 3 – Root arc (dotted lines) and non-root arcs (solid lines)

7.5 Breakpoint values and vertices

Each breakpoint tile-component has an associated parameter F_B that controls the cardinality of the set of possible break locations on each arc. Each arc at resolution r has length 2 in its own resolution, which is equivalent to a length of 2^d at the full resolution of the tile-component, where

$$d = N_L + 1 - r,$$

and each such arc has 2^{d+F_B} possible break locations, of the form

$$breakposition = (2k_b + 1)2^{-(F_B+1)} \tag{6}$$

Here, k_b is an integer in the range $0 \leq k_b < 2^{d+F_B}$ for an arc b. These possible break locations, indexed by k_b, are known here as *tick-points*. Note that diagonal arcs and their tick-points are projected onto the vertical axis for measurement purposes, so their lengths are still equal to 2^d.

The separation between tick-points, also known as the *tick-interval*, is 2^{-F_B}. The smallest possible value for F_B is 0, in which case the tick-interval is 1 and arcs at the finest resolution ($r = N_L$, $d = 1$) can have breaks at only two possible locations, corresponding the centres of the first and second halves of the arc. This is minimally sufficient to allow the breakpoints to be used in guiding the prediction processes of a breakpoint-dependent transform, as discussed in clause 8. Larger values of F_B allow breaks to be more precisely located along an arc. Figure 4 illustrates the tick-points resulting from a typical value of $F_B = 2$, for the case of the QuadBPT arrangement, at the finest resolution level.

NOTE – The F_B parameter can be interpreted as the number of "fraction bits" associated with the break location index k_b.

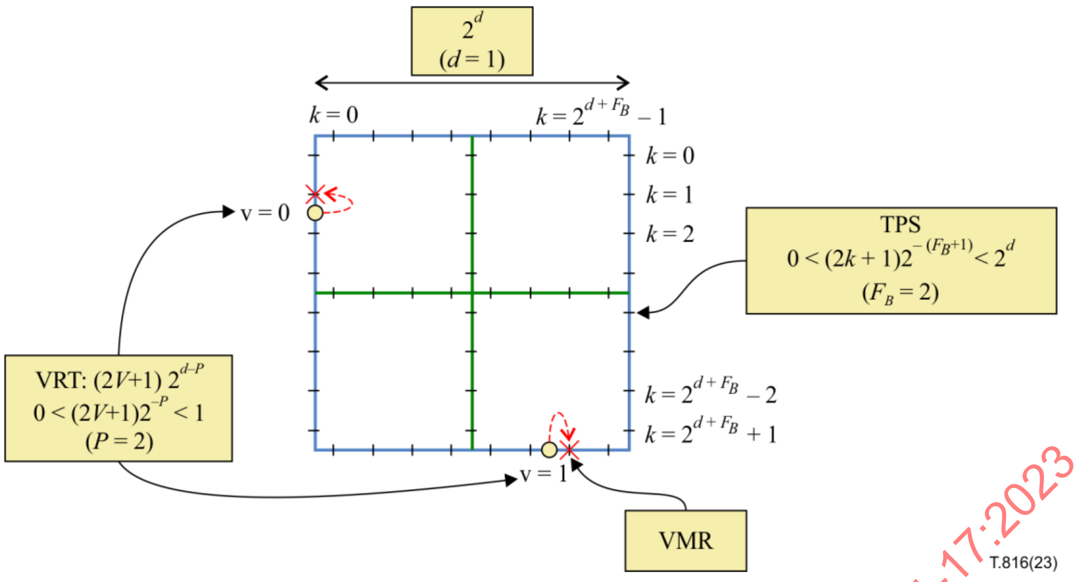
In addition to the break index k_b, each arc has a *break-type* value t_b in the range 0 to 3 and an integer *break-precision* value P_b that relates to the amount of information derived from the decoding of breakpoint code-blocks, from which breaks are inferred. For the TriBPT arrangement, breakpoints are also associated with an *extrapolation-qualifier* e_b that conveys information for performing extrapolation for arcs containing directly induced ($t_b = 2$) or spatially induced ($t_b = 1$) breaks. For all other break-type values the e_b parameter is irrelevant and can be ignored.

In following discussions, for the QuadBPT arrangement the parameters of a breakpoint on an arc b are denoted by (t_b, P_b, k_b) while for the TriBPT arrangement the breakpoint parameters are given by (t_b, P_b, k_b, e_b). Each of these parameters and the values that they can represent are discussed below.

Break-type values are as follows:

- $t_b = 3$ means that there is an explicitly coded break on the arc, known as a *vertex*; vertices are the significant values recovered from decoded breakpoint code-blocks. Vertex decoding is discussed in clause 9.
- $t_b = 2$ means that there is a break on the arc that is directly induced from a vertex or another directly induced break at the next lower resolution; direct induction is possible only for non-root arc breakpoints. Direct induction is discussed in 10.2.
- $t_b = 1$ means that there is a spatially induced break on the arc; spatial induction occurs when breaks are induced onto root-arcs based on breaks found in surrounding arcs at the same resolution, after which the condition is passed on recursively (through direct induction) to sub-arcs in finer resolutions. Spatial induction is discussed in 10.3 and 10.4.
- $t_b = 0$ means that there is no break on the arc.

The meaning of the break-precision value P_b depends upon the break-type t_b, as discussed next.



VRT	vertex
TPS	tick-points
VMR	vertex mapping rule of mapping vertex to nearest tick-point "away from arc centre"

NOTE – This figure depicts the case where d = 1, P = precision bits = 2 and F_B = 2 with arc length given by 2^d.

Figure 4 – Possible break locations (tick-points) at the finest resolution, with $F_B = 2$, shown for the QuadBPT case. Also shows the mapping of vertices with precision P_b to tick-points

Vertex Precision: For vertices ($t_b = 3$), the break-precision P_b shall satisfy

$$1 \le P_b - 1 \le d + F_B \tag{7}$$

where $P_b - 1$ is the number of bits of the vertex V_b that were actually decoded. The value $(2V_b + 1)$ is a P_b -bit unsigned integer, and the relationship between the tick-point k_b and the decoded vertex value V_b is as follows:

$$k_b = (2V_b + 1)2^{d+F_B-P_b} + \begin{cases} 0, & \text{if } (2V_b + 1)2^{-P_b} > \frac{1}{2} \\ -1, & \text{if } (2V_b + 1)2^{-P_b} < \frac{1}{2} \end{cases} \tag{8}$$

Figure 4 illustrates this mapping of the $(P_b - 1)$ bit vertex value V_b to tick-point k_b. The quantity $(2V_b + 1)2^{-P_b} \in (0,1)$ can be interpreted as a relative vertex position. As seen in the figure, the role of the offset on the right hand side of Formula (8) is to round this relative position outwards (away from the centre of the arc at relative position ½) to the nearest available tick-point.

Induced break precision: Directly induced breaks ($t_b = 2$) are obtained on non-root arcs, from vertices or other directly induced breaks at the next lower resolution, by decrementing P_b and discarding the most significant bit of k_b. Specifically, using the term "parent" to identify the corresponding arc in the next lower resolution,

$$P_{\text{induced}} = \max\{1, P_{\text{parent}} - 1\},\ k_{\text{induced}} = k_{\text{parent}} \bmod 2^{d+F_B-1} \tag{9}$$

Spatially induced breaks ($t_b = 1$) are derived from breaks on surrounding arcs at the same resolution, and their precision is the smallest value of P_b associated with those surrounding arcs whose breaks are involved in the induction.

Induced breaks with $P_b = 1$ are said to be *ambivalent*, since insufficient information is preserved to determine whether the break occurs in the first or second half of the arc. Both directly and spatially induced breaks can be ambivalent.

The special value $P_b = 0$ is used to identify *indefinite* breaks; these are induced breaks whose existence is uncertain because one or more of the arcs used in the induction process has break-precision $P_b \le 1$ that is too small to determine whether a break should exist. Both directly and spatially induced breaks can be indefinite.

No-break precision: When $t_b = 0$, meaning that there is no break on an arc, the value of k_b is meaningless, but the parameter P_b can be either 0 or 1. The combination $t_b = 0$, $P_b = 1$ means that the arc contains an *induction block*. Induction blocks prevent any break from being induced on the arc. Induction blocks are derived from decoding breakpoint code-blocks.

The extrapolation-qualifier e_b is comprised of two binary flags $e_{b,0}$ and $e_{b,1}$, such that $e_b = (e_{b,1}, e_{b,0})$. A break divides an arc into two parts and the binary value of $e_{b,0}$ conveys information required for performing extrapolation for the first part while $e_{b,1}$ corresponds to information for the second part of the arc. These binary flags have specific and distinct meaning for directly induced ($t_b = 2$) and spatially induced ($t_b = 1$) breaks for the TriBPT arrangement as detailed below.

For a directly induced break, $e_{b,0}$ = GRAD1_FLAG $_{induced}$ and $e_{b,1}$ = GRAD2_FLAG $_{induced}$. The predict step for an arc containing a directly induced break is conditional on these flags as described in 8.2.4. The values for GRAD1_FLAG $_{induced}$ and GRAD2_FLAG $_{induced}$ can be derived during the breakpoint synthesis stage and this is detailed in 10.2.2.

If an arc contains a spatially induced break, then $e_{b,0}$ = DRCTN1_FLAG $_{spatial}$ and $e_{b,1}$ = DRCTN2_FLAG $_{spatial}$. As detailed in 8.2.5, these binary flags signify a direction of search for an external gradient and are utilized in the predict step for arcs containing spatially induced breaks. The values for DRCTN1_FLAG $_{spatial}$ and DRCTN2_FLAG $_{spatial}$ can be determined from the 4-span triangle and inducing line geometry as explained in 10.4.4. For both flags a value of 1 indicates a search in the increasing x-coordinate direction for horizontal arcs and in the increasing y-coordinate direction for vertical and diagonal arcs. Conversely, a value of 0 indicates a search in the decreasing x-coordinate direction for horizontal arcs and in the decreasing y-coordinate direction for vertical and diagonal arcs.

8 Breakpoint-dependent spatial wavelet transformation

8.1 Overview

To perform breakpoint-dependent inverse discrete wavelet transformation (BD-IDWT), this Recommendation | International Standard uses the synthesized breakpoints, as described in clause 10 , to adapt a two-dimensional reconstruction of sub-band samples obtained by processing the codestream in accordance with Rec. ITU-T T.800 | ISO/IEC 15444-1 together with any additional signalled capability.

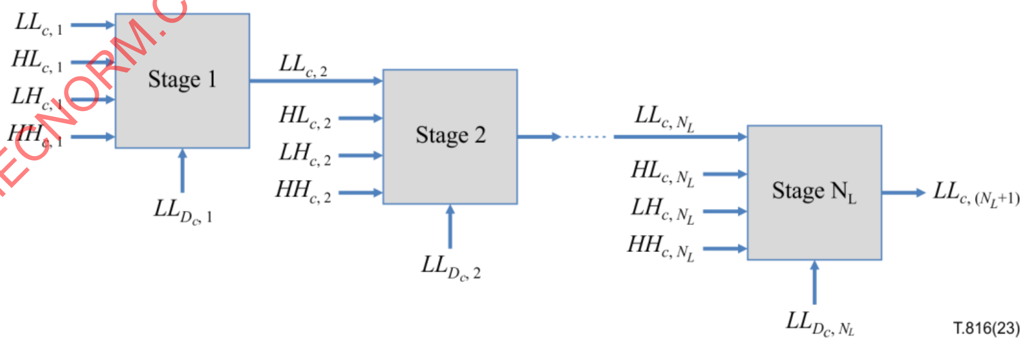
The BD-IDWT is performed in stages for a given tile t of a non-breakpoint component with index c, progressing from coarse to fine resolutions $r = 1,2, \dots, N_L$. The BD-IDWT uses spatial sub-band samples found within that tile-component, that are obtained by processing the codestream in accordance with Rec. ITU-T T.800 | ISO/IEC 15444-1, together with any additional signalled capability.

The BD-IDWT also uses breakpoint data from the same tile t of a separate breakpoint component with index D_c, where the value D_c is communicated via the SEcod field specified in Annex A.

Each stage r utilizes the four spatial sub-bands $LL_{c,r}, HL_{c,r}, LH_{c,r}, HH_{c,r}$ that belong to resolution r of the non-breakpoint tile-component indexed by c, together with a corresponding breakpoint band $LL_{D_c,r}$ which contains all breakpoint information at resolution r for the designated breakpoint tile-component indexed by D_c. The breakpoint data provided by $LL_{D_c,r}$ corresponds to the (partially) reconstructed breakpoints that appear at the same resolution r as determined using the reconstruction procedure in clause 10. Stage r uses this information to synthesize a sub-band $LL_{c,(r+1)}$, for use in the next stage, if any. The iterative procedure is illustrated in

$LL_{D_c,i}$ denotes the breakpoint LL band for resolution r=i.

Figure 5 where the output of the final stage $LL_{c,(N_L+1)}$ constitutes the decoded samples of the tile-component c.



$LL_{D_c,i}$ denotes the breakpoint LL band for resolution r=i.

Figure 5 – Stages of the breakpoint dependent inverse discrete wavelet transform

In the simplest case, where both components have the same number of decomposition levels N_L and identical sampling factors XRsiz^i and YRsiz^i reconstructed breakpoint data for resolution r is combined directly with the four spatial sub-bands $LL_{c,r}, HL_{c,r}, LH_{c,r}, HH_{c,r}$ to perform the reconstruction at stage $r = 1,2, \dots, N_L$.

If the breakpoint component and breakpoint-dependent component have different sampling factors XRsiz^i and YRsiz^i, the transformation process is not well defined. Some decoders might be capable of producing meaningful results when this happens, but this Recommendation | International Standard does not define the behaviour under such circumstances.

If the breakpoint component provides Δ_L more decomposition levels N_L than the breakpoint-dependent component, the transformation processes described here shall be applied after renumbering the resolutions of the breakpoint component so that original resolution r is interpreted as resolution $r - \Delta_L$. This effectively discards the unnecessary lower resolutions of breakpoint information.

If the breakpoint component provides only $N_L - 1$ decomposition levels, for the breakpoint-dependent component's N_L decomposition levels, the transformation processes described here shall be performed after renumbering the resolutions of the breakpoint component so that original resolution r is interpreted as resolution $r + 1$. This leaves the breakpoint component's LL band at resolution 0 providing the breakpoint data for the first stage of the transform at $r = 1$. If the breakpoint component provides even fewer decomposition levels, the transformation process is not defined by this Recommendation | International Standard.

Each stage of the BD-IDWT is performed in two steps. First the four spatial sub-bands $LL_{c,r}, HL_{c,r}, LH_{c,r}, HH_{c,r}$ are subjected to a mapping procedure 2D_GP_MAPPING that produces a single interleaved output band $I_{c,r}$. A breakpoint dependent 2D sub-band reconstruction procedure BD_2D_SR is then applied to $I_{c,r}$ that depends on the corresponding breakpoint band $LL_{D_c,r}$.

8.2 TriBPT-dependent irreversible transforms

8.2.1 Introduction

For the TriBPT case, the 2D_GP_MAPPING procedure maps the spatial sub-bands $LL_{c,r}, HL_{c,r}, LH_{c,r}, HH_{c,r}$ to corresponding grid-point locations to form an interleaved band $I_{c,r}$ such that $I_{c,r} =$ 2D_GP_MAPPING $(LL_{c,r}, HL_{c,r}, LH_{c,r}, HH_{c,r}, trx_o, trx_l, try_o, try_l)$ where (trx_o, try_o) specifies the upper left corner and $(trx_l - 1, try_l - 1)$ the lower right corner of the rectangular grid-point region at resolution r that corresponds to $I_{c,r}$. The mapping of sub-band samples to grid-point locations for a given resolution r is illustrated in Figure 6; while the specific example shown relates to the TriBPT-LR case, the mapping remains identical for TriBPT-RL.

NOTE – As shown in Figure 6, the grid-point mapping procedure associates the samples of the $HL_{c,r}, LH_{c,r}, HH_{c,r}$ spatial sub-bands at the same locations as horizontal, vertical and diagonal arcs from the breakpoint component D_C, respectively.

In the description that follows, it is convenient to identify different triangular elements within a 4-span (see Figure 3) by means of two variables, s_{LR} and s_{TB}. The first variable distinguishes between the mirror-image geometries TriBPT-LR and TriBPT-RL, with $s_{LR} = 1$ for the TriBPT-LR arrangement and $s_{LR} = -1$ for the TriBPT-RL arrangement. The second variable distinguishes between the upper and lower triangles within the 4-span, denoted TOP_Tri and BOT_Tri, with $s_{TB} = 1$ and $s_{TB} = -1$, respectively.

This clause defines the TriBPT-dependent irreversible transforms. The reversible version of these transforms are defined in 8.3.

The interleaved band $I_{c,r}$ produced by the 2D_GP_MAPPING procedure, is subject to 2D filtering BD_2D_SR, which consists of two lifting steps. The update step is applied first, modifying sub-band samples at grid-points (p_x, p_y) that are (even, even), based on sub-band samples and breakpoint values at connected neighbouring grid-points with (even, odd), (odd, even) and (odd, odd) locations. The update step is implemented by making the following assignment for each (even, even) location $(2n, 2m)$ of $I_{c,r}$.

$$I_{c,r}(2n, 2m) = I_{c,r}(2n, 2m) - \sum_{(\delta_n, \delta_m) \in \mathcal{N}_U} U\left(I_{c,r}, LL_{D_c,r}, 2n + \delta_n, 2m + \delta_m\right) \tag{10}$$

Here, $\boldsymbol{\mathcal{N}_U} = \{(\mathbf{1}, \mathbf{0}), (\mathbf{0}, \mathbf{1}), (\boldsymbol{s_{LR}}, \mathbf{1}), (-\mathbf{1}, \mathbf{0}), (\mathbf{0}, -\mathbf{1}), (-\boldsymbol{s_{LR}}, -\mathbf{1})\}$ identifies the update neighbourhood, and the update function $\boldsymbol{U}$ is given by

$$U(I_{c,r}, LL_{D_c,r}, p_x, p_y) = \begin{cases} \mu * I_{c,r}(p_x, p_y), & \text{if } t_b(p_x, p_y) = 0 \\ 0, & \text{otherwise} \end{cases}, \tag{11}$$

where $\mu = \frac{1}{8}$ and $t_b(p_x, p_y)$ is the arc break-type found from $LL_{D_c,r}(p_x, p_y)$. As defined in clause 7, $t_b = 0$ means that there is no break on the arc. Figure 7 shows the operation of the update step for the TriBPT-LR arrangement, providing examples with no breakpoints and when breakpoints are present on neighbouring arcs.

After the update step, the (even, even) indexed sub-band samples located at end points of arcs at resolution r are utilized in performing the second (predict) lifting step. The predict step is implemented by making the following assignment for each (even, odd), (odd, even) and (odd, odd) location within $I_{c,r}$:

$$I_{c,r}(\boldsymbol{p}) = I_{c,r}(\boldsymbol{p}) + P\left(I_{c,r}, LL_{D_c,r}, \boldsymbol{p}\right), \text{ where } \boldsymbol{p} = (p_x, p_y) \text{ and } p_x, p_y \text{ are not both even.} \quad (12)$$

For a root or non-root arc centred at location $\boldsymbol{p} = (p_x, p_y)$, the predict function $P()$ is dependent on the corresponding break-type $t_b(p_x, p_y)$, as well as the location of the break $k_b(p_x, p_y)$, as given by $LL_{D_c,r}(p_x, p_y)$. There are in total five distinct cases that give rise to specific breakpoint dependent predict functions, as detailed in 8.2.2 through 8.2.5.

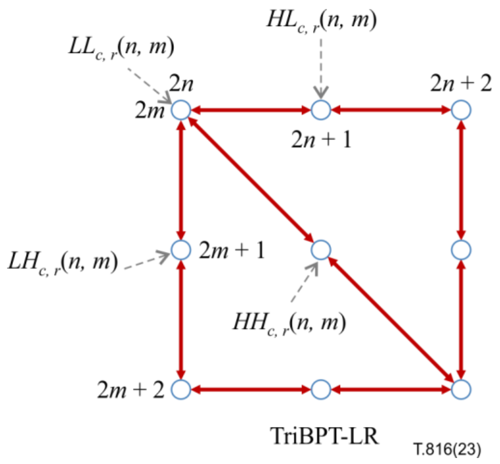
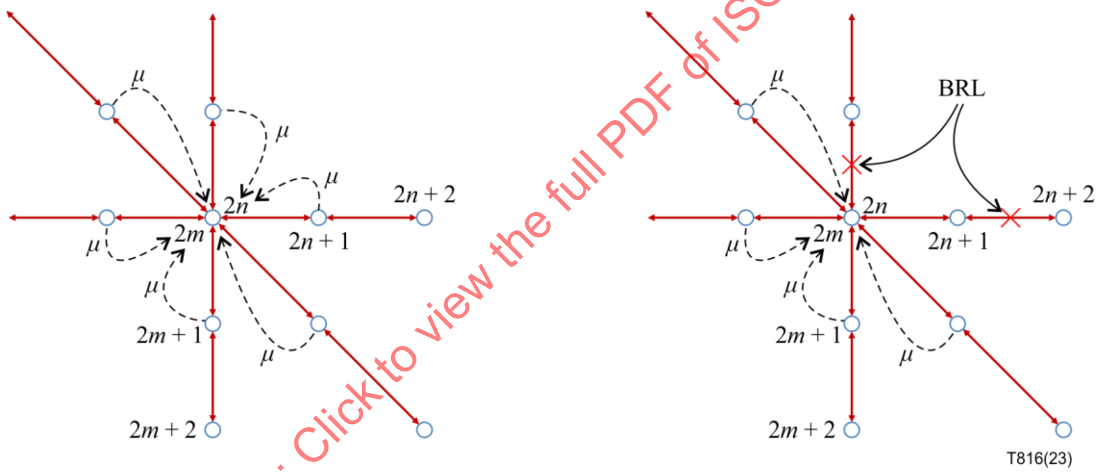


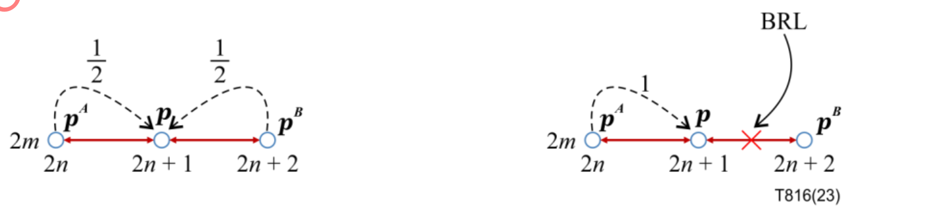
Figure 6 – Interleaving of sub-band coefficients to a triangular grid for a given resolution *r*



a) Update step in the absence of breakpoints   b) Update step in the presence of breakpoints

BRL   breakpoint location

Figure 7 – Update step examples for TriBPT-LR arrangement



a) Predict step in the absence of breakpoints   b) Predict step in the presence of a breakpoint

BRL   breakpoint location

Figure 8 – Predict step examples for a horizontal arc

8.2.2 Arcs with $t_b = 0$

For arcs without any breaks ($t_b(\boldsymbol{p}) = 0$), the predict function is given by

$$\boldsymbol{P}\left(I_{c,r}, LL_{D_c,r}, \boldsymbol{p}\right) = \frac{1}{2}\left(I_{c,r}(\boldsymbol{p}^{\boldsymbol{A}}) + I_{c,r}(\boldsymbol{p}^{\boldsymbol{B}})\right) \tag{13}$$

where the prediction source locations $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ are given by

$$[\boldsymbol{P}^{\boldsymbol{A}}, \boldsymbol{p}^{\boldsymbol{B}}] = \begin{cases} [(2n, 2m), (2n+2, 2m)] & \text{if } \boldsymbol{p} = (2n+1, 2m) \\ [(2n, 2m), (2n, 2m+2)] & \text{if } \boldsymbol{p} = (2n, 2m+1) \\ [(2n, 2m), (2n+2s_{LR}, 2m+2)] & \text{if } \boldsymbol{p} = (2n+s_{LR}, 2m+1) \end{cases} \tag{14}$$

An example involving a horizontal arc is shown in Figure 8 (left).

8.2.3 Arcs with $t_b = 3$

For arcs containing a vertex ($t_b(\boldsymbol{p}) = 3$), the predict function is given by

$$P\left(I_{c,r}, LL_{D_c,r}, \boldsymbol{p}\right) = \begin{cases} I_{c,r}(\boldsymbol{p}^{\boldsymbol{A}}) & k_b(\boldsymbol{p}) > 2^{d+F_B-1} \\ I_{c,r}(\boldsymbol{p}^{\boldsymbol{B}}) & k_b(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases} \tag{15}$$

where the prediction source locations $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ are given by Formula (14). An example involving a horizontal arc is shown in Figure 8 (right).

8.2.4 Arcs with $t_b = 2$

For arcs containing a directly induced break ($t_b(\boldsymbol{p}) = 2$), the predict function is given by

$$P\left(I_{c,r}, LL_{D_c,r}, \boldsymbol{p}\right) = \begin{cases} I_{c,r}(\boldsymbol{p}^{A}) + G_b(\boldsymbol{p}, 2\boldsymbol{p}^A - \boldsymbol{p}) & k_b(\boldsymbol{p}) > 2^{d+F_B-1} \\ I_{c,r}(\boldsymbol{p}^{B}) + G_b(\boldsymbol{p}, 2\boldsymbol{p}^B - \boldsymbol{p}) & k_b(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases} \tag{16}$$

Where the prediction source locations $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ are given by Formula (14) and $G_b(\boldsymbol{p}, \boldsymbol{p}^{\text{nbr}})$ refers to a gradient value

$$G_b\left(\boldsymbol{p}, \boldsymbol{p}^{\text{nbr}}\right) = \begin{cases} \frac{1}{2}\left[I_{c,r}\left(\boldsymbol{p}^{\text{nbr}} + \frac{\boldsymbol{p} - \boldsymbol{p}^{\text{nbr}}}{2}\right) - I_{c,r}\left(\boldsymbol{p}^{\text{nbr}} - \frac{\boldsymbol{p} - \boldsymbol{p}^{\text{nbr}}}{2}\right)\right], & \text{if SMOOTH}(\boldsymbol{p}, \boldsymbol{p}^{\text{nbr}}) \\ 0 & \text{otherwise} \end{cases}$$

Here, $\boldsymbol{p}^{\text{nbr}}$ denotes the location of a neighbouring arc within $LL_{D_c,r}$, to the one at location $\boldsymbol{p}$, and the boolean function SMOOTH$(\boldsymbol{p}, \boldsymbol{p}^{\text{nbr}})$ returns TRUE only if the following three conditions are satisfied:

- $\boldsymbol{p}^{\text{nbr}}$ belongs to $LL_{D_c,r}$ and
- $t_b\left(\boldsymbol{p}^{\text{nbr}}\right) = 0$ and
- GRADX_FLAG $_{induced}(\boldsymbol{p}) = 1$

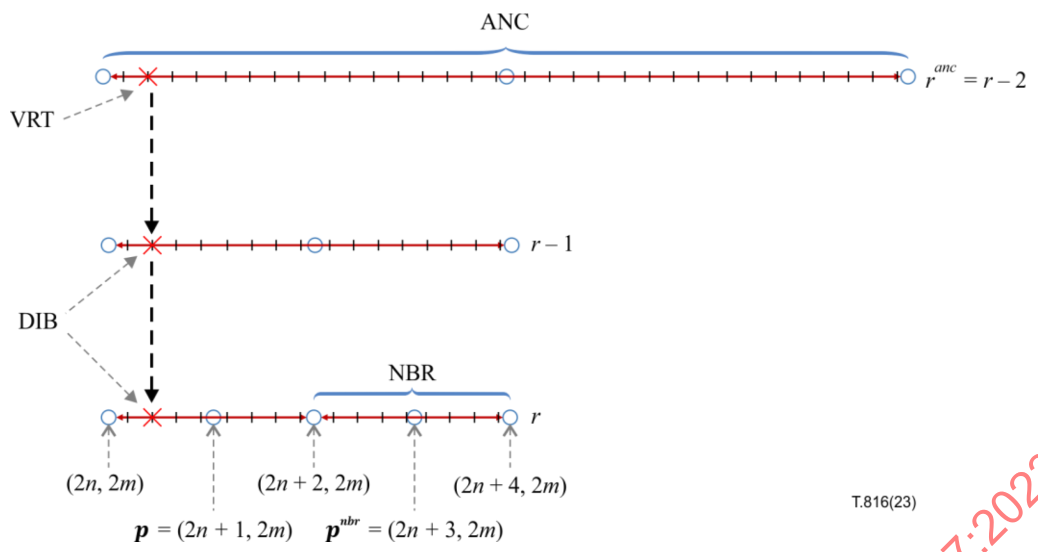
where GRADX_FLAG $_{induced}$ for the arc at $\boldsymbol{p}$ is defined as shown below.

$$\text{GRADX\_FLAG}_{\,induced}(\boldsymbol{p}) = \begin{cases} \text{GRAD1\_FLAG}_{\,induced} & k_b(\boldsymbol{p}) > 2^{d+F_B-1} \\ \text{GRAD2\_FLAG}_{\,induced} & k_b(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases}$$

The values of the binary flags GRAD1_FLAG $_{induced}$ and GRAD2_FLAG $_{induced}$ are defined by the extrapolation-qualifier e_b for the arc as specified in 7.5.

An example of direct induction is shown in Figure 9. The example shows a vertex at resolution r^{anc}, directly inducing a break on the horizontal arc centred at $\boldsymbol{p} = (2n+1,\ 2m)$ in resolution r. The neighbouring arc nbr at location $\boldsymbol{p}^{\mathbf{nbr}} = (2n+3, 2m)$ and ancestor arc (anc) are also shown. The extrapolation-qualifier for the second half of the current arc $GRAD2\_FLAG_{\,induced} = 1$ to indicate that the neighbouring arc nbr is a sub-arc of the ancestor arc anc. Since nbr is free from breaks and a sub-arc of anc, the samples at both endpoints of nbr are used to determine the gradient value $\boldsymbol{G}_{\boldsymbol{b}}(\boldsymbol{p}, \boldsymbol{p}^{\mathbf{nbr}})$. The extrapolation-qualifier for the first half of the current arc $GRAD1\_FLAG_{\,induced} = 0$.

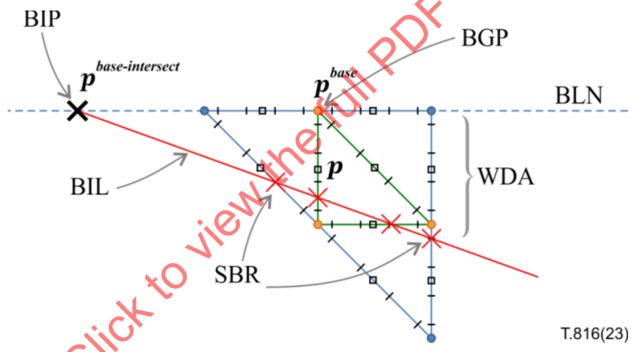
NOTE – As explained in clause 7, $t_b(\boldsymbol{p}) = 2$ means that the arc at $\boldsymbol{p}$ contains a break that is recursively induced from a decoded vertex on some ancestor arc at a coarser resolution $r^{anc} < r$. The predict function allows for linear extrapolation within the span of this ancestor arc.



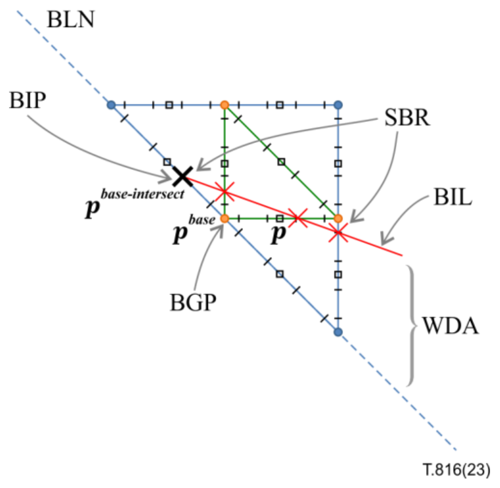
ANC	ancestor arc (anc)	DIB	directly induced breaks
VRT	vertex	NBR	neighbouring arc (nbr)

Figure 9 – Example of direct induction of a breakpoint

Figure 10 illustrates examples of spatially induced breaks on root arcs by source breaks on non-root arcs for the TriBPT-LR arrangement. The base-arc and base-grid-point along with the configuration of the corresponding wedge area for each case is shown.



a) Example illustrating spatially induced break on vertical root arc

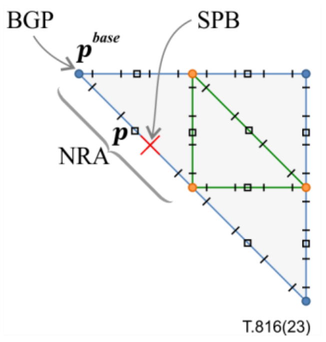


b) Example illustrating spatially induced break on horizontal root arc

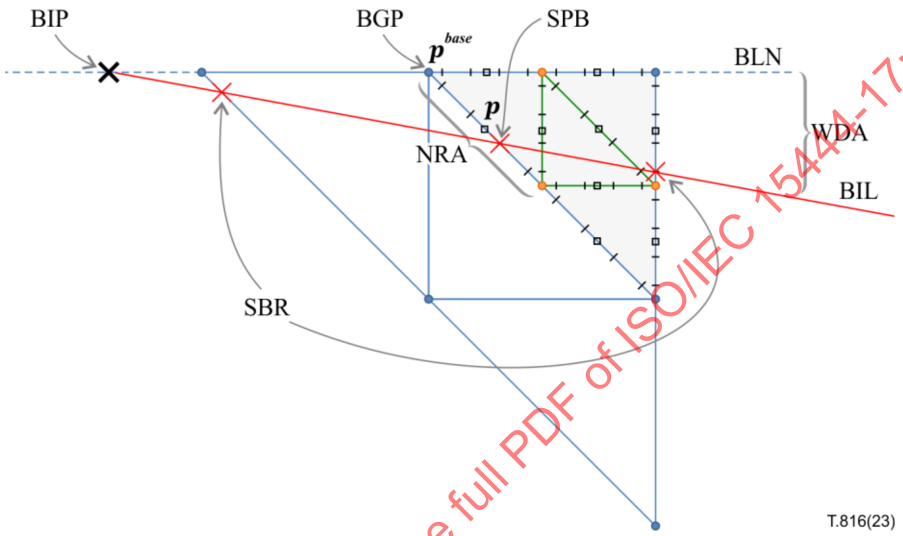
BIP	base-intersection-point	BGP	base-grid-point	BLN	base-line
BIL	break-inducing-line	WDA	wedge area	SBR	source breaks

Figure 10 – Example illustrating spatially induced breaks

Figure 11 gives an example of spatial induction for a non-root arc.



a) Example of a non-root arc with a spatially induced break at resolution r

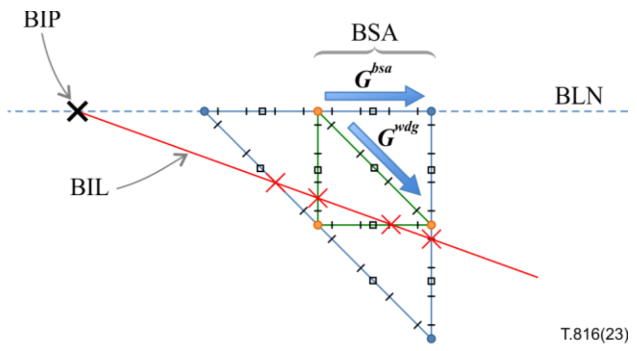


b) Source breaks and inducing line for parent-4-span at resolution $r^{rt} = r - 1$

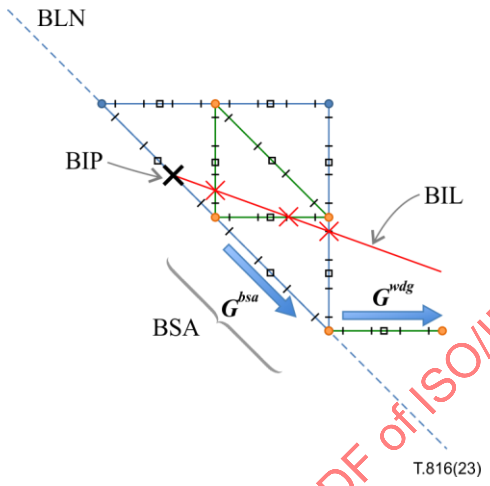
BGP	base-grid-point	SPB	spatially induced break	NRA	non-root arc
BIP	base-intersection-point	BLN	base-line	WDA	wedge area
BIL	break-inducing-line	SBR	source breaks		

Figure 11 – Example of a spatially induced breakpoint on a non-root arc for the TriBPT-LR arrangement

Figure 12 identifies the corresponding base-arc for the spatial induction cases illustrated in Figure 10.



a) Vertical root arc with binary flags ORN = Horizontal and DRCTNX_FLAG $_{induced}$ = 1.

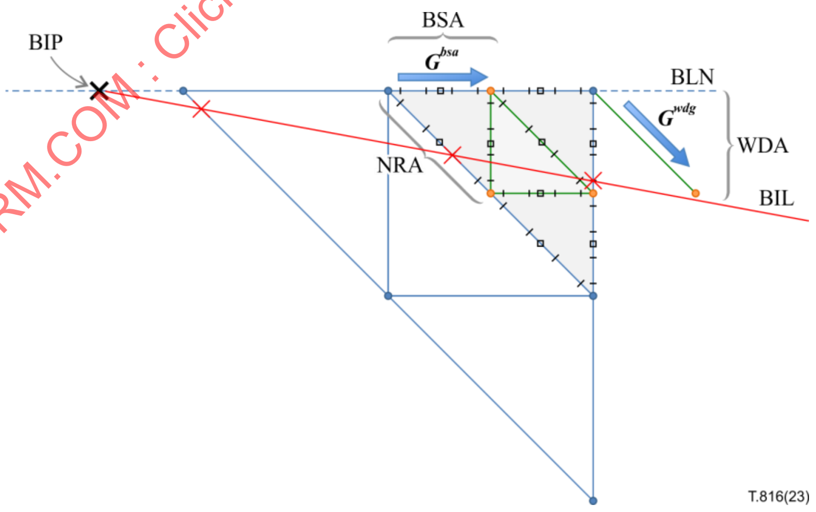


b) Horizontal root arc with binary flags ORN = Diagonal and DRCTNX_FLAG $_{induced}$ = 1.

BIP	base-intersection-point	BSA	base-arc
BLN	base-line	BIL	break-inducing-line

Figure 12 – Examples showing base-arc bsa and gradients G^{bsa} and G^{wdg} along with corresponding binary flags ORN and $DRCTNX\_FLAG$ $_{induced}$ for root arcs

Figure 13 identifies the base-arc for the spatial induction configuration presented in Figure 11.



BIP	base-intersection-point	BSA	base-arc	BLN	base-line
BIL	break-inducing-line	WDA	wedge area	NRA	non-root arc

For the diagonal non-root arc containing a spatially induced break, the corresponding binary flags ORN = Horizontal and DRCTNX_FLAG $_{induced}$ = 1.

Figure 13 – Example showing base-arc bsa and gradients G^{bsa} and G^{wdg} along with corresponding binary flags ORN and $DRCTNX\_FLAG$ $_{induced}$ for diagonal non-root arc containing a spatially induced break

8.2.5 Arcs with $t_b = 1$

8.2.5.1 Introduction

For an arc a centred at location $\boldsymbol{p}$ with $t_b(\boldsymbol{p}) = 1$, two options are pursued to infer a gradient along the arc. These options involve other arcs at the same resolution and within a specified spatial proximity of the arc a.

Let $\boldsymbol{p} = 2\boldsymbol{g} + \boldsymbol{\delta}$ denote the location of the centre of an arc a which contains a spatially induced break, where $\boldsymbol{g} = (g_n, g_m)$ is an integer valued vector and $\boldsymbol{\delta} = (\delta_n, \delta_m) \in \{(1{,}0), (0{,}1), (s_{LR}, 1)\}$. The end point of the arc a that falls on the same side of the break as the centre of the arc is identified as the *base-grid-point* $\boldsymbol{p}^{\text{base}}$ and is given by

$$\boldsymbol{p}^{\text{base}} = \begin{cases} 2\boldsymbol{g} & \text{if } k_b(\boldsymbol{p}) > 2^{d+F_B-1} \\ 2\boldsymbol{g} + 2\boldsymbol{\delta} & \text{if } k_b(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases} \tag{17}$$

At resolution r of the current arc a. the grid line which contains the base-grid-point and having orientation ORN is referred to as the *base-line*. The orientation, either horizontal, vertical or diagonal is determined by the operator ORN_4SPAN as shown below and defined in 8.2.5.5.

$$ORN = ORN\_4SPAN(p, k_b(p))$$

For an arc with a spatially induced break, there are always two breaks defined on other arcs that are the source breaks of the spatial induction. The line formed by these two induction source breaks, is referred to as the *break-inducing-line*. Examples of spatial induction for root arcs is shown in Figure 10 while an example of spatial induction for a non-root arc is illustrated by Figure 11.

For the arc a centred at location $\boldsymbol{p}$ with $t_b(\boldsymbol{p}) = 1$ two options, known as WDG and PRL, are considered to infer a gradient along the arc. Each of these two options has an associated validity flag, and a gradient value that is meaningful if the corresponding validity flag is TRUE. The validity flags and gradient values are listed below.

- Flag VALID_WDG is set to TRUE if the WDG option identifies a valid gradient G_b^{wdg}, derived from arcs located in the wedge region WDG_REGION bounded by the base-line and the break-inducing-line.
- Flag VALID_PRL is set to TRUE if the PRL option identifies a valid gradient G_b^{prl}, derived from an arc that is parallel to the arc a and shares the base-grid-point $\boldsymbol{p}^{\text{base}}$.

The conditions for setting the flags VALID_WDG and VALID_PRL and the process of determining the associated gradient values are detailed below. In these discussions, an operator $GRAD(\boldsymbol{p})$, as defined in Formula (18), is used to determine the gradient of an arc at resolution r, centred at location $\boldsymbol{p} = (2n + \delta_n, 2m + \delta_m)$ with $(\delta_n, \delta_m) \in \{(1{,}0), (0{,}1), (s_{LR}, 1)\}$.

$$GRAD(\boldsymbol{p}) = I_{c,r}(2n + 2\delta_n, 2m + 2\delta_m) - I_{c,r}(2n, 2m) \tag{18}$$

8.2.5.2 VALID_WDG and G_{wdg}

There are two arcs at resolution r that are on the base-line and share the base-grid-point $\boldsymbol{p}^{\text{base}}$. Of these two arcs, the arc with its centre located away from $\boldsymbol{p}^{\text{base}}$ in the direction DRCTNX_FLAG $_{induced}$ is identified as the base-arc bsa with $\boldsymbol{p}^{\text{bsa}}$ corresponding to the centre location of the arc. The direction DRCTNX_FLAG $_{induced}$ of the current arc a centred at location $\boldsymbol{p}$ is determined as shown below.

$$\text{DRCTNX\_FLAG}_{induced}(\boldsymbol{p}) = \begin{cases} \text{DRCTN1\_FLAG}_{induced} & k_b(\boldsymbol{p}) > 2^{d+F_B-1} \\ \text{DRCTN2\_FLAG}_{induced} & k_b(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases}$$

where the values of the binary flags DRCTN1_FLAG $_{induced}$ and DRCTN2_FLAG $_{induced}$ are defined by the extrapolation-qualifier e_b for the arc as specified in 7.5. The location of the base arc $\boldsymbol{p}^{\text{bsa}}$, along the base-line having orientation ORN, is given by

$$\boldsymbol{p}^{\text{bsa}} = \boldsymbol{p}^{\text{base}} + (2 \times \text{DRCTNX\_FLAG}_{induced}(\boldsymbol{p}) - 1)\, \boldsymbol{\delta}_{ORN}$$

$$\boldsymbol{\delta}_{ORN} = \begin{cases} (1{,}0) & \text{if } ORN = Horizontal \\ (0{,}1) & \text{if } ORN = Vertical \\ (s_{LR}, 1) & \text{if } ORN = Diagonal \end{cases}$$

The orientation value ORN of the base-line is determined as explained in 8.2.5.5.

If base-arc bsa contains a break, that is $t_b(\boldsymbol{p}^{\text{bsa}}) > 0$, then the flag VALID_WDG is set to false (VALID_WDG = FALSE). Otherwise, a process is initiated to identify a specific arc wdg located inside the WDG_REGION and to determine its gradient G^{wdg} and validity flag VALID_WDG, as detailed below.

To evaluate G^{wdg}, a set of candidate arcs W from the current resolution r is first determined such that the centre location $\boldsymbol{p}_c^{\mathrm{W}}$ of an arc in this set is given by

$$\boldsymbol{p}_c^{\mathrm{W}} = \boldsymbol{p} + c.\left(2 \times \mathrm{DRCTNX}_{\mathrm{FLAG}_{induced}}(\boldsymbol{p}) - 1\right) \boldsymbol{\delta}_{ORN}$$

where $c \in (1,2,3,\ldots C)$ with increasing values of c corresponding to increasing distance from $\boldsymbol{p}$ and the maximum index value C determined by the maximum allowed distance MAX_WDG.

The candidate arcs in the set W are evaluated in order of increasing distance from $\boldsymbol{p}$ to identify a specific arc wdg as detailed below.

The evaluation process starts with $c = 1$ and proceeds as follows:

- Step 1

  If $c > C$ then the evaluation process is terminated and VALID_WDG = FALSE.

- Step 2

  If the arc centred at $\boldsymbol{p}_c^{\mathrm{W}}$ contains a vertex (i.e., $t_b(\boldsymbol{p}_c^{\mathrm{W}}) = 3$) or directly induced break (i.e., $t_b(\boldsymbol{p}_c^{\mathrm{W}}) = 2$) then the evaluation process is terminated and VALID_WDG = FALSE.

- Step 3

  If the arc centred at $\boldsymbol{p}_c^{\mathrm{W}}$is devoid of any breaks, that is $t_b(\boldsymbol{p}_c^{\mathrm{W}}) = 0$, then wdg is equated to the arc centred at $\boldsymbol{p}_c^{\mathrm{W}}$ (i.e., $\boldsymbol{p}^{\mathrm{wdg}} = \boldsymbol{p}_c^{\mathrm{W}}$). The operator $PARENT\_RES\_4\_SPAN()$ is used to determine the highest resolution $r^{a,wdg}$ at which a 4-span arrangement is inclusive of both arcs wdg and a.

  $$r^{a,wdg} = PARENT\_RES\_4\_SPAN(\boldsymbol{p}^{wdg}, \boldsymbol{p})$$

  If the value for $r^{a,wdg}$ is less than 1 then this implies that a common 4-span arrangement does not exist for arcs wdg and a, hence VALID_WDG = FALSE. However, if $r^{a,wdg}$ is within permissible limits, ranging in value from $r - 1$ to 1, then VALID_WDG = TRUE. Finally, the evaluation process is terminated at this step.

- Step 4

  If the arc centred at $\boldsymbol{p}_c^{\mathrm{W}}$ contains a spatially induced break (i.e., $t_b(\boldsymbol{p}_c^{\mathrm{W}}) = 1$) then $c = c + 1$ and the evaluation process continues by returning to Step 1.

If in the evaluation process above the flag VALID_WDG is set to true (VALID_WDG = TRUE) then the gradient $G^{wdg} = GRAD(\boldsymbol{p}^{wdg})$ across the arc wdg is calculated in accordance with Formula (18). The identified arc wdg shall not be parallel to the current arc a . In such cases the gradient G^{bsa} observed along the base-arc bsa is also calculated using the gradient operator such that $G^{bsa} = GRAD(\boldsymbol{p}^{bsa})$. Examples of base-arc bsa and corresponding gradient G^{bsa}, along with gradient G^{wdg} of the identified arc wdg located inside the wedge area are shown in Figure 12 and Figure 13 for the spatial induction cases illustrated in Figure 10 and Figure 11 respectively. Since wdg and bsa represent two non-parallel arcs, a vector addition of their respective gradients G^{wdg} and G^{bsa} is utilized to provide a candidate gradient G_b^{wdg} along the non-root arc a. The general equation for calculating G_b^{wdg} is shown in Formula (19).

$$G_b^{wdg} = s_b\left(\alpha^{wdg} G^{wdg} + \alpha^{bsa} G^{bsa}\right) \tag{19}$$

If the arc wdg is parallel to arc a then the corresponding weights α^{wdg} and α^{bsa} in Formula (19) are given below.

- $\alpha^{wdg} = 1$ and $\alpha^{bsa} = 0$.

Otherwise, the following rules apply for the corresponding weights α^{η} where $\eta \in \{wdg, bsa\}$

- If current arc a is a diagonal arc then $\alpha^{\eta} = s_{LR}$ if η is a horizontal arc otherwise η is a vertical arc and $\alpha^{\eta} = 1$.
- If current arc a is a vertical arc then $\alpha^{\eta} = 1$ if η is a diagonal arc otherwise η is a horizontal arc and $\alpha^{\eta} = -s_{LR}$.
- If current arc a is a horizontal arc then $\alpha^{\eta} = s_{LR}$ if η is a diagonal arc otherwise η is a vertical arc and $\alpha^{\eta} = -s_{LR}$.

For arc a centred at location $\boldsymbol{p}$, the sign value s_b in Formula (19) is determined based on the location of break on arc a and is given by

$$s_b = \begin{cases} 1 & k_b(\boldsymbol{p}) > 2^{d+F_B-1} \\ -1 & k_b(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases} \tag{20}$$

Figure 14 and Figure 15 show examples of arcs centred at $\boldsymbol{p}$.

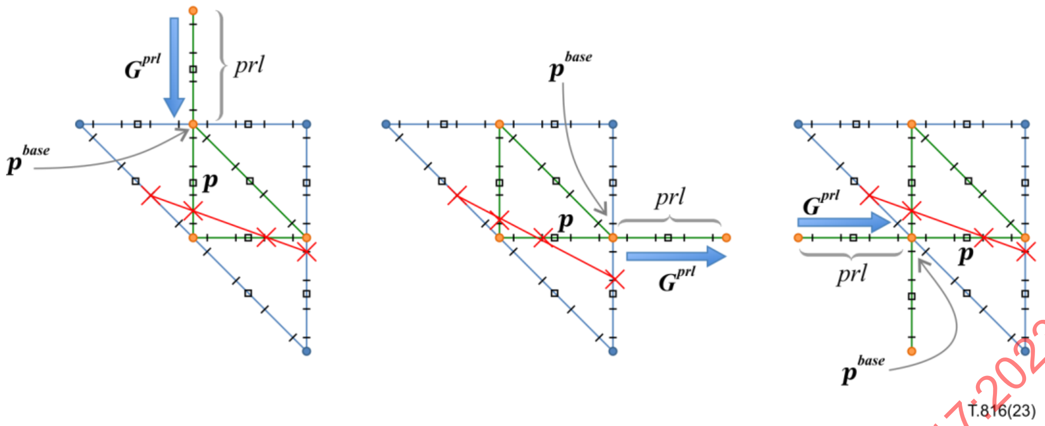
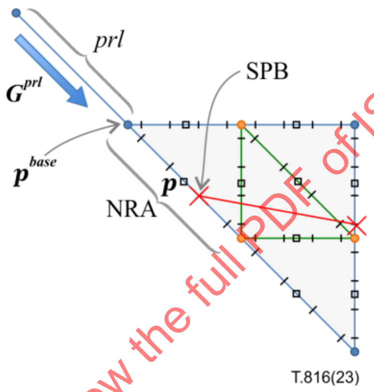


Figure 14 – Examples of arcs prl parallel in orientation to root arcs centred at p and containing a spatially induced break. Gradient calculated across arc prl is denoted as G^{prl}



SPB	spatially induced break
NRA	non-root arc

Figure 15 – Example of arc prl parallel to the non-root arc centred and at p and containing a spatially induced break. Corresponding gradient across arc prl is shown as G^{prl}

8.2.5.3 VALID_PRL and G^{prl}

The arc at resolution r that shares the base-grid-point $\boldsymbol{p}^{\text{base}}$ and has the same orientation as the current arc a is labelled prl. If the arc prl contains a break, that is $t_b(\boldsymbol{p}^{prl}) > 0$, then the flag VALID_PRL is set to false (VALID_PRL = FALSE). Alternatively, if prl is free of any breaks, that is $t_b(\boldsymbol{p}^{prl}) = 0$, then the highest resolution $r^{a,prl}$ at which a 4-span arrangement is inclusive of both arcs prl and arc a is determined using the operator $PARENT\_RES\_4\_SPAN(\,)$ as shown below.

$$\boldsymbol{r}^{a,\text{prl}} = PARENT\_RES\_4\_SPAN(\boldsymbol{p}^{\text{prl}}, \boldsymbol{p}) \tag{21}$$

If the value for $r^{a,prl}$ is less than 1 then this implies that a common 4-span arrangement does not exist for arcs prl and a, hence the flag VALID_PRL is set to false (VALID_PRL = FALSE). However, if $r^{a,prl}$ is within permissible limits, ranging in value from $r - 1$ to 1 then, the flag VALID_PRL is set to true (VALID_PRL = TRUE) and the gradient G^{prl} is calculated, such that $G^{prl} = GRAD(\boldsymbol{p}^{prl})$.

For performing extrapolation along current arc a centred at location $\boldsymbol{p}$, a candidate gradient G_b^{prl} is determined using the gradient G^{prl} of the neighbouring parallel arc prl and the sign value s_b that depends on the location of break on the current arc. The formula for determining G_b^{prl} is shown below where the value for s_b is specified by Formula (20).

$$G_b^{prl} = s_b\, G^{prl} \tag{22}$$

Examples of arcs prl, having parallel orientation with arc a centred at location $\boldsymbol{p}$, are shown in Figure 14 and Figure 15 along with the associated gradient G^{prl}.

8.2.5.4 Predict function

The predict function for an arc a with a spatially induced break (i.e., $t_b(\boldsymbol{p}) = 1$) is given by

$$P\left(I_{c,r}, LL_{D_c,r}, 2n + \delta_n, 2m + \delta_m\right) = \begin{cases} I_{c,r}(2n, 2m) + G_b & k_b(\boldsymbol{p}) > 2^{d+F_B-1} \\ I_{c,r}(2n + 2\delta_n, 2m + 2\delta_m) + G_b & k_b(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases} \quad (23)$$

and the procedure for determining G_b from the evaluated validity flags VALID_WDG and VALID_PRL, and corresponding gradients G_b^{wdg} and G_b^{prl} is specified in Figure 16.

```
IF (VALID_WDG = $FALSE$) AND (VALID_PRL = $FALSE$) THEN
 $G_b = 0$
ELSE IF (VALID_WDG = $TRUE$) AND (VALID_PRL = $FALSE$) THEN
 $G_b = 0.5 * G_b^{wdg}$
ELSE IF (VALID_WDG = $FALSE$) AND (VALID_PRL = $TRUE$) THEN
 $G_b = 0.5 * G_b^{prl}$
ELSE IF (VALID_WDG = $TRUE$) AND (VALID_PRL = $TRUE$) THEN
 IF ($r^{a,wdg} > r^{a,prl}$)
  $G_b = 0.5 * G_b^{wdg}$
 ELSE IF ($r^{a,prl} > r^{a,wdg}$)
  $G_b = 0.5 * G_b^{prl}$
 ELSE
```

Figure 16 – Algorithm for determining the gradient G_b based on validity flags VALID_WDG and VALID_PRL and corresponding attributes

NOTE – If both flags VALID_WDG and VALID_PRL are false, then the predict function essentially reverts to performing constant extrapolation along the arc a. If only one of the validity flags is true, then the corresponding valid gradient is utilized in performing linear extrapolation. If both flags are true, then the resolution at which there is a common shared parent becomes the determining factor. If both $r^{a,wdg}$ and $r^{a,prl}$ are equal, then an average of the gradients G_b^{wdg} and G_b^{prl} is used to perform linear extrapolation, otherwise the gradient corresponding to the higher resolution is chosen for the extrapolation function.

8.2.5.5 Identifying orientation ORN

For an arc a at resolution r, the operator $PARENT\_RES\_ROOT\_ARC(\,)$ is used to identify the resolution level r^{rt} at which a is either a root arc or becomes a sub-arc of a root arc. This is shown below.

$$\boldsymbol{r^{rt}} = PARENT\_RES\_ROOT\_ARC(\boldsymbol{p}) \quad (24)$$

where for arc a with a spatially induced break, r^{rt} can range in value from r to 1. The root arc at resolution level r^{rt} that is inclusive of a is labelled rt.

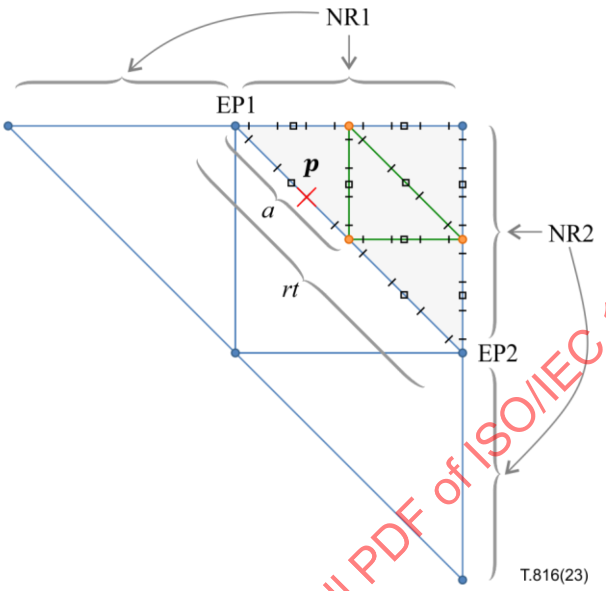
NOTE – If $r^{rt} = r$ then this means that $rt = a$. If $r^{rt} < r$ then a is a sub-arc of rt.

The 4-span arrangement at resolution r^{rt} that is inclusive of the root arc rt is referred to as the *parent-4-span*. An example of a non-root arc a with a spatially induced break for the TriBPT-LR arrangement at resolution r is shown in Figure 11 (top). The corresponding parent-4-span at resolution r^{rt} is shown in Figure 11 (bottom) where in this example $r^{rt} = r - 1$.

The centre coordinates $\boldsymbol{p}^{rt}$ of root arc rt, in terms of the coordinates $p = (n.m)$ of arc a, is given by

$$\boldsymbol{p}^{\mathrm{rt}} = (n^{rt}, m^{rt}) = \left(\left\lfloor \frac{n}{2^{r-r^{rt}+1}} \right\rfloor .2 + (n \bmod 2), \left\lfloor \frac{m}{2^{r-r^{rt}+1}} \right\rfloor .2 + (m \bmod 2) \right)$$

The two endpoints of arc rt are labelled *end-point1* and *end-point2*. The first endpoint, end- point1, is located on the left side of the arc for horizontal arcs or on top for vertical and diagonal arcs and the second endpoint, end- point2, is located on the right side for horizontal arcs or at the bottom for vertical and diagonal arcs. In the parent-4-span arrangement, there are two non-root arcs of the same orientation $ORN1$ that share end-point1. Similarly, there are two non-root arcs of orientation $ORN2$ that share the end-grid-point2. An example is provided in Figure 17 where for a non-root arc a at resolution r and centred at p, the corresponding root-arc rt in the parent-4-span arrangement occurs at resolution $r^{rt} = r - 1$. The two endpoints of rt and the orientations of the non-root arcs at each endpoint are shown.



EP1	end-point1
EP2	end-point2
NR1	non-root arcs at resolution r^{rt} sharing end-point1
NR2	non-root arcs at resolution r^{rt} sharing end-point2
a	non-root arc at resolution r
rt	root arc in the parent-4-span arrangement at resolution r^{rt}

The orientation of the non-root arcs at the endpoints of rt determine the values for ORN1 and ORN2 such that in this example $ORN1$=Horizontal and $ORN2$=Vertical.

Figure 17 – Parent 4-span arrangement at resolution $r^{rt} = r - 1$ for arc a at resolution r and centred at p

For an arc a centred at $\boldsymbol{p}$ containing a break at location $k_b(\boldsymbol{p})$, the operator $ORN\_4SPAN(\boldsymbol{p}, k_b(\boldsymbol{p}))$ identifies the corresponding orientation ORN as shown below.

$$ORN\_4SPAN(\boldsymbol{p}, k_b(\boldsymbol{p})) = \begin{cases} ORN1 & k_b(\boldsymbol{p}) > 2^{d+F_B-1} \\ ORN2 & k_b(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases}$$

8.3 TriBPT-dependent reversible transforms

In this clause, the reversible update and predict functions for TriBPT case is defined, relying upon the notations and definitions presented in 8.2.

The reversible update step is implemented by making the following assignment for each (even, even) location $(2n, 2m)$ of $I_{c,r}$.

$$I_{c,r}(2n, 2m) = I_{c,r}(2n, 2m) - \left\lfloor \frac{\sum_{(\delta_n,\delta_m)\in\mathcal{N}_U} U\left(I_{c,r}, LL_{D_c,r}, 2n+\delta_n, 2m+\delta_m\right) + 4}{8} \right\rfloor \quad (25)$$

where, as defined in 8.2, $\boldsymbol{\mathcal{N}_U}$ identifies the update neighbourhood.

The update function $\boldsymbol{U}$ is given by

$$U(I_{c,r}, LL_{D_c,r}, p_x, p_y) = \begin{cases} \mu * I_{c,r}(p_x, p_y), & \text{if } t_b(p_x, p_y) = 0 \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

where $\mu = 1$.

The predict step is defined in Formula (12) and the associated predict function $P\left(I_{c,r}, LL_{D_c,r}, \boldsymbol{p}\right)$ for each breakpoint type is define below for the reversible transform.

8.3.1 Arcs with $t_b = 0$

For arcs without any breaks ($t_b(\boldsymbol{p}) = 0$), the predict function is given by

$$\boldsymbol{P}\left(I_{c,r}, LL_{D_c,r}, \boldsymbol{p}\right) = \left\lfloor \frac{1}{2}\left(I_{c,r}(\boldsymbol{p}^{\boldsymbol{A}}) + I_{c,r}(\boldsymbol{p}^{\boldsymbol{B}})\right) \right\rfloor \tag{27}$$

where $\boldsymbol{p}^{\boldsymbol{A}}$ and $\boldsymbol{p}^{\boldsymbol{B}}$ are determined in accordance with Formula (14).

8.3.2 Arcs with $t_b = 3$

For arcs containing a vertex ($t_b(\boldsymbol{p}) = 3$), the predict function for the reversible transform is identical to that defined by Function (15).

8.3.3 Arcs with $t_b = 2$

For arcs containing a directly induced break ($t_b(\boldsymbol{p}) = 2$), the predict function is given by

$$P\left(I_{c,r}, LL_{D_c,r}, \boldsymbol{p}\right) = \begin{cases} I_{c,r}(\boldsymbol{p}^A) + \lfloor G_b(\boldsymbol{p}, 2\boldsymbol{p}^A - \boldsymbol{p}) \rfloor & k_b(\boldsymbol{p}) > 2^{d+F_B-1} \\ I_{c,r}(\boldsymbol{p}^B) + \lfloor G_b(\boldsymbol{p}, 2\boldsymbol{p}^B - \boldsymbol{p}) \rfloor & k_b(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases} \tag{28}$$

where the derivation of G_b and corresponding locations $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ are specified in 8.2.2.

8.3.4 Arcs with $t_b = 1$

The predict function for an arc with a spatially induced break (i.e., $t_b(\boldsymbol{p}) = 1$) is given by

$$P\left(I_{c,r}, LL_{D_c,r}, 2n + \delta_n, 2m + \delta_m\right) = \begin{cases} I_{c,r}(2n, 2m) + G_b & k_b(\boldsymbol{p}) > 2^{d+F_B-1} \\ I_{c,r}(2n + 2\delta_n, 2m + 2\delta_m) + G_b & k_b(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases}$$

The procedure for determining G_b for the reversible transform is specified in Figure 18. The required validity flags VALID_WDG and VALID_PRL, and corresponding gradients G_b^{wdg} and G_b^{prl} are defined in 8.2.5.

```
IF (VALID_WDG $= FALSE$) AND (VALID_PRL $= FALSE$) THEN
    $G_b = 0$
ELSE IF (VALID_WDG $= TRUE$) AND (VALID_PRL $= FALSE$) THEN
    $G_b = \left\lfloor \frac{1}{2} * G_b^{wdg} \right\rfloor$
ELSE IF (VALID_WDG $= FALSE$) AND (VALID_PRL $= TRUE$) THEN
    $G_b = \left\lfloor \frac{1}{2} * G_b^{prl} \right\rfloor$
ELSE IF (VALID_WDG $= TRUE$) AND (VALID_PRL $= TRUE$) THEN
    IF ($r^{a,wdg} > r^{a,prl}$)
        $G_b = \left\lfloor \frac{1}{2} * G_b^{wdg} \right\rfloor$
    ELSE IF ($r^{a,prl} > r^{a,wdg}$)
        $G_b = \left\lfloor \frac{1}{2} * G_b^{prl} \right\rfloor$
    ELSE
```

Figure 18 – Algorithm for determining the gradient G_b for the reversible transform, based on validity flags VALID_WDG and VALID_PRL and corresponding attributes

8.4 QuadBPT-dependent transforms

8.4.1 Introduction

For QuadBPT, the 2D_GP_MAPPING procedure maps the spatial sub-bands $LL_{c,r}, HL_{c,r}, LH_{c,r}, HH_{c,r}$ to corresponding grid-point locations to form an interleaved band $I_{c,r}$ such that $I_{c,r} =$ 2D_GP_MAPPING $(LL_{c,r}, HL_{c,r}, LH_{c,r}, HH_{c,r}, trx_o, trx_l, try_o, try_l)$, where (trx_o, try_o) specifies the upper left corner and $(trx_l - 1, try_l - 1)$ the lower right corner of the rectangular grid-point region at resolution r that corresponds to $I_{c,r}$. The mapping of sub-band samples to grid-point locations for a given resolution r is illustrated in Figure 19.

NOTE – As shown in Figure 19, the grid-point mapping procedure associates the samples of the $HL_{c,r}$ and $LH_{c,r}$ spatial sub-bands at the same locations as respective horizontal, vertical non-root arcs from the breakpoint component D_C, while $HH_{c,r}$ is at the same location as where both the horizontal, vertical root arcs are associated in D_C.

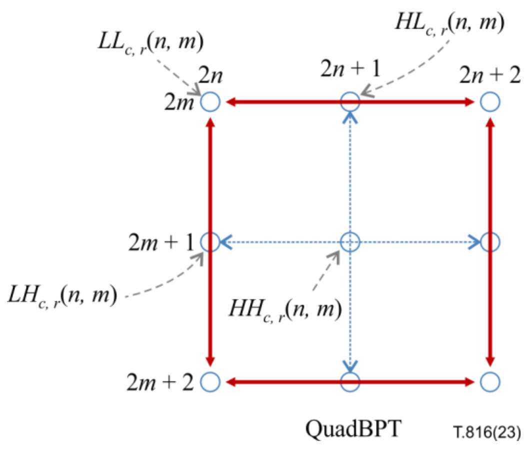


Figure 19 – Interleaving of sub-band coefficients to a quadrilateral grid for a given resolution *r*

The interleaved band $I_{c,r}$ produced by the 2D_GP_MAPPING procedure, is subject to a breakpoint adaptive 2D sub-band reconstruction BD_2D_SR, which consists of four lifting steps conducted in two phases. At the completion of Phase 1, samples at (odd, odd) locations are reconstructed after which performing Phase 2 reconstructs the remaining samples at locations (even, even), (even, odd) and (odd, even).

8.4.2 BD_2D_SR phase 1

In Phase 1, an update step is applied first, modifying sub-band samples at grid-points (p_x, p_y) that are (even, even), (even, odd) and (odd, even). The update step is dependent on sub-band samples at (odd, odd) locations $(2n+1, 2m+1)$ of $I_{c,r}$ and corresponding binary variables $\beta(2n+1, 2m+1)$ determined by

$$\beta(2n+1,2m+1) = \begin{cases} 1 & \text{if } t_b(2n+\delta_n, 2m+\delta_m) = t_b(2n+1,2m+1,H) = t_b(2n+1,2m+1,V) = 0 \\ 0 & \text{otherwise} \end{cases} \quad (29)$$

where $(\delta_n, \delta_m) = \{(1,0),(0,1),(1,2),(2,1)\}$, $t_b(2n+\delta_n, 2m+\delta_m)$ is the arc break-type found from $LL_{D_c,r}(2n+\delta_n, 2m+\delta_m)$ for non-root arcs and $t_b(2n+1,2m+1,H)$ and $t_b(2n+1,2m+1,V)$ refer to the breakpoint type for horizontal and vertical root arcs respectively at $LL_{D_c,r}(2n+1,2m+1)$. As defined in clause 7, $t_b = 0$ means that there is no break on the arc.

NOTE – $\beta(2n+1,2m+1)$ is equal to 1 if all 6 arcs of the 2-span centred at $(2n+1,2m+1)$ are free of any breaks, otherwise it is set to 0.

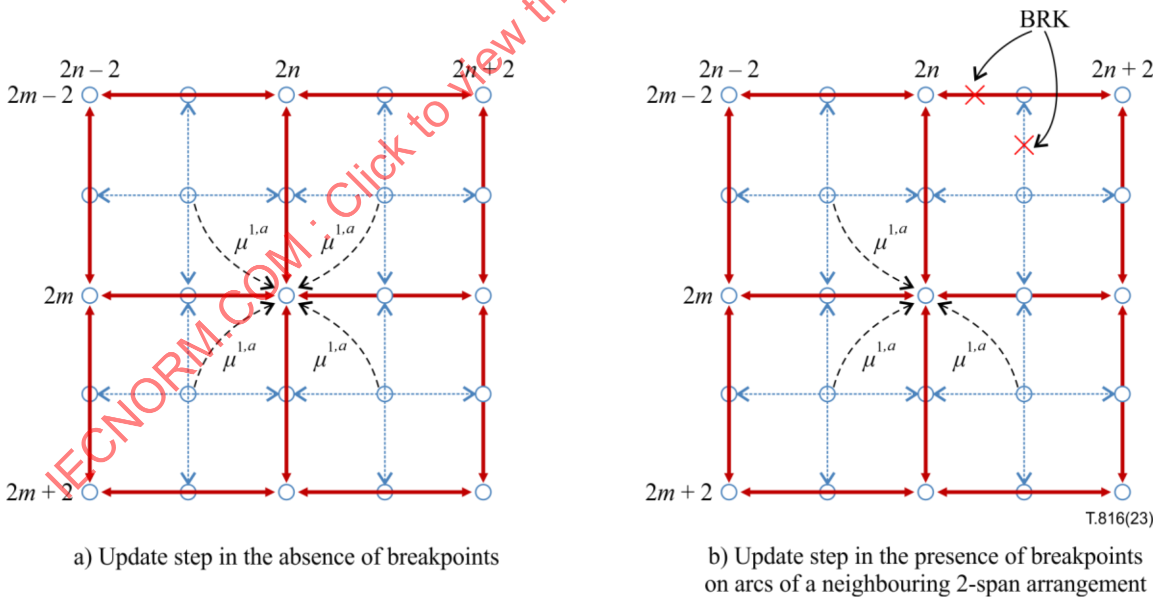
The update step that modifies (even, even) locations during Phase 1 is defined below.

$$I_{c,r}(2n,2m) = I_{c,r}(2n,2m) - \sum_{(\delta_n,\delta_m)\in\mathcal{N}_{1,a}} U^{1,a}\left(I_{c,r}, LL_{D_c,r}, 2n+\delta_n, 2m+\delta_m\right) \quad (30)$$

Here, $\mathcal{N}_{1,a} = \{(1,1),(1,-1),(-1,1),(-1,-1)\}$ identifies the update neighbourhood, and the update function $U^{1,a}$ is given by

$$U^{1,a}(I_{c,r}, LL_{D_c,r}, p_x, p_y) = \begin{cases} \mu^{1,a} * I_{c,r}(p_x,p_y), & \text{if } \beta(p_x,p_y) = 1 \\ 0, & \text{otherwise} \end{cases} \quad (31)$$

where $\mu^{1,a} = 0.04701$. Figure 20 shows the operation of the update step for an (even, even) location, providing examples with no breakpoints and when breakpoints are present on a neighbouring 2-span QuadBPT arrangement.


2n – 2
2n
2n + 2
2m – 2
2m
2m + 2
μ^1,a
BRK
T.816(23)
a) Update step in the absence of breakpoints
b) Update step in the presence of breakpoints on arcs of a neighbouring 2-span arrangement

BRK	breakpoint locations

Figure 20 – Phase 1 update step for (even, even) location $(2n, 2m)$

The update step that modifies (odd, even) location is defined below.

$$I_{c,r}(2n+1,2m) = I_{c,r}(2n+1,2m) - \sum_{(\delta_n,\delta_m)\in\mathcal{N}_{1,b}} U^{1,b}\left(I_{c,r}, LL_{D_c,r}, 2n+\delta_n, 2m+\delta_m\right) \tag{32}$$

where $\mathcal{N}_{1,b} = \{(1,1),(1,-1)\}$ identifies the update neighbourhood and the update function $\boldsymbol{U^{1,b}}$ is given by

$$U^{1,b}(I_{c,r}, LL_{D_c,r}, p_x, p_y) = \begin{cases} \mu^{1,b} * I_{c,r}(p_x,p_y), & \text{if } \beta(p_x,p_y) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{33}$$

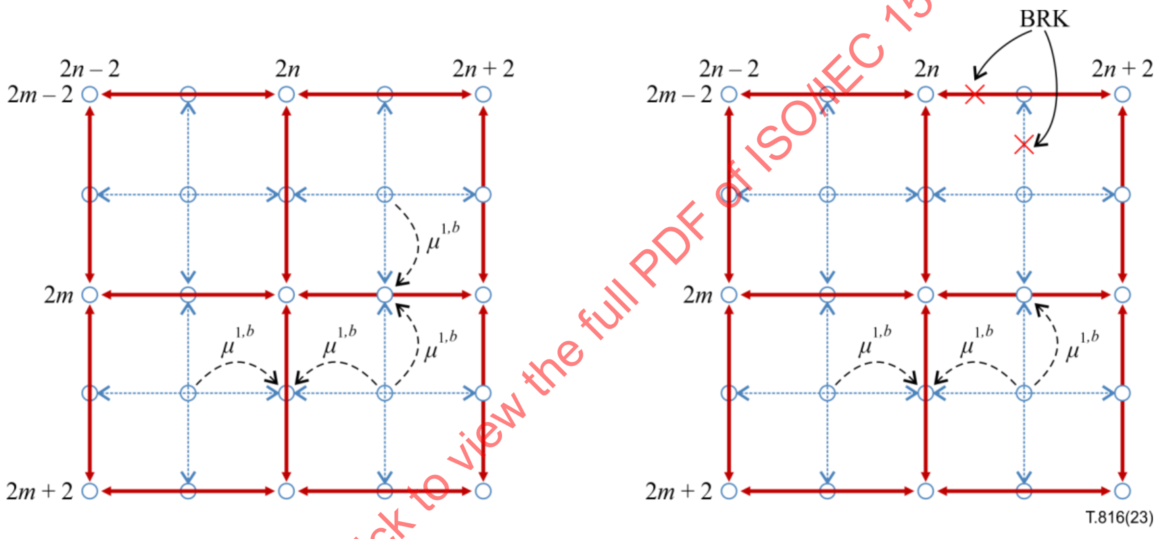
with $\mu^{1,b} = 0.1479$.

The update step that modifies (even, odd) locations is given by

$$I_{c,r}(2n,2m+1) = I_{c,r}(2n,2m+1) - \sum_{(\delta_n,\delta_m)\in\mathcal{N}_{1,c}} U^{1,b}\left(I_{c,r}, LL_{D_c,r}, 2n+\delta_n, 2m+\delta_m\right) \tag{34}$$

where $\mathcal{N}_{1,c} = \{(1,1),(-1,1)\}$ identifies the update neighbourhood and the update function $U^{1,b}$ is defined above in Formula (33).

Examples of the update step for locations $(2n+1, 2m)$ and $(2n, 2m+1)$ are shown in Figure 21.



a) Update step in the absence of breakpoints b) Update step in the presence of breakpoints

BRK breakpoint locations

Figure 21 – Phase 1 update step for (even, odd) location $(2n, 2m+1)$ and (odd, even) location $(2n+1, 2m)$

After the update step, a prediction step is performed that modifies (odd, odd) locations of $I_{c,r}$. To perform this predict step an intermediate interleaved band $I_{c,r}^*$ is first created, to store intermediate values at (odd, even) and (even, odd) locations as shown below.

$$I_{c,r}^*(\boldsymbol{p}) = \begin{cases} I_{c,r}(\boldsymbol{p}) + I_{c,r}(\boldsymbol{p}^A) & \text{if } t_b(\boldsymbol{p}) \neq 0 \text{ and } k_{\boldsymbol{b}}(\boldsymbol{p}) > 2^{d+F_B-1} \\ I_{c,r}(\boldsymbol{p}) + I_{c,r}(\boldsymbol{p}^B), & \text{if } t_b(\boldsymbol{p}) \neq 0 \text{ and } k_{\boldsymbol{b}}(\boldsymbol{p}) < 2^{d+F_B-1} \\ \frac{5}{6} I_{c,r}(\boldsymbol{p}) + \frac{I_{c,r}(\boldsymbol{p}^A) + I_{c,r}(\boldsymbol{p}^B)}{2}, & \text{if } t_b(\boldsymbol{p}) = 0 \end{cases} \tag{35}$$

where $\boldsymbol{p} \in \{(2m+1, 2n), (2m, 2n+1)\}$ and locations $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ are given by

$$[\boldsymbol{P}^A, \boldsymbol{p}^B] = \begin{cases} [(2n,2m),(2n+2,2m)] & \text{if } \boldsymbol{p} = (2n+1,2m) \\ [(2n,2m),(2n,2m+2)] & \text{if } \boldsymbol{p} = (2n,2m+1) \end{cases} \tag{36}$$

The predict step, which reconstructs the sample at (odd, odd) locations by utilising the intermediate interleaved band $I^*_{c,r}$ values at neighbouring (odd, even) and (even, odd) locations is shown below.

$$\begin{aligned} I_{c,r}(2n+1,2m+1) &\\ &= I_{c,r}(2n+1,2m+1) + \frac{1}{2}P_H\left(I^*_{c,r}, LL_{D_c,r}, 2n+1, 2m+1\right) \\ &+ \frac{1}{2}P_V\left(I^*_{c,r}, LL_{D_c,r}, 2n+1, 2m+1\right) \end{aligned} \tag{37}$$

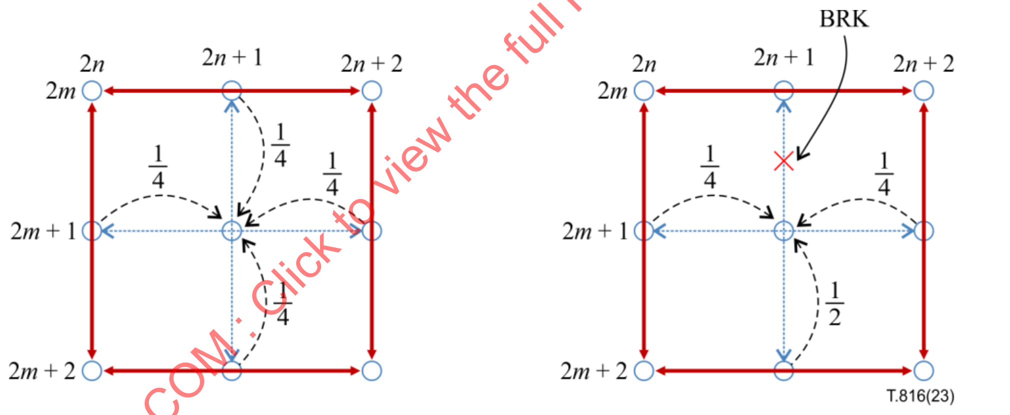
where the predict functions P_H and P_V utilize intermediate values located at endpoints of horizontal and vertical root arcs respectively. For horizontal and vertical root arcs centred at $\boldsymbol{p} = (2n+1, 2m+1)$, the predict functions P_H and P_V are defined below.

$$P_H\left(I^*_{c,r}, LL_{D_c,r}, \boldsymbol{p}\right) = \begin{cases} \frac{1}{2}\left(I^*_{c,r}(\boldsymbol{p^A}) + I^*_{c,r}(\boldsymbol{p^B})\right) & t_b(\boldsymbol{p}, H) = 0 \\ I^*_{c,r}(\boldsymbol{p^A}) & \text{if } t_b(\boldsymbol{p}, H) \neq 0 \text{ and } k_{\boldsymbol{b}}(\boldsymbol{p}) > 2^{d+F_B-1} \\ I^*_{c,r}(\boldsymbol{p^B}) & \text{if } t_b(\boldsymbol{p}, H) \neq 0 \text{ and } k_{\boldsymbol{b}}(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases} \tag{38}$$

Here source locations $\boldsymbol{p^A} = (2n, 2m+1)$ and $\boldsymbol{p^B} = (2n+2, 2m+1)$ in the predict function P_H above.

$$P_V\left(I^*_{c,r}, LL_{D_c,r}, \boldsymbol{p}\right) = \begin{cases} \frac{1}{2}\left(I^*_{c,r}(\boldsymbol{p^A}) + I^*_{c,r}(\boldsymbol{p^B})\right) & t_b(\boldsymbol{p}, V) = 0 \\ I^*_{c,r}(\boldsymbol{p^A}) & \text{if } t_b(\boldsymbol{p}, V) \neq 0 \text{ and } k_{\boldsymbol{b}}(\boldsymbol{p}) > 2^{d+F_B-1} \\ I^*_{c,r}(\boldsymbol{p^B}) & \text{if } t_b(\boldsymbol{p}, V) \neq 0 \text{ and } k_{\boldsymbol{b}}(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases} \tag{39}$$

Source locations $\boldsymbol{p^A} = (2n+1, 2m)$ and $\boldsymbol{p^B} = (2n+1, 2m+2)$ in the predict function P_V. Examples of the source locations of $I^*_{c,r}$ used in the predict function are shown in Figure 22.



a) Predict step in the absence of breakpoints   b) Predict step in the presence of breakpoints

BRK   breakpoint locations

Figure 22 – Phase 1 predict step for (odd, odd) location $(2n+1, 2m+1)$ with source locations of $I^*_{c,r}$ shown

8.4.3 BD_2D_SR phase 2

In Phase 2, an update step is applied first, modifying sub-band samples at grid-points (p_x, p_y) that are (even, even). The update step is dependent on sub-band samples at (even, odd) locations $(2n, 2m+1)$ and (odd, even) locations $(2n+1, 2m)$ of $I_{c,r}$. The update step is implemented by making the following assignment for each (even, even) location $(2n, 2m)$ of $I_{c,r}$.

$$I_{c,r}(2n, 2m) = I_{c,r}(2n, 2m) - \sum\nolimits_{(\delta_n, \delta_m) \in \mathcal{N}_2} U^{(2)}\left(I_{c,r}, LL_{D_c,r}, 2n + \delta_n, 2m + \delta_m\right) \tag{40}$$

Here, $\mathcal{N}_2 = \{(1,0),(0,1),(-1,0),(0,-1)\}$ identifies the update neighbourhood, and the update function $U^{(2)}$ is given by

$$U^{(2)}(I_{c,r}, LL_{D_{c,r}}, p_x, p_y) = \begin{cases} \mu^{(2)} * I_{c,r}(p_x, p_y), & \text{if } t_b(p_x, p_y) = 0 \\ 0, & \text{otherwise} \end{cases}, \tag{41}$$

where $\mu^{(2)} = \frac{1}{6}$ and $t_b(p_x, p_y)$ is the arc break-type found from $LL_{D_{c,r}}(p_x, p_y)$. Figure 23 shows the operation of the update step for an (even, even) location, providing examples with no breakpoints and when breakpoints are present on a neighbouring arc.

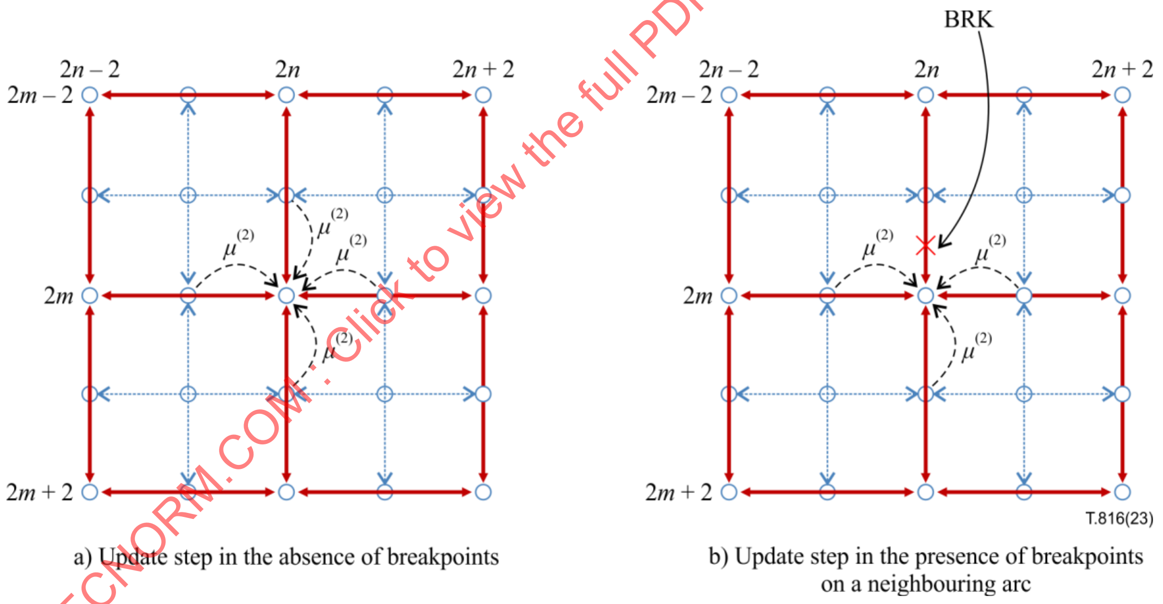
After the update step of Phase 2, the (even, even) indexed sub-band samples located at end points of arcs at resolution r are utilized in performing a predict lifting step by making the following assignment for each (even, odd) and (odd, even) location $\boldsymbol{p}$ within $I_{c,r}$:

$$I_{c,r}(\boldsymbol{p}) = I_{c,r}(\boldsymbol{p}) + P^{(2)}(I_{c,r}, LL_{D_{c,r}}, \boldsymbol{p}), \tag{42}$$

For a non-root arc centred at location $\boldsymbol{p} = (p_x, p_y)$, the predict function $P^{(2)}()$ is dependent on the corresponding break-type $t_b(p_x, p_y)$, as well as the location of the break $k_b(p_x, p_y)$, as given by $LL_{D_{c,r}}(p_x, p_y)$. The predict function is given by

$$P^{(2)}(I_{c,r}, LL_{D_{c,r}}, \boldsymbol{p}) = \begin{cases} \frac{1}{2}(I_{c,r}(\boldsymbol{p}^A) + I_{c,r}(\boldsymbol{p}^B)) & t_b(\boldsymbol{p}) = 0 \\ I_{c,r}(\boldsymbol{p}^A) & \text{if } t_b(\boldsymbol{p}) \neq 0 \text{ and } k_{\boldsymbol{b}}(\boldsymbol{p}) > 2^{d+F_B-1} \\ I_{c,r}(\boldsymbol{p}^B) & \text{if } t_b(\boldsymbol{p}) \neq 0 \text{ and } k_{\boldsymbol{b}}(\boldsymbol{p}) < 2^{d+F_B-1} \end{cases} \tag{43}$$

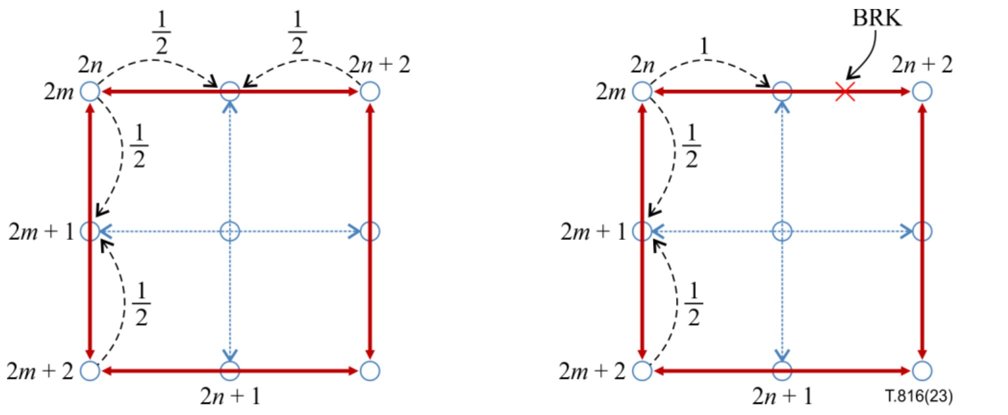
where source locations $\boldsymbol{p}^A$ and $\boldsymbol{p}^B$ are given by Formula (36). Examples of the Phase 2 predict function are shown in Figure 24.



2n – 2
2n
2n + 2
2m – 2
2m
2m + 2
μ(2)
a) Update step in the absence of breakpoints
BRK
b) Update step in the presence of breakpoints on a neighbouring arc
T.816(23)

BRK breakpoint locations

Figure 23 – Phase 2 update step for (even, even) location $(2n, 2m)$



a) Predict step in the absence of breakpoints b) Predict step in the presence of breakpoints

BRK breakpoint locations

Figure 24 – Phase 2 predict step examples

9 Decoding of breakpoint code-blocks

9.1 Embedded bit-plane decoding of breakpoints

Breakpoint code-blocks are decoded progressively, following a sequence of coding passes, each associated with a particular bit-plane in the binary representation of possible vertices V_b. Similar to the block decoding algorithm described in Rec. ITU-T T.800 | ISO/IEC 15444-1, there are three coding passes per bit-plane and each arc within a code-block has an associated binary state variable σ_b, called its significance state. Significance states are initialized to 0 (insignificant) and may become 1 (significant) during the course of the decoding of the code-block. A significant arc is one that contains a vertex. The coding passes employed in breakpoint decoding are classified as follows:

- Pass-0: Position refinement coding pass
- Pass-1: Non-root significance coding pass
- Pass-2: Root significance coding pass

The maximum number of bit-planes for a code-block is the value M_b, as specified in Rec. ITU-T T.800 | ISO/IEC 15444-1. Decoding proceeds from the most significant bit-plane $p = p_{\max}$, down to the least significant bit-plane $p = 0$, where

$$p_{max} = M_b - 1 - P$$

and P is the number of missing most significant bit-planes (or zero bit-planes), as specified in Rec. ITU-T T.800 | ISO/IEC 15444-1. Decoding starts from Pass-1 in bit-plane $p_{\max}$, followed by Pass-2, proceeding to Pass-0 in bit-plane $p_{\max} - 1$, and so forth, so that the maximum number of coding passes for the code-block is $3p_{max} + 2 = 3(M_b - P) - 1$.

NOTE 1 – This maximum number of coding passes is one more than the corresponding value for non-breakpoint code-blocks, as described in Rec. ITU-T T.800 | ISO/IEC 15444-1.

Each decoding pass visits a defined subset of the arcs within the code-block, decoding new information for that arc for the relevant bit-plane p. Pass-1 (non-root significance coding) visits only the non-root arcs that are not yet significant. Pass-2 (root significance coding) visits only root arcs that are not yet significant. Pass-0 (position refinement) visits all arcs in the code-block that have been found significant in a previous (more significant) bit-plane, except those that have been found to hold induction blocks or whose vertex position cannot be refined any further, as explained below.

For code-blocks in the LL sub-band of a breakpoint tile-component, all arcs are considered to be root arcs, so that Pass-1 has no members at resolution $r = 0$.

All coding passes employ the MQ binary arithmetic decoding procedures defined in Rec. ITU-T T.800 | ISO/IEC 15444-1, together with a set of contexts that are specific to the coding pass and breakpoint configuration type (QuadBPT or TriBPT, as appropriate).

In significance coding passes, a binary significance symbol is decoded for each member arc b, which becomes the arc's new significance state σ_b. If this leaves $\sigma_b = 1$, additional symbols are decoded, as required, to determine the break-type t_b and the most significant bit m_b of any vertex V_b for the arc. Specific details of these decoding steps are provided in 9.4 and 9.5. A newly significant vertex is assigned type $t_b = 3$, precision $P_b = 2$ and vertex value $V_b = m_b$, while a newly discovered induction block is assigned $t_b = 0$ and $P_b = 1$, as explained in 7.5.

A vertex that first becomes significant in bit-plane p is also assigned a priority indicator q_b that is equal to $M_b - 1 - \mathrm{p}$. The value of q_b is thus 0 if the vertex becomes significant in the highest (earliest) possible bit-plane $M_b - 1$. In general, the larger the value of q_b, the less significant the vertex was considered to be during encoding. In the intra-band decoding mode, the value of q_b is purely informative, but in the inter-band decoding mode, q_b is used to form decoding contexts for code-blocks in higher resolutions, as explained in 9.2.

The refinement coding passes are governed by a *termination counter* τ_b that identifies the maximum number of refinement symbols that can be decoded for arc b. Induction blocks are immediately assigned $\tau_b = 0$ (refinement terminated immediately), because they have no position information that can be refined. For each arc b with $\sigma_b = 1$ and $\tau_b > 0$, an "early termination" symbol is decoded to determine whether or not τ_b should become 0 (refinement terminated); in this case, no refinement symbol r_b is decoded. If not terminated, a refinement symbol r_b is decoded, the termination counter τ_b is decremented, and the break-precision and vertex value for the arc are updated as follows (a refinement step):

$$P_b \coloneqq P_b + 1 \text{ and } V_b \coloneqq 2V_b + r_b$$

NOTE 2 – The number of bits in a vertex value V_b is always P_b-1. The break-precision P_b can be understood as the precision of the non-zero value $2V_b + 1$, from which the tick-point position is recovered according to Formula (8).

Each breakpoint tile-component has an associated integer parameter $F_{\text{vtx}} \geq 1$ that determines the maximum precision of a decoded vertex. Specifically, each P_b shall satisfy

$$P_b \leq d + F_{\text{vtx}},$$

where $d = N_L + 1 - r$. This means that arc b can be subjected to at most $d + F_{\text{vtx}} - 2$ refinement steps, and so the termination counter τ_b for each arc b in a code-block is initialized prior to any decoding steps according to:

$$\tau_b = d + F_{\text{vtx}} - 2$$

Note that when d and F_{vtx} take their minimum values of 1, all arcs are initialized with $\tau_b = 0$ and so no significant arc can be refined at all – the refinement pass will be empty in this case.

F_{vtx} and F_b play closely related roles, with F_{vtx} constraining the precision of coded vertices and F_b constraining the precision of break locations on an arc. These quantities shall satisfy

$$1 \leq F_{\text{vtx}} \leq F_b$$

NOTE 3 – The precision and value of a decoded vertex do not inherently depend upon the bit-plane in which the arc is first found to be significant, and the same is true for induction blocks. However, if only some of the coding passes are decoded, the break-precision and even the significance of arcs that first become significant in later bit-planes are most strongly affected. Thus, encoders can be expected to arrange for arcs whose breakpoint information is more important to become significant in earlier bit-planes. This is discussed further in B.3, which provides an informative description of a breakpoint media format that can preserve the importance of breakpoint information on an arc b through a quality value Q_b. Encoders can use Q_b to determine the bit-plane in which an arc should first become significant, while decoders can recover the value of Q_b by taking note of the bit-plane in which the arc first becomes significant.

After all decoding passes have been performed, the vertex value V_b of a significant arc is converted to a tick-point location k_b using Formula (8).

9.2 Inter-band coding mode (BPT_INTER)

This Recommendation | International Standard describes two different modes for breakpoint coding, known as "intra-band coding" and "inter-band coding." In the intra-band coding mode, corresponding to BPT_INTER = 0, each decoding pass depends only upon state information produced by previous decoding steps within the same code-block. In the inter-band coding mode, corresponding to BPT_INTER = 1, significance information, denoted $\sigma'_b(\mathrm{p})$, is imported for each non-root arc in the code-block from the breakpoint component's next lower resolution, if there is one. The imported information $\sigma'_b(p) = 1$ if the parent arc b' in the lower resolution has $t_{b'} = 3$ and $q_{b'} \leq M_b - 1 - p$. Otherwise, $\sigma'_b(p) = 0$. That is, $\sigma'_b(p) = 1$ if the parent arc b' was coded as a significant vertex (not an induction block) at the same point or an earlier point than the current bit-plane p, where "earlier" is assessed in terms of the priority value $q_{b'}$, which indicates how far beyond the first possible bit-plane for a code-block the vertex was first coded as significant.

NOTE 1 – The value of $\sigma'_b(p)$ depends on the bit-plane p for which significance decoding is being performed within a code-block. As p decreases between successive significance decoding passes for an arc, the value of $\sigma'_b(p)$ can increase from 0 to 1, but not decrease.

If there is no lower resolution, or BPT_INTER = 0, $\sigma'_b(p)$ is taken to be 0 for all p. Otherwise, significance decoding within bit-plane p is only possible if the code-block or code-blocks that contain all of the block's parent arcs in the lower resolution have either been completely decoded or have been decoded at least to the bit-plane $p + M_{b'} - M_b$.

That is, for each parent arc b', $\sigma'_b(p)$ can be evaluated in the block decoder only if b' has been subjected to significance decoding in bit-plane $p + M_{b'} - M_b$, where $M_{b'}$ is the maximum number of bit-planes for the parent resolution, as specified in Rec. ITU-T T.800 | ISO/IEC 15444-1, except in the case where all originally encoded bit-planes for the parent arc are known to have been decoded. A decoder can deduce whether or not all originally encoded bit-planes have been decoded for a code-block from the zero-completeness and pass-completeness information recovered during packet header decoding, as described in 9.6.

NOTE 2 – The second condition above, related to the decoding of all originally encoded bit-planes, arises because an encoder is not obliged to produce all possible $M_{b'}$ bit-planes for the code-block containing parent arc b'. Encoders can terminate their encoding process early, in which case a decoder that has decoded all encoded content can be sure that it can reconstruct all inter-band conditioning information $\sigma'_b(p)$ that was used by the encoder.

9.3 Cell-based scanning patterns and CBAP-based block flipping

As explained in 7.3, code-blocks of breakpoint components are associated with arrays of 2×2 cells, consisting of H_{blk} cell rows, with W_{blk} cells on each row. For all LL band coding passes, and for the position refinement passes of CL band code-blocks cells are scanned in raster order.

For CL band code-blocks, however, the non-root and root significance coding passes employ a different scanning pattern that involves scanning through 2×2 groups of 2×2 cells. For QuadBPT components, the CL band significance coding passes scan groups in raster order, scanning the cells within each group also in raster order, as illustrated in Figure 25. Only those cells of a group that are actually contained within the code-block are visited in this scanning pattern.

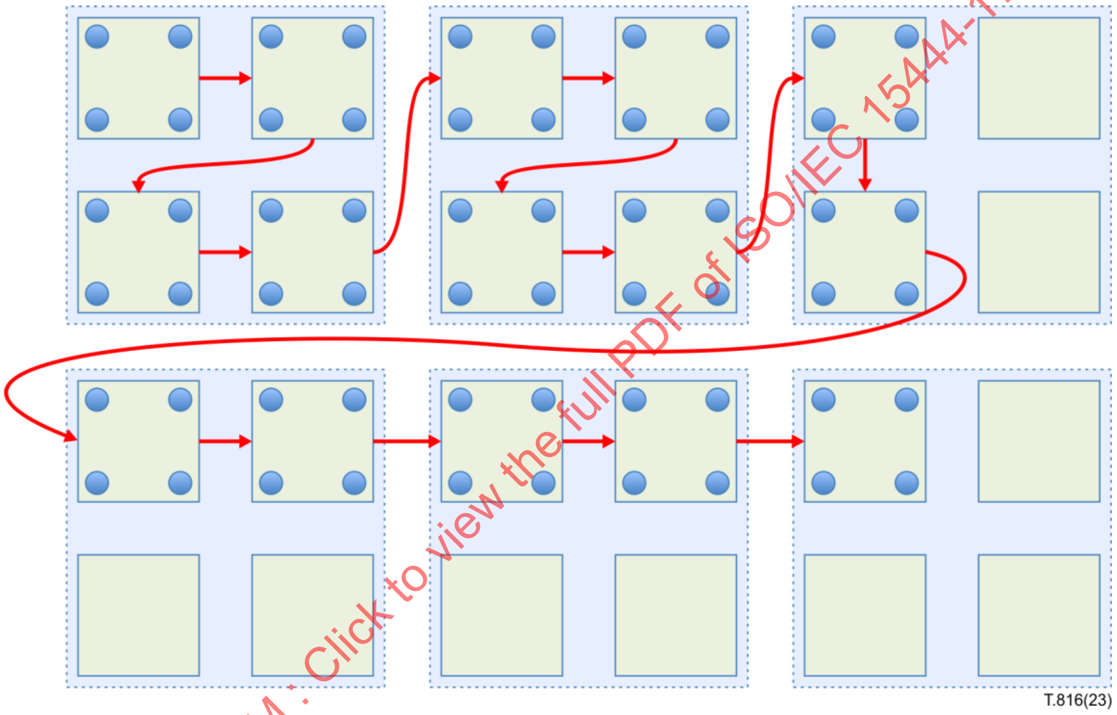


Figure 25 – Canonical scanning pattern used for the significance coding passes of CL band code-blocks in a QuadBPT component. In this case, missing cells within a 2 × 2 group are not included in the scan

For TriBPT components, each group within a CL band code-block represents a single 4-span; the groups are visited in raster order, and decoding procedures are described directly in terms of the arcs within each 4-span, rather than on a cell by cell basis. Each group in this case is considered to have 12 TriBPT arcs, regardless of whether or not all cells of the group are contained within the code-block. The scanning patterns for TriLR and TriRL breakpoint configurations are shown in Figure 26 and Figure 27, respectively, providing labels for each of the 12 arcs of each 4-span group. Notice that the TriLR and TriRL cases are substantially similar, except that the locations of root (DR1 and DR2) and non-root (NR1 and NR2) diagonal arcs change.

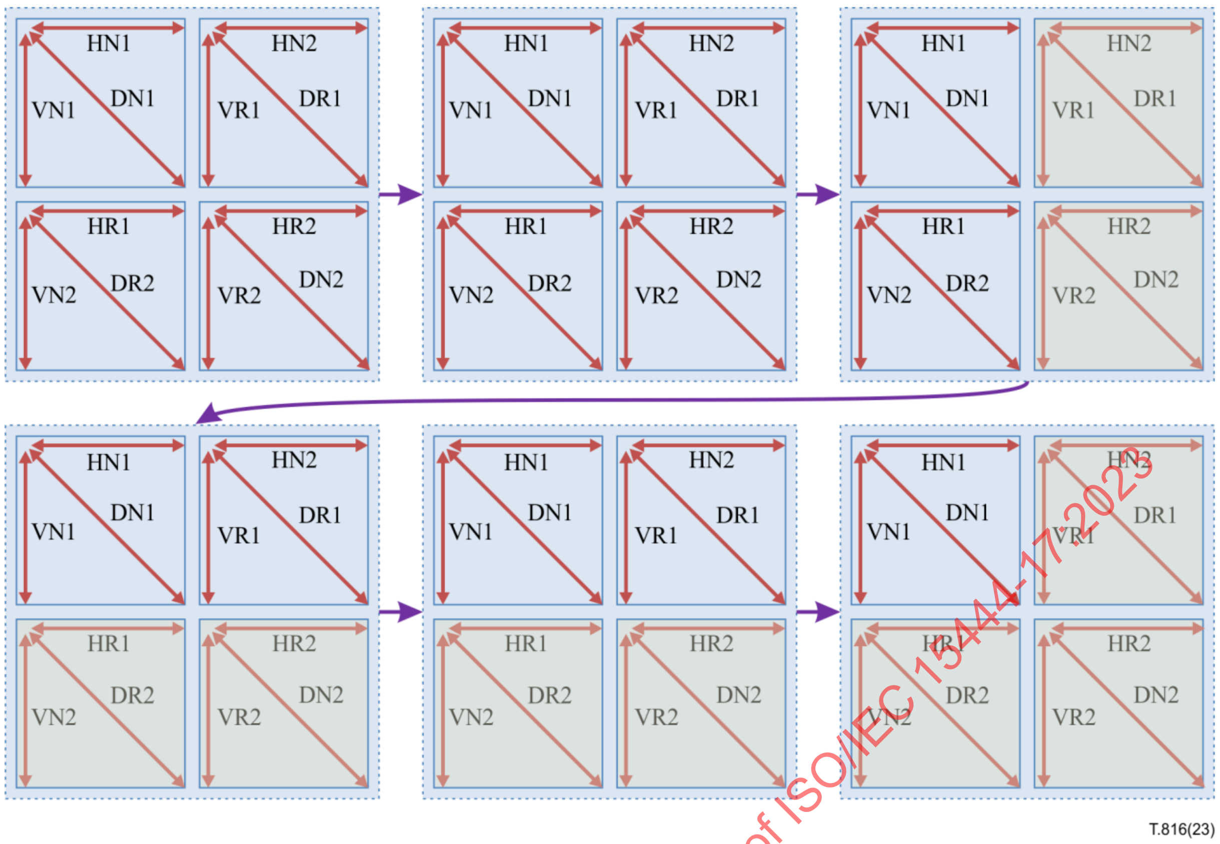


Figure 26 – Canonical scanning pattern used for the significance coding passes of CL band code-blocks in TriLR components. Arcs belonging to cells that lie beyond the code-block boundaries are included in the significance coding passes

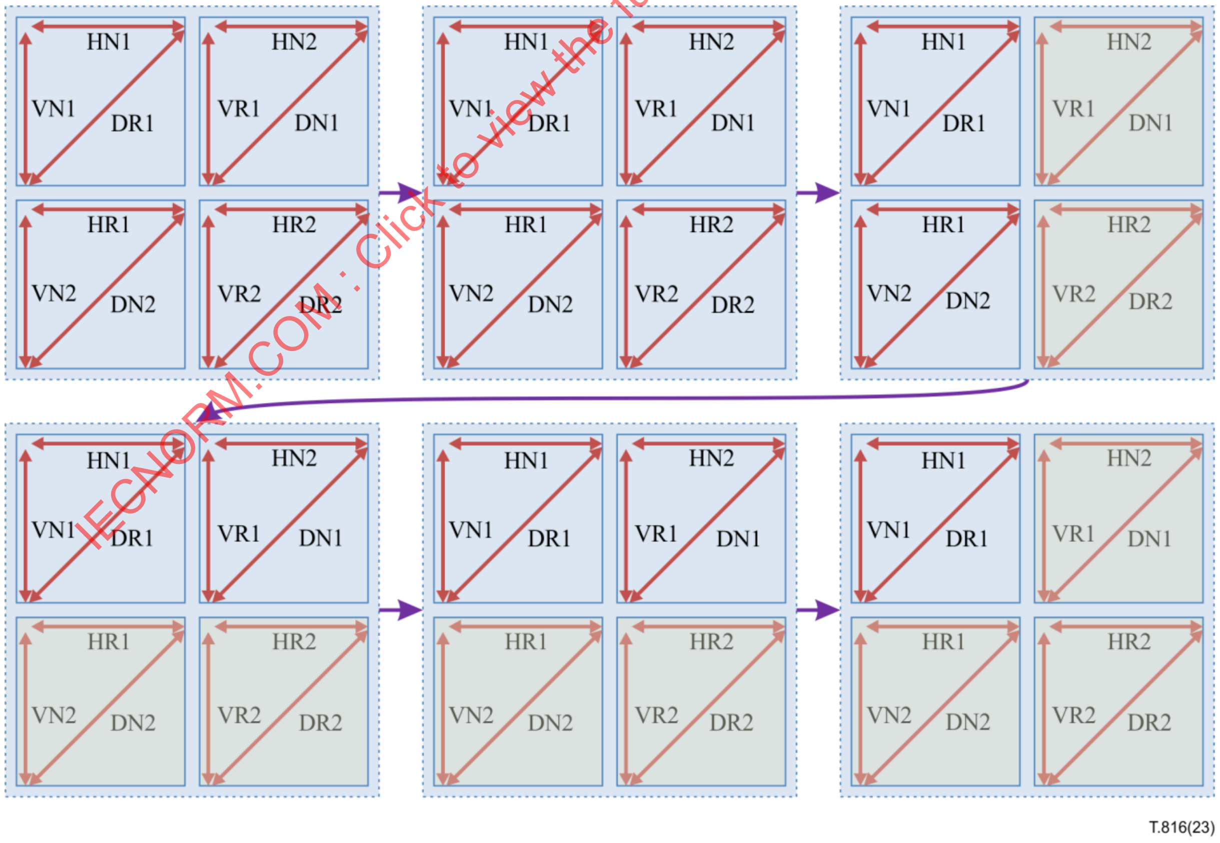


Figure 27 – Canonical scanning pattern used for the significance coding passes of CL band code-blocks in TriRL components. Arcs belonging to cells that lie beyond the code-block boundaries are included in the significance coding passes

These various scanning patterns all have geometric variants that correspond to different values of the CBAP coordinates (z_x, z_y). If the CBAP coordinate $z_x = 1$, groups and cells are scanned from right to left instead of left to right, and groups are aligned with the right edge of the code-block, so that there might be missing cells on the left but not the right edge of a block for which W_{blk} is odd. Similarly, if $z_y = 1$, groups and cells are scanned from bottom to top, rather than from top to bottom, and groups are aligned with the lower edge of the code-block.

Equivalently, all block decoding operations can be performed by following the *canonical scan* associated with $z_x = z_y = 0$, after which the decoded breakpoint data for the code-block is flipped horizontally if $z_x = 1$ and flipped vertically if $z_y = 1$. In this clause, all block decoding procedures are described from this *canonical decoding* point of view, assuming that the horizontal and vertical flipping operations will be performed as required, as a post-processing step.

The canonical decoding procedures, that involve only the canonical scan, involve cells whose upper-left corner always corresponds to a grid-point with even-valued coordinates. Figure 1 shows the arcs for which breakpoint information is decoded within any given cell in the canonical scan, along with their relationships to the grid-points of the cell. For TriBPT code-blocks in CL bands, the canonical decoding procedures involve groups whose upper-left corner always corresponds to a grid-point whose coordinates are divisible by 4, meaning that each group corresponds to an aligned 4-span. These alignment constraints follow from the constraints on precinct size parameters PPx and PPy found in Table 1.

NOTE 1 – To appreciate the significance of this, observe that the vertical and horizontal arcs associated with the top-left cell in the canonical scan both terminate at the top left grid-point in the code-block, while no arcs can terminate at the bottom right grid-point in the code-block. If the CBAP coordinates are $(z_x, z_y) = (1,1)$, so that the arc breakpoint data is flipped horizontally and vertically after canonical decoding, the top left canonical cell's arcs ultimately terminate at the bottom right corner of the code-block, while no arcs terminate at the top left corner of the block.

As already explained, it can happen that the canonical decoding steps need to be followed by horizontal or flipping operations, depending on the CBAP coordinates (z_x, z_y). To be more precise, horizontal flipping ($z_x = 1$), involves the following steps:

- The breakpoint data for an arc centred on a grid-point in column c of the canonical decoded result, is moved to the arc centred at the corresponding grid-point in column $2W_{\text{blk}} - 1 - c$ in the flipped result, where $c = 0$ corresponds to the left edge of the code-block.
- The tick-point value k_b of any break on a horizontal or diagonal arc is converted to a tick-point value $2^{d+F_B} - 1 - k_b$.

Similarly, vertical flipping ($z_y = 1$), involves the following steps:

- The breakpoint data for an arc centred on a grid-point in row r of the canonical decoded result, is moved to the arc centred at the corresponding grid-point in column $2H_{\text{blk}} - 1 - r$ in the flipped result, where $r = 0$ corresponds to the top edge of the code-block.
- The tick-point value k_b of any break on a vertical or diagonal arc is converted to a tick-point value $2^{d+F_B} - 1 - k_b$.

If horizontal and vertical flipping are both required ($z_x = z_y = 1$), then both of the conversions described above are applied and the order of application is not important.

NOTE 2 – The k_b conversions for diagonal arcs cancel each other when ($z_x = z_y = 1$).

Flipping in either direction also causes a change in geometry for code-blocks belonging to one of the TriBPT components, from TriBPT-LR to TriBPT-RL and from TriBPT-RL to TriBPT-LR. These geometry changes cancel when $z_x = z_y = 1$. If, however, $z_x \neq z_y$, the canonical decoding of code-blocks that belong to a TriBPT-LR component shall be performed as if the component had the TriBPT-RL geometry, and vice-versa.

9.4 QuadBPT decoding procedures

9.4.1 MQ Coder contexts and initial states for QuadBPT decoding

Up to 33 different MQ coder contexts are used for decoding QuadBPT CL band code-blocks. Table 2 lists all context labels, along with the initial states for each context. Up to 6 contexts are used for non-root arc significance decoding, the last of these being a special "PARENT" context that is used only in the inter-band coding mode (BPT_INTER = 1). Nine contexts are used for root arc significance and induction block decoding. Sixteen contexts are used for the skip symbols used to identify the point at which magnitude refinement is skipped, if any, while just one context is used for magnitude refinement itself. Finally, a uniform context is provided for use with a run mode that plays the same role as the run mode used for significance decoding of non-breakpoint components in Rec. ITU-T T.800 | ISO/IEC 15444-1.

For LL band code-blocks, decoding procedures are much simpler, using only a subset of the context labels shown in Table 2 and with different interpretations, but the same initial state values. The LL band context labels and their usage are provided in Table 3.

Table 2 – MQ coder contexts for QuadBPT CL code-block decoding passes. State index values correspond to the Qe values and probability estimation transitions tabulated in Rec. ITU-T T.800 | ISO/IEC 15444-1

Usage in CL band code-blocks	Context label(s)	Initial state index	
		MQ probability state index	Initial MPS value
Non-root significance NO-ODD	0	42	0
Non-root significance ANY-ODD-OLD	1	40	0
Non-root significance ONE-ODD-NEW	2	4	0
Non-root significance TWO-ODD-NEW	3	3	0
Non-root significance RUN	4	42	0
Non-root significance PARENT	5	28	0
Root pair significance INSIG	6	38	0
Root pair significance ANY-SIG-OLD	7	40	0
Root pair significance ONE-SIG-NEW	8	4	0
Root pair significance MULTI-SIG-NEW	9	4	0
Root pair significance RUN	10	44	0
Root pair induction block	11-12	0	0
Root significance PRIMARY-DIR	13	0	0
Root significance SECONDARY-DIR	14	0	0
Magnitude bit	15	0	0
Magnitude refinement TERMINATE	16-31	40	0
UNIFORM	32	46	0

Table 3 – Context labels from Table 2 that are used for QuadBPT LL code-block decoding passes

Usage in LL band code-blocks	Context label(s)	Initial state index (unchanged from Table 2)	
		MQ probability state index	Initial MPS value
Significance flag decoding	1	40	0
Vertex flag decoding	2	4	0
Direct flag decoding	3	3	0
Degenerate flag decoding	4	42	0
Magnitude bit	15	0	0
Magnitude refinement TERMINATE	16-31	40	0
UNIFORM	32	46	0

9.4.2 Derivation of context labels for QuadBPT CL band significance coding passes

Context labels employed when processing a (current) 2x2 cell during any of the CL band significance decoding passes are derived using a context neighbourhood that is depicted in Figure 28. In this figure, the root arcs of the current cell are denoted "CV" and "CH", while its non-root arcs are denoted "NV" and "NH". Significance decoding contexts are formed using the significance of additional non-root arcs from neighbouring cells. These neighbouring cells are above (top neighbour), above and to the right (top-right neighbour), to the left (left neighbour), to the right (right neighbour), below (bottom neighbour) and below and to the left (bottom-left neighbour). The relevant non-root arcs are denoted "TV", "TH", "TRV", "LV", "LH", "RV", "BH" and "BLH", respectively. If any of these neighbouring cells lies outside the bounds of the code-block then its arcs are considered to be insignificant for the purpose of context formation.

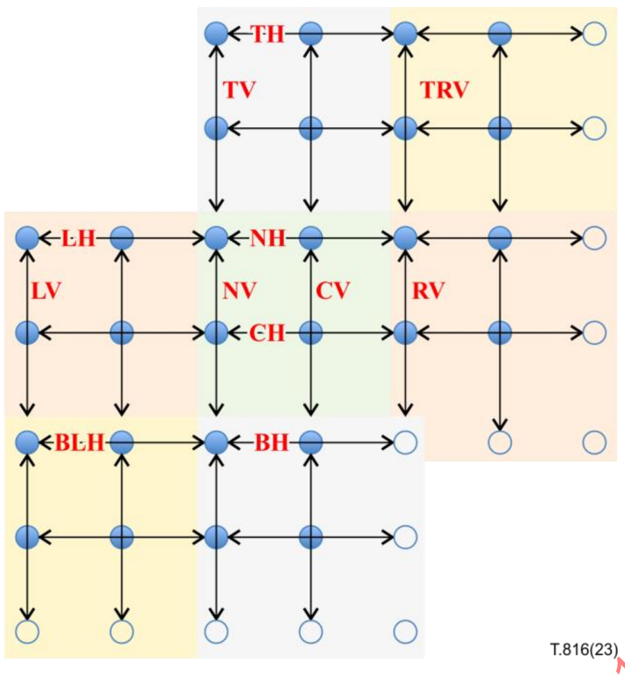


Figure 28 – Neighbouring cells and arcs involved in forming coding contexts for QuadBPT significance decoding in CL band code-blocks

Significance coding contexts for the non-root arcs "NV" and "NH" depend on four "cupping" values, each of which describe the significance of the non-root neighbouring arcs that "cup" each side of the non-root arc being decoded, taking values of 0, 1, 2 or 3. Specifically, writing

$$\bar{\sigma}_b(p) = \sigma_b \mid \sigma'_b(p)$$

for the inclusive OR of an arc b's significance σ_b and its inter-band significance $\sigma'_b(p)$ for the current bit-plane p, the "NV" non-root arc has cupping context values

$$\mathrm{CUP}_{\mathrm{left}} = \bar{\sigma}_{\mathrm{LH}}(p) + \bar{\sigma}_{\mathrm{LV}}(p) + \bar{\sigma}_{\mathrm{BLH}}(p)$$

and

$$\mathrm{CUP}_{\mathrm{right}} = \bar{\sigma}_{\mathrm{NH}}(p) + \bar{\sigma}_{\mathrm{RV}}(p) + \bar{\sigma}_{\mathrm{BH}}(p)$$

Meanwhile, the "NH" non-root arc has cupping values

$$\mathrm{CUP}_{\mathrm{top}} = \bar{\sigma}_{\mathrm{TV}}(p) + \bar{\sigma}_{\mathrm{TH}}(p) + \bar{\sigma}_{\mathrm{TRV}}(p)$$

and

$$\mathrm{CUP}_{\mathrm{bottom}} = \bar{\sigma}_{\mathrm{NV}}(p) + \bar{\sigma}_{\mathrm{BH}}(p) + \bar{\sigma}_{\mathrm{RV}}(p)$$

Significance coding contexts for the non-root arcs "NV" and "NH" also depend on two binary "life" state variables, denoted $\mathrm{LIFE}_{\mathrm{nv}}$ and $\mathrm{LIFE}_{\mathrm{nh}}$, that are initialized to 0 before decoding for the code-block commences. $\mathrm{LIFE}_{\mathrm{nv}}$ is equal to 1 if only if the value of $\mathrm{CUP}_{\mathrm{left}}$ or $\mathrm{CUP}_{\mathrm{right}}$ has increased since the last significance decoding step for non-root arc NV. This property is achieved by setting $\mathrm{LIFE}_{\mathrm{nv}}$ to 1 after any significance decoding step which leaves any of the arcs contributing to $\mathrm{CUP}_{\mathrm{left}}$ or $\mathrm{CUP}_{\mathrm{right}}$ newly significant and then resetting $\mathrm{LIFE}_{\mathrm{nv}}$ to 0 after each significance decoding step for NV itself. Similarly, $\mathrm{LIFE}_{\mathrm{nh}}$ is equal to 1 if and only if the value of $\mathrm{CUP}_{\mathrm{top}}$ or $\mathrm{CUP}_{\mathrm{bottom}}$ has increased since the last significance decoding step for non-root arc NH.

In the inter-band coding mode (BPT_INTER = 1), an increase in cupping values can occur due to a change in any of the $\sigma'_b(p)$ values involved in the construction of that cupping value, in comparison to $\sigma'_b(p+1)$, unless arc b was already significant in an earlier coding pass. To clarify this point, a decoder can correctly incorporate the inter-band coding mode by recomputing all $\sigma'_b(p)$ values immediately before the first significance decoding pass for a bit-plane p, updating all cupping values accordingly, and setting life state variables to 1 if any of the relevant cupping values increases in this process.

The significance context label from Table 2 for non-root arc "NV" or "NH" is derived from these quantities as

$$\kappa_{\mathrm{nv}}^{\mathrm{qbpt}} = \mathrm{KAPPA}_{\mathrm{nrt}}^{\mathrm{qbpt}}(\mathrm{CUP}_{\mathrm{left}}, \mathrm{CUP}_{\mathrm{right}}, \mathrm{LIFE}_{\mathrm{nv}}, \sigma'_{\mathrm{NV}}(\mathrm{p}))$$

and

$$\kappa_{\mathrm{nh}}^{\mathrm{qbpt}} = \mathrm{KAPPA}_{\mathrm{nrt}}^{\mathrm{qbpt}}(\mathrm{CUP}_{\mathrm{top}},\mathrm{CUP}_{\mathrm{bottom}},\mathrm{LIFE}_{\mathrm{nh}}, \sigma'_{\mathrm{NH}}(\mathrm{p})),$$

where function $\mathrm{KAPPA}_{\mathrm{nrt}}^{\mathrm{qbpt}}$ is defined as follows

$$\mathrm{KAPPA}_{\mathrm{nrt}}^{\mathrm{qbpt}}(C_0, C_1, L, \sigma') = \begin{cases} 0 & \text{if } \sigma' = 0, \mathrm{MOD}(C_0, 2) + MOD(C_1, 2) = 0 \\ 1 & \text{if } \sigma' = 0, L = 0, \text{and } \mathrm{MOD}(C_0, 2) + MOD(C_1, 2) \neq 0 \\ 2 & \text{if } \sigma' = 0, L = 1, \text{and } \mathrm{MOD}(C_0, 2) + MOD(C_1, 2) = 1 \\ 3 & \text{if } \sigma' = 0, L = 1, \text{and } \mathrm{MOD}(C_0, 2) + MOD(C_1, 2) = 2 \\ 5 & \text{if } \sigma' = 1 \end{cases}$$

Here, $\mathrm{MOD}(C, 2)$ is equal to 0 if $C \in \{0,2\}$ and 1 if $C \in \{1,3\}$.

NOTE 1 – The $\mathrm{KAPPA}_{\mathrm{nrt}}^{\mathrm{qbpt}}$ function returns one of the context labels 0 through 3 or else 5 (in inter-band coding mode). Context label 4 from Table 2 is used for RUN symbol decoding, which is explained in 9.4.3.

Root arc significance decoding is performed jointly on the CH/CV arc pair. The MQ coding context for root arc significance decoding is formed using the significance of the neighbouring non-root arcs NV, NH, RV and BH shown in Figure 28. Specifically, horizontal and vertical significant boundary count variables are formed as

$$\mathrm{CNT}_{\mathrm{bv}} = \bar{\sigma}_{\mathrm{RV}}(p) + \bar{\sigma}_{\mathrm{NV}}(p)$$

and

$$\mathrm{CNT}_{\mathrm{bh}} = \bar{\sigma}_{\mathrm{BH}}(p) + \bar{\sigma}_{\mathrm{NH}}(p)$$

Additionally, a binary "life" state variable $\mathrm{LIFE}_{\mathrm{root}}$ is employed to derive the MQ coding context for significance decoding of each CH/CV root arc pair. Similar to the other "life" state variables, $\mathrm{LIFE}_{\mathrm{root}}$ is initialized to 0, set to 1 if either of $\mathrm{CNT}_{\mathrm{bv}}$ or $\mathrm{CNT}_{\mathrm{bh}}$ has increased since the last significance decoding step for the CH/CV root arc pair, and is reset to 0 after each significance decoding step for the root arc pair.

As with the other life state variables, a decoder can correctly incorporate the inter-band coding mode (BPT_INTER = 1) by recomputing all $\sigma'_b(p)$ values immediately before the first significance decoding pass for a bit-plane p, updating all significant boundary counts $\mathrm{CNT}_{\mathrm{bv}}$ and $\mathrm{CNT}_{\mathrm{bh}}$ accordingly, and setting $\mathrm{LIFE}_{\mathrm{root}}$ to 1 if either of the associated boundary counts increases in this process.

From these three quantities, the root arc pair significance context label $\kappa_{\mathrm{root}}^{\mathrm{qbpt}}$ from Table 2 is formed from

$$\kappa_{\mathrm{root}}^{\mathrm{qbpt}} = 6 + \begin{cases} 0 & \text{if } (\mathrm{CNT}_{\mathrm{bv}} + \mathrm{CNT}_{\mathrm{bh}}) = 0 \\ 1 & \text{if } (\mathrm{CNT}_{\mathrm{bv}} + \mathrm{CNT}_{\mathrm{bh}}) \neq 0 \text{ and } \mathrm{LIFE}_{\mathrm{root}} = 0 \\ 2 & \text{if } (\mathrm{CNT}_{\mathrm{bv}} + \mathrm{CNT}_{\mathrm{bh}}) = 1 \text{ and } \mathrm{LIFE}_{\mathrm{root}} = 1 \\ 3 & \text{if } (\mathrm{CNT}_{\mathrm{bv}} + \mathrm{CNT}_{\mathrm{bh}}) \geq 2 \text{ and } \mathrm{LIFE}_{\mathrm{root}} = 1 \end{cases}$$

The quantities $\mathrm{CNT}_{\mathrm{bv}}$ and $\mathrm{CNT}_{\mathrm{bh}}$ are used to derive two other important quantities for significance decoding of root arcs CH and CV. The first of these is the context label $\kappa_{\mathrm{iblock}}^{\mathrm{qbpt}}$ that is used to decode an induction block symbol for a root arc pair that has just become significant. This label is found from

$$\kappa_{\mathrm{iblock}}^{\mathrm{qbpt}} = \begin{cases} 11 & \text{if } \mathrm{CNT}_{\mathrm{bh}} + \mathrm{CNT}_{\mathrm{bv}} = 2 \\ 12 & \text{otherwise} \end{cases}$$

The second quantity derived from $\mathrm{CNT}_{\mathrm{bv}}$ and $\mathrm{CNT}_{\mathrm{bh}}$ is a primary direction label

$$D_{\mathrm{root}}^{\mathrm{qbpt}} = \begin{cases} 0 & \text{if } \mathrm{CNT}_{\mathrm{bh}} \geq \mathrm{CNT}_{\mathrm{bv}} \\ 1 & \text{if } \mathrm{CNT}_{\mathrm{bh}} < \mathrm{CNT}_{\mathrm{bv}} \end{cases}$$

which identifies whether the significance of CH ($D_{\mathrm{root}}^{\mathrm{qbpt}} = 0$) or CV ($D_{\mathrm{root}}^{\mathrm{qbpt}} = 1$) is decoded first, once a root-arc pair has been found to be significant and not an induction block.

NOTE 2 – The root significance coding context labels defined above take values from 6 through 9 and 11 through 14. Context label 10 from Table 2 is used for RUN symbol decoding, which is explained in 9.4.4.

9.4.3 Non-root significance decoding for QuadBPT CL code-blocks

The non-root significance decoding pass for bit-plane p within a QuadBPT CL band code-block processes groups of 2x2 cells in raster order, as shown in Figure 25. For each such cell group, the decoder enters a special "RUN mode" if all cells in the group are found to have entirely insignificant context. A cell's context is entirely insignificant if the cell's four cupping values $\mathrm{CUP}_{\mathrm{left}}$, $\mathrm{CUP}_{\mathrm{right}}$, $\mathrm{CUP}_{\mathrm{top}}$ and $\mathrm{CUP}_{\mathrm{bottom}}$ are all 0, and its root arcs CH and CV are also insignificant. Equivalently, a cell's context is entirely insignificant if all of the labelled arcs b shown in Figure 28 have $\sigma_b = 0$ and $\sigma'_b(p) = 0$, this second condition being relevant only in the inter-band coding mode. When all cells of the group satisfy this condition, the MQ decoder is invoked with context label $\kappa = 4$ (non-root significance RUN context – see Table 2) to decode an end-of-run symbol.

If the end-of-run symbol is 0, the entire group remains insignificant and decoding proceeds to the next cell group, if any. If the end-of-run symbol is 1, then at least one of the non-root arcs in at least one of the group's cells is significant. The MQ decoder is then used with context label $\kappa = 32$ (UNIFORM context – see Table 2) to decode three consecutive symbols R_a, R_b and then R_c. A cell run-length is formed as $R = 2R_a + R_b$, identifying the first cell in the group that contains a significant non-root arc, following the raster order shown in Figure 25. If R_c=0, decoding continues from the identified cell with flag SKIP_TO_NH set to true, meaning that the cell's NV arc is insignificant and the first significant arc of the cell is NH. Otherwise, if R_c=1, decoding continues from the identified cell with flag SKIP_TO_NV set to true, meaning that the cell's NV arc is significant.

After processing the cell identified by run-length R, any remaining cells in the group are processed in raster order, with flags SKIP_TO_NH and SKIP_TO_NV both set to false. If a cell group does not have an entirely insignificant context, all of its cells are processed in raster order, with flags SKIP_TO_NH and SKIP_TO_NV both set to false.

For each cell, the decoder performs decoding steps first for the vertical non-root arc NV and then for the horizontal non-root arc NH. Specifically, if SKIP_TO_NH is false and $\sigma_{\mathrm{NV}} = 0$, the decoder performs the following NV decoding steps:

V1. If SKIP_TO_NV is false, the MQ decoder is invoked with context label $\kappa_{\mathrm{nv}}^{\mathrm{qbpt}}$ to decode the SIG symbol; otherwise, the SKIP_TO_NV is true, and the decoder sets SIG to 1.
V2. If SIG = 1, significance state σ_{NV} is set to 1, the cupping values $\mathrm{CUP}_{\mathrm{left}}$, $\mathrm{CUP}_{\mathrm{right}}$, $\mathrm{CUP}_{\mathrm{top}}$ and $\mathrm{CUP}_{\mathrm{bottom}}$ of all cells that depend on σ_{NV} are updated, and the life values $\mathrm{LIFE}_{\mathrm{nv}}$, $\mathrm{LIFE}_{\mathrm{nh}}$ and $\mathrm{LIFE}_{\mathrm{root}}$ of all cells that depend on σ_{NV} are set to 1.
V3. If SIG = 1, the MQ decoder is invoked with context label $\kappa = 15$ (magnitude bit context – see Table 2) to decode the most significant value bit m_b of the arc b=NV. Then, the type, precision and value for the arc are set to $t_b = 3$, $P_b = 2$ and $V_b = m_b$.
V4. The $\mathrm{LIFE}_{\mathrm{nv}}$ state variable for the processed cell itself is reset to 0.

If $\sigma_{\mathrm{NH}} = 0$, the decoder then performs the following NH decoding steps:
H1. If SKIP_TO_NH is false, the MQ decoder is invoked with context label $\kappa_{\mathrm{nh}}^{\mathrm{qbpt}}$ to decode the SIG symbol; otherwise, the SKIP_TO_NH is true, and the decoder sets SIG to 1.
H2. If SIG = 1, significance state σ_{NH} is set to 1, the cupping values $\mathrm{CUP}_{\mathrm{left}}$, $\mathrm{CUP}_{\mathrm{right}}$, $\mathrm{CUP}_{\mathrm{top}}$ and $\mathrm{CUP}_{\mathrm{bottom}}$ of all cells that depend on σ_{NH} are updated, and the life values $\mathrm{LIFE}_{\mathrm{nv}}$, $\mathrm{LIFE}_{\mathrm{nh}}$ and $\mathrm{LIFE}_{\mathrm{root}}$ of all cells that depend on σ_{NH} are set to 1.
H3. If SIG = 1, the MQ decoder is invoked with context label $\kappa = 15$ (magnitude bit context – see Table 2) to decode the most significant value bit m_b of the arc b=NH. Then, the type, precision and value for the arc are set to $t_b = 3$, $P_b = 2$ and $V_b = m_b$.
H4. The $\mathrm{LIFE}_{\mathrm{nh}}$ state variable for the processed cell itself is reset to 0.

9.4.4 Root significance decoding for QuadBPT CL code-blocks

The root significance decoding pass for bit-plane p within a QuadBPT CL band code-block processes groups of 2x2 cells in raster order, as shown in Figure 25. For each such cell group, the decoder enters a special "RUN mode" if all cells in the group are found to have entirely insignificant context. Exactly as in 9.4.3, a cell's context is entirely insignificant if all of the labelled arcs b shown in Figure 28 have $\sigma_b = 0$ and $\sigma_b'(p) = 0$, this second condition being relevant only in the inter-band coding mode (BPT_INTER = 1). When all cells of the group satisfy this condition, the MQ decoder is invoked with context label $\kappa = 15$ (root significance RUN context – see Table 2) to decode an end-of-run symbol.

If the end-of-run symbol is 0, the entire group remains insignificant and decoding proceeds to the next cell group, if any. If the end-of-run symbol is 1, then at least one of the root arcs in at least one of the group's cells is significant. The MQ decoder is then used with context label $\kappa = 32$ (UNIFORM context – see Table 2) to decode two consecutive symbols R_a and then R_b. A cell run-length is formed as $R = 2R_a + R_b$, identifying the first cell in the group that contains a significant root arc pair, following the raster order shown in Figure 25. Decoding continues from the identified cell with flag SKIP_TO_SIG set to true.

After processing the cell identified by run-length R, any remaining cells in the group are processed in raster order, with flag SKIP_TO_SIG set to false. If a cell group does not have an entirely insignificant context, all of its cells are processed in raster order, with flag SKIP_TO_SIG set to false.

For each cell, if $\sigma_{\mathrm{CV}} = 0$ and $\sigma_{\mathrm{CH}} = 0$, the decoder performs the following steps:

1. If SKIP_TO_SIG is false, the MQ decoder is invoked with context label $\kappa_{\mathrm{root}}^{\mathrm{qbpt}}$ to decode the SIG symbol; otherwise, the SKIP_TO_SIG is true and the decoder sets SIG to 1.
2. If SIG = 1, the MQ decoder is invoked with context label $\kappa_{\mathrm{iblock}}^{\mathrm{qbpt}}$ to decode symbol IBLOCK. If IBLOCK=1, the decoder sets $\sigma_b = 1$, $t_b = 0$, $P_b = 1$ (induction block) and $\tau_b = 0$ (not refineable), for both of the root arcs $b \in \{\mathrm{CV,CH}\}$. Otherwise, IBLOCK=0 and the decoder performs the following sub-steps:

a. The MQ decoder is invoked with context label $\kappa = 13$ (PRIMARY-DIR context – see Table 2) to decode symbol SIG. If SIG = 0 flag SKIP_TO_SECONDARY is set to true. Otherwise, SKIP_TO_SECONDARY is set to false and the MQ decoder is invoked with context label $\kappa = 15$ (magnitude bit context) to decode the most significant value bit m_b of the primary root arc b, setting $\sigma_b = 1$, $t_b = 3$, $P_b = 2$ and $V_b = m_b$, where

$$b = \begin{cases} \text{CH} & \text{if } D_{\text{root}}^{\text{qbpt}} = 0 \\ \text{CV} & \text{if } D_{\text{root}}^{\text{qbpt}} = 1 \end{cases}$$

b. If SKIP_TO_SECONDARY is set to false, the MQ decoder is invoked with context label $\kappa = 14$ (SECONDARY-DIR context – see Table 2) to decode symbol SIG; otherwise, SKIP_TO_SECONDARY is true, and SIG is set to 1.

c. If SIG = 1, the MQ decoder is invoked with context label $\kappa = 15$ (magnitude bit context – see Table 2) to decode the most significant value bit m_b of the secondary root arc b, setting $\sigma_b = 1$, $t_b = 3$, $P_b = 2$ and $V_b = m_b$, where

$$b = \begin{cases} \text{CV} & \text{if } D_{\text{root}}^{\text{qbpt}} = 0 \\ \text{CH} & \text{if } D_{\text{root}}^{\text{qbpt}} = 1 \end{cases}$$

3. The $\text{LIFE}_{\text{root}}$ state variable for the processed cell is reset to 0.

9.4.5 Root significance decoding for QuadBPT LL code-blocks

The significance decoding pass for bit-plane p within a QuadBPT LL band code-block processes 2x2 cells in raster order. In the canonical scan, rows of cells are visited from top to bottom, cells of each row are visited from left to right, and the four arcs b of each cell are visited in the order NV, CV, NH then CH.

For each arc b, following this scanning order, if $\sigma_b = 0$, the MQ decoder is invoked with context label κ=1 (significance flag context – see Table 3) to decode symbol SIG. If SIG = 0, arc b remains insignificant. Otherwise, the following steps are performed:

1. The significance state for arc b is changed to $\sigma_b = 1$
2. The MQ decoder is invoked with context label $\kappa = 2$ (vertex flag context – see Table 3) to decode symbol VTX.
3. If VTX = 1, the breakpoint type for arc b is set to $t_b = 3$. Otherwise, the MQ decoder is invoked with context label $\kappa = 3$ (direct flag context – see Table 3) to decode symbol DIRECT, and the type for arc b is set to $t_b = 1 + \text{DIRECT}$.
4. VTX = 0, the MQ decoder is invoked with context label $\kappa = 4$ (degenerate flag context – see Table 3) to decode symbol DEGEN; otherwise DEGEN is set to 0.
5. If DEGEN = 1, the following steps are performed:
   a. The MQ decoder is invoked with context label $\kappa = 32$ (uniform context – see Table 3) to decode symbol AMBIVALENT.
   b. The precision and value parameters for arc b are set to $P_b = \text{AMBIVALENT}$ and $V_b = 0$ and the refinement counter is set to $\tau_b = 0$ (not refineable).
6. Otherwise, DEGEN = 0 and the MQ decode is invoked with context label 15 (magnitude bit context) to decode the most significant value bit m_b of arc b. Then, the precision and value for the arc are set to $t_b = 3$, $P_b = 2$ and $V_b = m_b$.

NOTE 1 – Coding procedures for LL band code-blocks are intended to be very simple, since in most cases all LL code-blocks are expected to be entirely insignificant. Coding tools are provided for the LL band of a breakpoint component primarily to allow the communication of breaks when N_L is heavily constrained, or even 0. For example, the breakpoints described by the breakpoint media format suggested in Annex B can be encapsulated directly within the LL band of a breakpoint component with $N_L = 0$ decomposition levels.

NOTE 2 – No means is provided to identify whether or not an arc without a break might hold an induction block, whereas this information can be recorded (as a hint) within the breakpoint media format suggested in Annex B. Induction blocks play no role in breakpoint dependent transforms and so there is no need to record them within resolution $r = 0$ of a breakpoint component.

9.4.6 Position refinement decoding for QuadBPT code-blocks

Position refinement decoding is the same for CL and LL band code-blocks of a QuadBPT component, using MQ coder context labels 15 through 31 from Table 2 and Table 3, and processing 2×2 cells in raster order. In the canonical scan, rows of cells are visited from top to bottom, cells of each row are visited from left to right, and the four arcs b of each cell are visited in the order NV, CV, NH then CH.

For each arc b, following this scanning order, if $\sigma_b = 1$ and $\tau_b > 0$, the following steps are performed:

1. The MQ decoder is invoked with context label $\kappa = 16 + \min\{(P_b - 2), 15\}$ to decode symbol TERM.
2. If TERM = 1, the decoder sets $\tau_b = 0$

3. Otherwise, TERM = 0 and the MQ decoder is invoked with context label $\kappa = 15$ (magnitude refinement context) to decode refinement bit r_b, after which the decoder the arc's state variables as follows: $P_b \coloneqq P_b + 1$; $V_b \coloneqq 2V_b + r_b$ and $\tau_b \coloneqq \tau_b - 1$.

9.5 TriBPT decoding procedures

9.5.1 MQ Coder contexts and initial states for TriBPT decoding

Up to 27 different MQ coder contexts are used for decoding TriBPT CL band code-blocks. Table 4 lists all context labels, along with the initial states for each context. Up to 4 contexts are used for non-root arc significance decoding, the last of these being a special "PARENT" context that is used only in the inter-band coding mode (BPT_INTER = 1). 5 contexts are used for root arc significance and induction block decoding. 16 contexts are used for the skip symbols used to identify the point at which magnitude refinement is skipped, if any, while just one context is used for magnitude refinement itself. Finally, a uniform context is provided for use with a run mode that plays the same role as the run mode used for significance decoding of non-breakpoint components in Rec. ITU-T T.800 | ISO/IEC 15444-1.

For LL band code-blocks, decoding procedures are much simpler, using only a subset of the context labels shown in Table 4 and with different interpretations, but the same initial state values. The LL band context labels and their usage are provided in Table 5.

Table 4 – MQ coder contexts for TriBPT CL code-block decoding passes. State index values correspond to the Qe values and probability estimation transitions tabulated in Rec. ITU-T T.800 | ISO/IEC 15444-1

Usage in CL band code-blocks	Context label(s)	Initial state index	
		MQ probability state index	Initial MPS value
Non-root significance INSIG	0	38	0
Non-root significance ANY-CUP	1	5	0
Non-root significance RUN	2	43	0
Non-root significance PARENT	3	38	0
Root significance INSIG	4	38	0
Root significance ANY-VTX	5	3	0
Root significance RUN	6	40	0
Root induction block	7	43	0
Root induction block	8	4	0
Magnitude bit	9	0	0
Magnitude refinement TERMINATE	10-25	40	0
UNIFORM	26	46	0

Table 5 – Context labels from Table 4 that are used for TriBPT LL code-block decoding passes

Usage in LL band code-blocks	Context label(s)	Initial state (unchanged from Table 4)	
		MQ probability state index	Initial MPS value
Significance flag decoding	1	5	0
Vertex flag decoding	5	3	0
Direct flag decoding	8	4	0
Degenerate flag decoding	6	40	0
Magnitude bit	9	0	0
Magnitude refinement TERMINATE	10-25	40	0
UNIFORM	26	46	0

9.5.2 Derivation of context labels for TriBPT CL band significance coding passes

Context labels employed when processing a (current) 4-span group during any of the CL band significance decoding passes are derived using context neighbourhoods that are depicted in Figure 29 and Figure 30, for the TriLR and TriRL orientations, respectively. In these figures, neighbouring arcs are labelled according to the relative location of their 4-span group, using a notation derived from directions on the compass: arcs coming from the 4-span group above the one being processed are labelled with the suffix "n" for "north"; arcs coming from the 4-span group to the right are labelled with the suffix "e" for "east"; arcs coming from the 4-span group above and to the right of the one being processed are labelled with the suffix "ne" for "north-east"; and so forth. If any of the neighbouring cells (shaded grey in Figure 29 and Figure 30) lie outside the boundaries of the code-block, their arcs shall be treated as insignificant for the purpose of generating context labels.

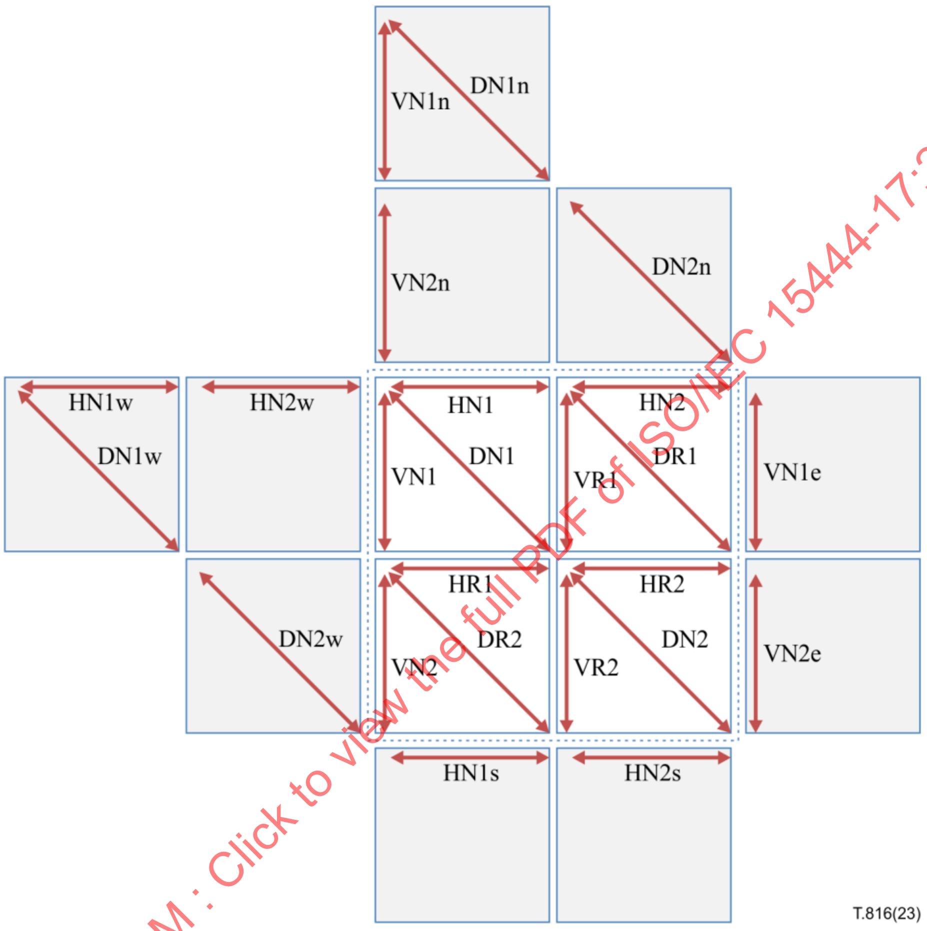


Figure 29 – Neighbouring cells and arcs involved in forming coding contexts for TriLR significance decoding in CL band code-blocks

For non-root significance decoding, arcs are grouped into pairs (VN1,VN2), (HN1,HN2) and (DN1,DN2). The context label used for the arcs in each pair are formed using two binary "cupping" values. The cupping values $\mathrm{CUP}_{\mathrm{Va}}$ and $\mathrm{CUP}_{\mathrm{Vb}}$ identify the significance of any of the neighbouring non-root arcs that surround the (VN1,VN2) pair on the left and right, respectively. Specifically, writing

$$\bar{\sigma}_b(p) = \sigma_b \mid \sigma'_b(p)$$

for the inclusive OR of an arc b's significance σ_b and its inter-band significance $\sigma'_b(p)$ for the current bit-plane p, the vertical non-root arc cupping values are given by

$$\mathrm{CUP}_{\mathrm{Va}} = \begin{cases} \bar{\sigma}_{\mathrm{HN1w}}(p) \mid \bar{\sigma}_{\mathrm{HN2w}}(p) \mid \bar{\sigma}_{\mathrm{DN1w}}(p) \mid \bar{\sigma}_{\mathrm{DN2w}}(p) & \text{for TriLR} \\ \bar{\sigma}_{\mathrm{HN1sw}}(p) \mid \bar{\sigma}_{\mathrm{HN2sw}}(p) \mid \bar{\sigma}_{\mathrm{DN1w}}(p) \mid \bar{\sigma}_{\mathrm{DN2w}}(p) & \text{for TriRL} \end{cases}$$

$$\mathrm{CUP}_{\mathrm{Vb}} = \begin{cases} \bar{\sigma}_{\mathrm{HN1s}}(p) \mid \bar{\sigma}_{\mathrm{HN2s}}(p) \mid \bar{\sigma}_{\mathrm{DN1}}(p) \mid \bar{\sigma}_{\mathrm{DN2}}(p) & \text{for TriLR} \\ \bar{\sigma}_{\mathrm{HN1}}(p) \mid \bar{\sigma}_{\mathrm{HN2}}(p) \mid \bar{\sigma}_{\mathrm{DN1}}(p) \mid \bar{\sigma}_{\mathrm{DN2}}(p) & \text{for TriRL} \end{cases}$$

The horizontal non-root arc pair cupping values, denoted CUP_{Ha} and CUP_{Hb}, are given by

$$\text{CUP}_{\text{Ha}} = \begin{cases} \bar{\sigma}_{\text{VN1n}}(p) \mid \bar{\sigma}_{\text{VN2n}}(p) \mid \bar{\sigma}_{\text{DN1n}}(p) \mid \bar{\sigma}_{\text{DN2n}}(p) & \text{for TriLR} \\ \bar{\sigma}_{\text{VN1ne}}(p) \mid \bar{\sigma}_{\text{VN2ne}}(p) \mid \bar{\sigma}_{\text{DN1n}}(p) \mid \bar{\sigma}_{\text{DN2n}}(p) & \text{for TriRL} \end{cases}$$

$$\text{CUP}_{\text{Hb}} = \begin{cases} \bar{\sigma}_{\text{VN1e}}(p) \mid \bar{\sigma}_{\text{VN2e}}(p) \mid \bar{\sigma}_{\text{DN1}}(p) \mid \bar{\sigma}_{\text{DN2}}(p) & \text{for TriLR} \\ \bar{\sigma}_{\text{VN1}}(p) \mid \bar{\sigma}_{\text{VN2}}(p) \mid \bar{\sigma}_{\text{DN1}}(p) \mid \bar{\sigma}_{\text{DN2}}(p) & \text{for TriRL} \end{cases}$$

Finally, the diagonal non-root arc pair cupping values, denoted CUP_{Da} and CUP_{Db}, are given by

$$\text{CUP}_{\text{Da}} = \begin{cases} \bar{\sigma}_{\text{VN1}}(p) \mid \bar{\sigma}_{\text{VN2}}(p) \mid \bar{\sigma}_{\text{HN1s}}(p) \mid \bar{\sigma}_{\text{HN2s}}(p) & \text{for TriLR} \\ \bar{\sigma}_{\text{VN1}}(p) \mid \bar{\sigma}_{\text{VN2}}(p) \mid \bar{\sigma}_{\text{HN1}}(p) \mid \bar{\sigma}_{\text{HN2}}(p) & \text{for TriRL} \end{cases}$$

$$\text{CUP}_{\text{Db}} = \begin{cases} \bar{\sigma}_{\text{VN1e}}(p) \mid \bar{\sigma}_{\text{VN2e}}(p) \mid \bar{\sigma}_{\text{HN1}}(p) \mid \bar{\sigma}_{\text{HN2}}(p) & \text{for TriLR} \\ \bar{\sigma}_{\text{VN1e}}(p) \mid \bar{\sigma}_{\text{VN2e}}(p) \mid \bar{\sigma}_{\text{HN1s}}(p) \mid \bar{\sigma}_{\text{HN2s}}(p) & \text{for TriRL} \end{cases}$$

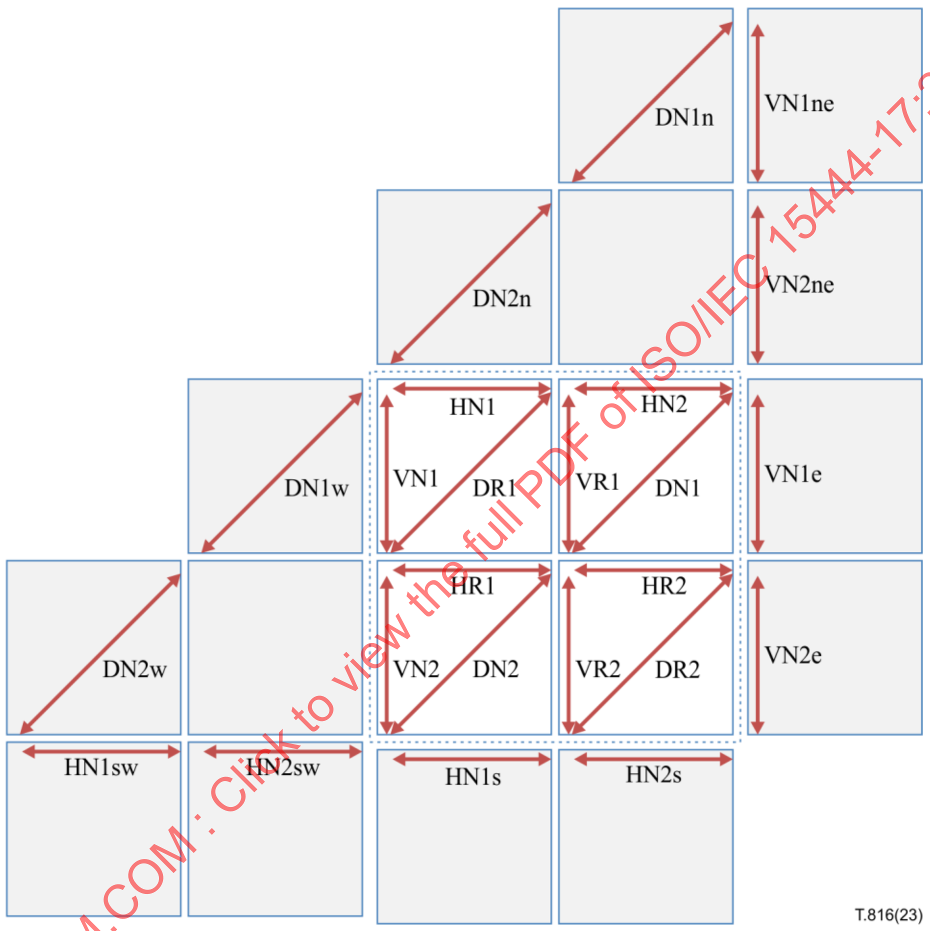


Figure 30 – Neighbouring cells and arcs involved in forming coding contexts for TriRL significance decoding in CL band code-blocks

The MQ context labels used for significance decoding of any of the non-root arcs in the 4-span group are given by

$\kappa_{\text{VN1}}^{\text{tbpt}} = \text{KAPPA}_{\text{nrt}}^{\text{tbpt}}(\text{CUP}_{\text{Va}},\text{CUP}_{\text{Vb}},\sigma'_{\text{VN1}}(\text{p}));$

$\kappa_{\text{VN2}}^{\text{tbpt}} = \text{KAPPA}_{\text{nrt}}^{\text{tbpt}}(\text{CUP}_{\text{Va}},\text{CUP}_{\text{Vb}},\sigma'_{\text{VN2}}(\text{p}));$

$\kappa_{\text{HN1}}^{\text{tbpt}} = \text{KAPPA}_{\text{nrt}}^{\text{tbpt}}(\text{CUP}_{\text{Ha}},\text{CUP}_{\text{Hb}},\sigma'_{\text{HN1}}(\text{p}));$

$\kappa_{\text{HN2}}^{\text{tbpt}} = \text{KAPPA}_{\text{nrt}}^{\text{tbpt}}(\text{CUP}_{\text{Ha}},\text{CUP}_{\text{Hb}},\sigma'_{\text{HN2}}(\text{p}));$

$\kappa_{\text{DN1}}^{\text{tbpt}} = \text{KAPPA}_{\text{nrt}}^{\text{tbpt}}(\text{CUP}_{\text{Da}},\text{CUP}_{\text{Db}},\sigma'_{\text{DN1}}(\text{p}));$ and

$\kappa_{\text{DN2}}^{\text{tbpt}} = \text{KAPPA}_{\text{nrt}}^{\text{tbpt}}(\text{CUP}_{\text{Da}},\text{CUP}_{\text{Db}},\sigma'_{\text{DN2}}(\text{p}))$

where function $\mathrm{KAPPA}_{\mathrm{nrt}}^{\mathrm{tbpt}}$ is defined as follows

$$\mathrm{KAPPA}_{\mathrm{nrt}}^{\mathrm{tbpt}}(C_0, C_1, \sigma') = \begin{cases} \max\{C_0, C_1\} & \text{if } \sigma' = 0 \\ 3 & \text{if } \sigma' = 1 \end{cases}$$

Each of the context labels is evaluated immediately prior to decoding an MQ symbol to determine whether the non-root arc in question is transitioning from insignificant to significant.

NOTE 1 – The $\mathrm{KAPPA}_{\mathrm{nrt}}^{\mathrm{tbpt}}$ function returns one of the context labels 0, 1 or else 3 (in inter-band coding mode). Context label 2 from Table 4 is used for RUN symbol decoding, which is explained in 9.5.3.

The MQ context labels used in the root significance pass are based on two quantities for each "root arc complex." A root arc complex consists of 3 root arcs that are surrounded by 6 non-root arcs. It is convenient to label the root arc complexes at the top and bottom of a 4-span group as $C_T = (V_T, H_T, D_T)$ and $C_B = (V_B, H_B, D_B)$, respectively, and to label the 6 non-root arcs surrounding each complex as $(V1_T, V2_T, H1_T, H2_T, D1_T, D2_T)$ and $(V1_B, V2_B, H1_B, H2_B, D1_B, D2_B)$. These labels are associated with the arcs shown in Figure 29 and Figure 30, for the TriLR and TriRL configurations, as follows:

$$(V_T, H_T, D_T) = \begin{cases} (VR1, HR2, DR1) & \text{for TriLR} \\ (VR1, HR1, DR1) & \text{for TriRL} \end{cases}$$

$$(V_B, H_B, D_B) = \begin{cases} (VR2, HR1, DR2) & \text{for TriLR} \\ (VR2, HR2, DR2) & \text{for TriRL} \end{cases}$$

$$(V1_T, V2_T, H1_T, H2_T, D1_T, D2_T) = \begin{cases} (VN1e, VN2e, HN1, HN2, DN1, DN2) & \text{for TriLR} \\ (VN1, VN2, HN1, HN2, DN1, DN2) & \text{for TriRL} \end{cases}$$

$$(V1_B, V2_B, H1_B, H2_B, D1_B, D2_B) = \begin{cases} (VN1, VN2, HN1s, HN2s, DN1, DN2) & \text{for TriLR} \\ (VN1e, VN2e, HN1s, HN2s, DN1, DN2) & \text{for TriRL} \end{cases}$$

The first quantity used to determine root significance context labels for each complex is the joint significance of the surrounding non-root arc pairs, given by

$$\mathrm{NBR}_T = \left(\bar{\sigma}_{V1_T}(p) \mid \bar{\sigma}_{V2_T}(p)\right) \mid \left(\bar{\sigma}_{H1_T}(p) \mid \bar{\sigma}_{H2_T}(p)\right) \mid \left(\bar{\sigma}_{D1_T}(p) \mid \bar{\sigma}_{D2_T}(p)\right) \in \{0,1\}$$

$$\mathrm{NBR}_B = \left(\bar{\sigma}_{V1_B}(p) \mid \bar{\sigma}_{V2_B}(p)\right) \mid \left(\bar{\sigma}_{H1_B}(p) \mid \bar{\sigma}_{H2_B}(p)\right) \mid \left(\bar{\sigma}_{D1_B}(p) \mid \bar{\sigma}_{D2_B}(p)\right) \in \{0,1\}$$

The second quantity is a binary flag indicating whether or not any other root arcs in the complex has been found to be significant as a vertex – i.e., not an induction block. These flags, denoted VTX_T and VTX_B are updated after each root arc is processed in the root significance decoding pass. Specifically, VTX_T and VTX_B can be understood as binary state variables that are initialized to 0 at the start of the block decoding procedure and set to 1 after any root arc in the corresponding root arc complex is decoded as a vertex.

The MQ context labels used in the root significance coding pass are given by

$$\kappa_{C_T}^{\mathrm{tbpt}} = \mathrm{KAPPA}_{\mathrm{root}}^{\mathrm{tbpt}}(\mathrm{NBR_T}, \mathrm{VTX_T})$$

and

$$\kappa_{C_B}^{\mathrm{tbpt}} = \mathrm{KAPPA}_{\mathrm{root}}^{\mathrm{tbpt}}(\mathrm{NBR_B}, \mathrm{VTX_B})$$

where function $\mathrm{KAPPA}_{\mathrm{root}}^{\mathrm{tbpt}}$ is defined as follows:

$$\mathrm{KAPPA}_{\mathrm{root}}^{\mathrm{tbpt}}(N, V) = 4 + \max\{N, V\}$$

The quantities $\mathrm{NBR_T}$, $\mathrm{NBR_B}$, $\mathrm{VTX_T}$ and $\mathrm{VTX_B}$ are also used to derive context labels $\kappa_{\mathrm{ibT}}^{\mathrm{tbpt}}$ and $\kappa_{\mathrm{ibB}}^{\mathrm{tbpt}}$ that are used to decode induction block symbols for each newly significant root arc in complex C_T or C_B, as appropriate. These labels are found from

$$\kappa_{\mathrm{ibT}}^{\mathrm{qbpt}} = \begin{cases} 7 & \text{if } \mathrm{NBR_T} < 2 \text{ or } \mathrm{VTX_T} \neq 0 \\ 8 & \text{if } \mathrm{NBR_T} \geq 2 \text{ and } \mathrm{VTX_T} = 0 \end{cases}$$

and

$$\kappa_{\mathrm{ibB}}^{\mathrm{qbpt}} = \begin{cases} 7 & \text{if } \mathrm{NBR_B} < 2 \text{ or } \mathrm{VTX_B} \neq 0 \\ 8 & \text{if } \mathrm{NBR_B} \geq 2 \text{ and } \mathrm{VTX_B} = 0 \end{cases}$$

NOTE 2 – The root significance decoding contexts described above involve context labels 4, 5, 7 and 8 only. Context label 6 from Table 4 is used for RUN symbol decoding, which is explained in 9.5.4.

9.5.3 Non-root significance decoding for TriBPT CL code-blocks

The non-root significance decoding pass for bit-plane p within a TriBPT CL band code-block processes 4-span groups in raster order, as shown in Figure 26 and Figure 27. For each 4-span group, the decoder enters a special "RUN mode" if the 4-span group has an entirely insignificant context, meaning that all cupping values are 0 and all root arcs within the group are also insignificant. That is, a 4-span group has entirely insignificant context if $\mathrm{CUP_{Va}}=\mathrm{CUP_{Vb}}=\mathrm{CUP_{Ha}}=\mathrm{CUP_{Hb}}=\mathrm{CUP_{Da}}=\mathrm{CUP_{Db}}=0$, $\sigma_{VR1}=\sigma_{VR2}=0$, $\sigma_{HR1}=\sigma_{HR2}=0$ and $\sigma_{DR1}=\sigma_{DR2}=0$. When this condition is satisfied, the MQ decoder is invoked with context label $\kappa=2$ (non-root significance RUN context – see Table 4) to decode an end-of-run symbol.

If the end-of-run symbol is 0, the entire group remains insignificant and decoding proceeds to the next 4-span group, if any. If the end-of-run symbol is 1, then at least one of the 6 non-root arcs in the 4-span group is significant. The MQ decoder is then used with context label 26 (UNIFORM context – see Table 4) to decode either 2 or 3 symbols, so as to determine values SKIP_TO_NV, SKIP_TO_NH and SKIP_TO_ND in accordance with the following steps:

1. SKIP_TO_NV, SKIP_TO_NH and SKIP_TO_ND are first initialized to 0.
2. The MQ decoder is invoked with context label $\kappa=26$ to obtain symbol Ra.
3. If Ra = 1, the MQ decoder is invoked with $\kappa=26$ to obtain symbol Rc, setting SKIP_TO_NV=2-Rc
4. Otherwise, Ra = 0, and the following steps are performed:
   a. SKIP_TO_NV is set to 2
   b. The MQ decoder is invoked twice with $\kappa=26$ to obtain symbol Rb and then symbol Rc:
   c. If Rb = 1, the decoder sets SKIP_TO_NH=2-Rc
   d. Otherwise, Rb = 0 and the decoder sets SKIP_TO_NH=2 and SKIP_TO_ND=2-Rc

If the 4-span group does not have an entirely insignificant context, SKIP_TO_NV, SKIP_TO_NH and SKIP_TO_ND are all set to 0. In this case, and in the case where an end-of-run symbol of 1 was decoded, the decoder performs decoding steps first for the vertical non-root arcs VN1 and VN2, then for the horizontal non-root arcs HN1 and HN2, and finally for the diagonal non-root arcs DN1 and DN2, of the 4-span group, as follows:

If SKIP_TO_NV < 2 and $\sigma_{\mathrm{VN1}}=0$, the decoder performs the following decoding steps for arc VN1:

V1. If SKIP_TO_NV = 0, the MQ decoder is invoked with context label $\kappa_{\mathrm{VN1}}^{\mathrm{tbpt}}$ to decode the SIG symbol; otherwise, SKIP_TO_NV = 1 and the decoder sets SIG to 1.
V2. If SIG = 1, significance state σ_{VN1} is set to 1 and the following steps are performed:
   a. The cupping values of all cells that depend on σ_{VN1} are updated.
   b. The MQ decoder is invoked with context label $\kappa=9$ (magnitude bit context – see Table 4) to decode the most significant value bit m_b of the arc b=VN1, and the type, precision and value for the arc are set to $t_b=3$, $P_b=2$ and $V_b=m_b$.

Next, if SKIP_TO_NH = 0 and $\sigma_{\mathrm{VN2}}=0$, the decoder performs the following steps for arc VN2:

V3. If SKIP_TO_NV < 2, the MQ decoder is invoked with context label $\kappa_{\mathrm{VN2}}^{\mathrm{tbpt}}$ to decode the SIG symbol; otherwise, the SKIP_TO_NV = 2 and the decoder sets SIG to 1.
V4. If SIG = 1, significance state σ_{VN2} is set to 1 and the following steps are performed:
   a. The cupping values of all cells that depend on σ_{VN2} are updated.
   b. The MQ decoder is invoked with context label $\kappa=9$ (magnitude bit context – see Table 4) to decode the most significant value bit m_b of the arc b=VN2, and the type, precision and value for the arc are set to $t_b=3$, $P_b=2$ and $V_b=m_b$.

Next, $\mathrm{LIFE_V}$ is decremented, unless it is already 0, and then if SKIP_TO_NH < 2 and $\sigma_{\mathrm{HN1}}=0$, the decoder performs the following steps for arc HN1:

H1. If SKIP_TO_NH = 0, the MQ decoder is invoked with context label $\kappa_{\mathrm{HN1}}^{\mathrm{tbpt}}$ to decode the SIG symbol; otherwise, SKIP_TO_NH = 1 and the decoder sets SIG to 1.
H2. If SIG = 1, significance state σ_{HN1} is set to 1 and the following steps are performed:
   a. The cupping values of all cells that depend on σ_{HN1} are updated.
   b. The MQ decoder is invoked with context label $\kappa=9$ (magnitude bit context – see Table 4) to decode the most significant value bit m_b of the arc b=HN1, and the type, precision and value for the arc are set to $t_b=3$, $P_b=2$ and $V_b=m_b$.

Next, if SKIP_TO_ND = 0 and $\sigma_{\mathrm{HN2}}=0$, the decoder performs the following steps for arc HN2:

H3. If SKIP_TO_NH < 2, the MQ decoder is invoked with context label $\kappa_{\mathrm{HN2}}^{\mathrm{tbpt}}$ to decode the SIG symbol; otherwise, the SKIP_TO_NH = 2 and the decoder sets SIG to 1.
H4. If SIG = 1, significance state σ_{HN2} is set to 1 and the following steps are performed:
   a. The cupping values of all cells that depend on σ_{HN2} are updated.

b. The MQ decoder is invoked with context label $\kappa = 9$ (magnitude bit context – see Table 4) to decode the most significant value bit m_b of the arc b=HN2, and the type, precision and value for the arc are set to $t_b = 3$, $P_b = 2$ and $V_b = m_b$.

Next, LIFE$_H$ is decremented, unless it is already 0, and then if SKIP_TO_ND < 2 and $\sigma_{\text{DN1}} = 0$, the decoder performs the following steps for arc DN1:

D1. If SKIP_TO_ND = 0, the MQ decoder is invoked with context label $\kappa_{\text{DN1}}^{\text{tbpt}}$ to decode the SIG symbol; otherwise, SKIP_TO_ND = 1 and the decoder sets SIG to 1.
D2. If SIG = 1, significance state σ_{DN1} is set to 1 and the following steps are performed:
   a. The cupping values of all cells that depend on σ_{DN1} are updated.
   b. The MQ decoder is invoked with context label $\kappa = 9$ (magnitude bit context – see Table 4) to decode the most significant value bit m_b of the arc b=DN1, and the type, precision and value for the arc are set to $t_b = 3$, $P_b = 2$ and $V_b = m_b$.

Next, if $\sigma_{\text{DN2}} = 0$, the decoder performs the following steps for arc DN2:

D3. If SKIP_TO_ND < 2, the MQ decoder is invoked with context label $\kappa_{\text{DN2}}^{\text{tbpt}}$ to decode the SIG symbol; otherwise, the SKIP_TO_ND = 2 and the decoder sets SIG to 1.
D4. If SIG = 1, significance state σ_{DN2} is set to 1 and the following steps are performed:
   c. The cupping values of all cells that depend on σ_{DN2} are updated.
   d. The MQ decoder is invoked with context label $\kappa = 9$ (magnitude bit context – see Table 4) to decode the most significant value bit m_b of the arc b=DN2, and the type, precision and value for the arc are set to $t_b = 3$, $P_b = 2$ and $V_b = m_b$.

9.5.4 Root significance decoding for TriBPT CL code-blocks

The root significance decoding pass for bit-plane p within a TriBPT CL band code-block processes 4-span groups in raster order, as shown in Figure 26 and Figure 27. For each 4-span group, the decoder enters a special "RUN mode" if the 4-span group has an entirely insignificant context, meaning that all cupping values are 0 and all root arcs within the group are also insignificant. This is exactly the same condition described for non-root significance decoding in 9.5.3. When this condition is satisfied, the MQ decoder is invoked with context label $\kappa = 6$ (root significance RUN context – see Table 4) to decode an end-of-run symbol.

If the end-of-run symbol is 0, the entire group remains insignificant and decoding proceeds to the next 4-span group, if any. If the end-of-run symbol is 1, then at least one of the 6 root arcs in the 4-span group is significant. The MQ decoder is then used with context label 26 (UNIFORM context – see Table 4) to decode either 2 or 3 symbols, so as to determine values SKIP_TO_TOP and SKIP_TO_BOTTOM in accordance with the following steps:

1. The MQ decoder is invoked with context label $\kappa = 26$ to obtain symbol Ra
2. The MQ decoder is then invoked with $\kappa = 26$ to obtain symbol Rb
3. If Rb = 0, the MQ decoder is invoked again with $\kappa = 26$ to obtain symbol Rc; otherwise, Rc=0
4. If Ra = 1, SKIP_TO_TOP is set to $3 - 2 \cdot \text{Rb} - \text{Rc}$ and SKIP_TO_BOTTOM is set to 0
5. Otherwise, Ra = 0 and SKIP_TO_TOP is set to 3 and SKIP_TO_BOTTOM is set to $3 - 2 \cdot \text{Rb} - \text{Rc}$.

If the 4-span group does not have an entirely insignificant context, SKIP_TO_TOP and SKIP_TO_BOTTOM are both set to 0. In this case, and in the case where an end-of-run symbol of 1 was decoded, the decoder performs decoding steps first for the top root arc complex C_T and then for the bottom root arc complex C_B of the 4-span group. In the following description of this process, the root arcs of complex C_T are identified using the labels V_T, H_T and D_T, as defined in 9.5.2, while the root arcs of complex C_B are identified using the labels V_B, H_B and D_B. This relabelling of the arcs keeps the description of the decoding process independent of whether the breakpoint configuration is TriLR or TriRL.

If SKIP_TO_TOP < 2 and $\sigma_{V_T} = 0$, the decoder performs the following decoding steps for arc $b = V_T$:

V1. Context labels $\kappa_{C_T}^{\text{tbpt}}$ and $\kappa_{\text{ibT}}^{\text{tbpt}}$ are evaluated from NBR$_T$ and VTX$_T$, as described in 9.5.2.
V2. If SKIP_TO_TOP = 0, the MQ decoder is invoked with context label $\kappa_{C_T}^{\text{tbpt}}$ to decode the SIG symbol; otherwise, SKIP_TO_TOP = 1 and the decoder sets SIG to 1.
V3. If SIG = 1, the MQ decoder is invoked with context label $\kappa_{\text{ibT}}^{\text{tbpt}}$ to decode symbol IBLOCK.
V4. If SIG = 1 and IBLOCK = 1, the decoder sets $\sigma_b = 1$, $t_b = 0$, $P_b = 1$ (induction block) and $\tau_b = 0$.
V5. If SIG = 1 and IBLOCK = 0, VTX$_T$ is incremented and then the MQ decoder is invoked with context label 9 (magnitude bit context – see Table 4) to decode symbol m_b, setting $\sigma_b = 1$, $t_b = 3$, $P_b = 2$ and $V_b = m_b$.

If SKIP_TO_TOP < 3 and $\sigma_{H_T} = 0$, the decoder performs the following decoding steps for arc $b = H_T$:

H1. Context labels $\kappa_{C_T}^{\text{tbpt}}$ and $\kappa_{\text{ibT}}^{\text{tbpt}}$ are evaluated from NBR$_T$ and VTX$_T$, as described in 9.5.2.

H2. If SKIP_TO_TOP < 2, the MQ decoder is invoked with context label $\kappa_{C_T}^{\text{tbpt}}$ to decode the SIG symbol; otherwise, SKIP_TO_TOP = 2 and the decoder sets SIG to 1.
H3. If SIG = 1, the MQ decoder is invoked with context label $\kappa_{\text{ibT}}^{\text{tbpt}}$ to decode symbol IBLOCK.
H4. If SIG = 1 and IBLOCK = 1, the decoder sets $\sigma_b = 1$, $t_b = 0$, $P_b = 1$ (induction block) and $\tau_b = 0$.
H5. If SIG = 1 and IBLOCK = 0, VTX_T is incremented and then the MQ decoder is invoked with context label 9 (magnitude bit context – see Table 4) to decode symbol m_b, setting $\sigma_b = 1$, $t_b = 3$, $P_b = 2$ and $V_b = m_b$.

If SKIP_TO_BOTTOM = 0 and $\sigma_{D_T} = 0$, the decoder performs the following decoding steps for arc $b = D_T$:

D1. Context labels $\kappa_{C_T}^{\text{tbpt}}$ and $\kappa_{\text{ibT}}^{\text{tbpt}}$ are evaluated from $\text{NBR}_\text{T}$and VTX_T, as described in 9.5.2.
D2. If SKIP_TO_TOP < 3, the MQ decoder is invoked with context label $\kappa_{C_T}^{\text{tbpt}}$ to decode the SIG symbol; otherwise, SKIP_TO_TOP = 3 and the decoder sets SIG to 1.
D3. If SIG = 1, the MQ decoder is invoked with context label $\kappa_{\text{ibT}}^{\text{tbpt}}$ to decode symbol IBLOCK.
D4. If SIG = 1 and IBLOCK = 1, the decoder sets $\sigma_b = 1$, $t_b = 0$, $P_b = 1$ (induction block) and $\tau_b = 0$.
D5. If SIG = 1 and IBLOCK = 0, VTX_T is incremented and then the MQ decoder is invoked with context label 9 (magnitude bit context – see Table 4) to decode symbol m_b, setting $\sigma_b = 1$, $t_b = 3$, $P_b = 2$ and $V_b = m_b$.

If SKIP_TO_BOTTOM < 2 and $\sigma_{V_B} = 0$, the decoder performs the following decoding steps for arc $b = V_B$:

V6. Context labels $\kappa_{C_B}^{\text{tbpt}}$ and $\kappa_{\text{ibB}}^{\text{tbpt}}$ are evaluated from NBR_B and VTX_B, as described in 9.5.2.
V7. If SKIP_TO_BOTTOM = 0, the MQ decoder is invoked with context label $\kappa_{C_B}^{\text{tbpt}}$ to decode the SIG symbol; otherwise, SKIP_TO_BOTTOM = 1 and the decoder sets SIG to 1.
V8. If SIG = 1, the MQ decoder is invoked with context label $\kappa_{\text{ibB}}^{\text{tbpt}}$ to decode symbol IBLOCK.
V9. If SIG = 1 and IBLOCK = 1, the decoder sets $\sigma_b = 1$, $t_b = 0$, $P_b = 1$ (induction block) and $\tau_b = 0$.
V10. If SIG = 1 and IBLOCK = 0, VTX_B is incremented and then the MQ decoder is invoked with context label 9 (magnitude bit context – see Table 4) to decode symbol m_b, setting $\sigma_b = 1$, $t_b = 3$, $P_b = 2$ and $V_b = m_b$.

If SKIP_TO_BOTTOM < 3 and $\sigma_{H_B} = 0$, the decoder performs the following decoding steps for arc $b = H_B$:

H6. Context labels $\kappa_{C_B}^{\text{tbpt}}$ and $\kappa_{\text{ibB}}^{\text{tbpt}}$ are evaluated from NBR_B and VTX_B, as described in 9.5.2.
H7. If SKIP_TO_BOTTOM < 2, the MQ decoder is invoked with context label $\kappa_{C_B}^{\text{tbpt}}$ to decode the SIG symbol; otherwise, SKIP_TO_BOTTOM = 2 and the decoder sets SIG to 1.
H8. If SIG = 1, the MQ decoder is invoked with context label $\kappa_{\text{ibB}}^{\text{tbpt}}$ to decode symbol IBLOCK.
H9. If SIG = 1 and IBLOCK = 1, the decoder sets $\sigma_b = 1$, $t_b = 0$, $P_b = 1$ (induction block) and $\tau_b = 0$.
H10. If SIG = 1 and IBLOCK = 0, VTX_B is incremented and then the MQ decoder is invoked with context label 9 (magnitude bit context – see Table 4) to decode symbol m_b, setting $\sigma_b = 1$, $t_b = 3$, $P_b = 2$ and $V_b = m_b$.

If $\sigma_{D_B} = 0$, the decoder performs the following decoding steps for arc $b = D_B$:

D6. Context labels $\kappa_{C_B}^{\text{tbpt}}$ and $\kappa_{\text{ibB}}^{\text{tbpt}}$ are evaluated from NBR_B and VTX_B, as described in 9.5.2.
D7. If SKIP_TO_BOTTOM < 3, the MQ decoder is invoked with context label $\kappa_{C_B}^{\text{tbpt}}$ to decode the SIG symbol; otherwise, SKIP_TO_BOTTOM = 3 and the decoder sets SIG to 1.
D8. If SIG = 1, the MQ decoder is invoked with context label $\kappa_{\text{ibB}}^{\text{tbpt}}$ to decode symbol IBLOCK.
D9. If SIG = 1 and IBLOCK = 1, the decoder sets $\sigma_b = 1$, $t_b = 0$, $P_b = 1$ (induction block) and $\tau_b = 0$.
D10. If SIG = 1 and IBLOCK = 0, VTX_B is incremented and then the MQ decoder is invoked with context label 9 (magnitude bit context – see Table 4) to decode symbol m_b, setting $\sigma_b = 1$, $t_b = 3$, $P_b = 2$ and $V_b = m_b$.

9.5.5 Root significance decoding for TriBPT LL code-blocks

The significance decoding pass for bit-plane p within a TriBPT LL band code-block processes 2x2 cells in raster order. In the canonical scan, rows of cells are visited from top to bottom, cells of each row are visited from left to right, and the three arcs b of each cell are visited in the order V (vertical), H (horizontal) then D (diagonal).

For each arc b, following this scanning order, if $\sigma_b = 0$, the MQ decoder is invoked with context label κ=1 (significance flag context – see Table 5) to decode symbol SIG. If SIG = 0, arc b remains insignificant. Otherwise, the following steps are performed:

1. The significance state for arc b is changed to $\sigma_b = 1$
2. The MQ decoder is invoked with context label $\kappa = 5$ (vertex flag context – see Table 5) to decode symbol VTX.
3. If VTX = 1, the breakpoint type for arc b is set to $t_b = 3$. Otherwise, the MQ decoder is invoked with context label $\kappa = 8$ (direct flag context – see Table 5) to decode symbol DIRECT, and the type for arc b is set to $t_b = 1 + \text{DIRECT}$.

4. If VTX = 0, the MQ decoder is invoked with context label $\kappa = 6$ (degenerate flag context – see Table 5) to decode symbol DEGEN; otherwise DEGEN is set to 0.
5. If DEGEN = 1, the following steps are performed:
    a. The MQ decoder is invoked with context label $\kappa = 26$ (uniform context – see Table 5) to decode symbol AMBIVALENT.
    b. The precision and value parameters for arc b are set to P_b = AMBIVALENT and $V_b = 0$ and the refinement counter is set to $\tau_b = 0$ (not refinable).
6. Otherwise, DEGEN = 0 and the MQ decoder is invoked with context label $\kappa = 9$ (magnitude bit context) to decode the most significant value bit m_b of arc b. Then, the precision and value for the arc are set to $t_b = 3$, $P_b = 2$ and $V_b = m_b$.
7. If VTX = 0 and DEGEN = 0, the MQ decoder is invoked twice with context label $\kappa = 26$ (uniform context – see Table 5) to decode symbol E_A and then E_B, and the extrapolation qualifier for arc b is set to $e_b = 2 \cdot E_A + E_B$.

NOTE 1 – Coding procedures for LL band code-blocks are intended to be very simple, since in most cases all LL code-blocks are expected to be entirely insignificant. Coding tools are provided for the LL band of a breakpoint component primarily to allow the communication of breaks when N_L is heavily constrained, or even 0. For example, the breakpoints described by the breakpoint media format suggested in Annex B can be encapsulated directly within the LL band of a breakpoint component with $N_L = 0$ decomposition levels.

NOTE 2 – No means is provided to identify whether or not an arc without a break might hold an induction block, whereas this information can be recorded (as a hint) within the breakpoint media format suggested in Annex B. Induction blocks play no role in breakpoint dependent transforms and so there is no need to record them within resolution $r = 0$ of a breakpoint component.

9.5.6 Position refinement decoding for TriBPT code-blocks

Position refinement decoding is the same for CL and LL band code-blocks of a TriBPT component, using MQ coder context labels 9 through 25 from Table 4 and Table 5, and processing 2×2 cells in raster order. In the canonical scan, rows of cells are visited from top to bottom, cells of each row are visited from left to right, and the three arcs b of each cell are visited in the order V (vertical), H (horizontal) then D (diagonal).

For each arc b, following this scanning order, if $\sigma_b = 1$ and $\tau_b > 0$, the following steps are performed:

1. The MQ decoder is invoked with context label $\kappa = 10 + \min\{(P_b - 2), 15\}$ to decode symbol TERM.
2. If TERM = 1, the decoder sets $\tau_b = 0$
3. Otherwise, TERM = 0 and the MQ decoder is invoked with context label $\kappa = 9$ (magnitude refinement context) to decode refinement bit r_b, after which the decoder the arc's state variables as follows: $P_b \coloneqq P_b + 1$; $V_b \coloneqq 2V_b + r_b$ and $\tau_b \coloneqq \tau_b - 1$.

9.6 Quality layers and packets for breakpoint components

As with other JPEG 2000 components, the coded information for a breakpoint tile-component is distributed across one or more layers in the codestream, each layer consisting of zero or more consecutive coding passes from each code-block in the tile. The number of coding passes in the layer can vary from code-block to code-block and the number of layers for the tile-component is signalled in the COD marker segment, exactly as for other JPEG 2000 components. Each layer corresponds to one packet of each precinct in each resolution of the tile-component.

The packets of a breakpoint component have the same structure as packets of other components, as described in Rec. ITU-T T.800 | ISO/IEC 15444-1, with the same packet header syntax, except that additional signalling is introduced to identify the point at which a code-block's representation can be considered *complete*. Completeness is an important concept for breakpoint code-blocks. As mentioned in 9.1, the maximum bit-plane index for a code-block is

$$p_{max} = M_b - 1 - P,$$

where P is the number of missing most significant bit-planes that is signalled via the packet header of the first packet to which the code-block contributes, and the maximum number of coding passes for the code-block is then

$$Z_{max} = 3p_{max} + 2.$$

However, not all of these coding passes need appear within packet headers. Certainly, the code-block's representation is complete if all of its Z_{max} coding passes are found within codestream packets. A transcoder or other intermediate agent may truncate the representation, discarding packets so that fewer than Z_{max} coding passes are available for decoding. In this case, the representation is incomplete and in the inter-band coding mode (BPT_INTER = 1) this lack of completeness can prevent a decoder from processing some coding passes of other code-blocks that depend upon the information that has been removed. Alternatively, though, an encoder can choose to produce only a subset of the possible Z_{max} coding passes, or indeed it can choose to produce no data at all for some code-block. This is fundamentally different from the removal of content by transcoding, because a decoder can still recover all originally encoded content, allowing it to be sure that all information required for decoding other code-blocks in the inter-band coding mode is actually available. To address the difference between these two ways in which the number of coding passes available for a code-block can be less than Z_{max}, the packet header syntax for breakpoint components is augmented to support the explicit signalling of code-block completeness.

Two specific types of completeness are identified as *zero-completeness* and *pass-completeness*. A code-block is said to be *zero-complete* if it has no coding passes at all, but is nonetheless complete. This is a very common and important condition. A code-block for which no coding passes have yet been communicated via packet headers might have important information in later (possibly discarded) packets that is required to support the decoding of other code-blocks in the inter-band coding mode. However, a code-block which is zero-complete will never contribute any information and any other code-block that depends on the zero-complete code-block can assume that $\sigma_{b'}(p) = 0$ for all bit-planes p and all arcs b' belonging to the zero-complete code-block.

Zero-completeness for all code-blocks of a breakpoint precinct is signalled within the precinct's very first packet header, using an augmented version of the "code-block inclusion" signalling procedure. If the first packet for the precinct is a zero length packet, as defined in Rec. ITU-T T.800 | ISO/IEC 15444-1 (i.e., if the first bit in the packet header is 0), then zero-completeness is not coded in any packet header for the precinct and no code-block of the precinct is zero-complete.

In the very first packet of a breakpoint component's precinct, the code-block inclusion signalling procedure is augmented with information that signals *all-completeness* and *any-completeness* values for each node in the precinct's tag tree. During the original code-block inclusion parsing process for layer 0, the nodes of the tag tree are traversed in the manner described in Rec. ITU-T T.800 | ISO/IEC 15444-1 to determine answers to the question "do any code-blocks covered by this node contribute coding passes to layer 0?" For each node visited within the tag tree scan, the original code-block inclusion parsing process decodes one bit i, such that the answer to the question is "yes" if $i = 1$ and "no" if $i = 0$.

For breakpoint components, this process is augmented to simultaneously answer the following two questions: 1) "are all code-blocks covered by this node zero-complete?" and 2) "is any code-block covered by this node zero-complete?" These three questions are answered by first decoding a bit z. If the value of $z = 1$ then the answers are "no", "yes" and "yes" – that is, all code-blocks covered by the node are zero-complete and hence no code-block covered by the node makes any contribution to layer 0. If the value of $z = 0$, the usual inclusion bit i is decoded to answer the first question ("do any code-blocks contribute?") and a second bit n is then decoded to answer the third question ("is any code-block zero-complete?"). If the node is a leaf node, meaning that it represents only one code-block, then the n bit is not decoded, because the answer to the second and third questions must be the same ($n = z$). If a parent node has already answered the first question with "no", then the i bit is not decoded because the answer $i = 0$ is known. If a parent node has already answered the second question with "yes", then no bits are decoded because the answers to all questions are known ($i = 0$, $z = 1$ and $n = 1$). If a parent node has already answered the third question with "no", then no z or n bit is decoded since the answers to the second and third questions are known ($z = n = 0$).

A code-block is said to be *pass-complete* if at least one coding pass of the code-block appears within codestream packets and the last such coding pass has been signalled as completing the representation. A pass-complete code-block cannot be zero-complete, because it contributes at least one coding pass. Pass completeness is signalled via a single bit c, that is decoded immediately after the "code-block inclusion information" for a code-block that has one or more included passes. Equivalently, bit c is decoded immediately before the "zero bit-plane information" for the code-block, if present, or else immediately before the "number of coding passes information" for the code-block, as described in Rec. ITU-T T.800 | ISO/IEC 15444-1. If the value of bit c is 1, then the code-block is pass-complete.

Code-blocks that have been identified as pass-complete or zero-complete do not contribute "code-block inclusion" information to subsequent packets of their precinct.

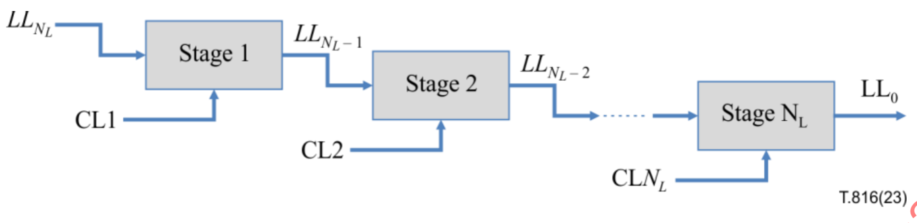
There are only two types of codeword segments for breakpoint code-blocks. Multiple codeword segments arise when termination of the MQ arithmetic coder occurs between coding passes which are included in a packet. However, the selective arithmetic coding bypass mode has no meaning within breakpoint components, so that termination of the MQ coder occurs either at the end of every coding pass or only at the end of the code-block. Thus, the two types of codeword segments that can arise for a breakpoint component are: a) one segment per code-block; and b) one segment per coding pass.

Apart from selective arithmetic coding bypass and vertically causal contexts, all other block coding modes signalled via the code-block style information within SPcod and SPcoc fields of COD and COC marker segments apply to breakpoint components. Specifically, context probabilities may be reset on coding pass boundaries, the arithmetic coder may be terminated on each coding pass, the use of predictable arithmetic codeword termination may be identified, and segmentation symbols may be used. These all have the same meaning that they have for other JPEG 2000 component types, as specified in Rec. ITU-T T.800 | ISO/IEC 15444-1.

10 Reconstruction of breakpoint components

10.1 Overview

This clause describes the synthesis of breakpoint values for each arc of each resolution in a breakpoint tile-component, using vertex and induction-block values recovered by decoding all relevant breakpoint code-blocks. Similar to the inverse wavelet transform, synthesis of a breakpoint tile-component proceeds in stages, from low to high resolution. Each stage combines breakpoint values from an LL band with a CL band, to form an intermediate LL band at the next higher resolution. This is illustrated in Figure 31.



CL1	CL band at resolution $r = 1$
CL2	CL band at resolution $r = 2$
CLN_L	CL band at resolution $r = N_L$
LL_{N_L}	input LL band at resolution $r = 0$
LL_{N_L-1}	intermediate LL band at resolution $r = 1$
LL_{N_L-2}	intermediate LL band at resolution $r = 2$
LL_0	output LL band

Figure 31 – Breakpoint synthesis stages

There is one synthesis stage for each non-base resolution $r = 1,2,\dots,N_L$. The input to stage r consists of breakpoint values from an "LL band" at resolution $r-1$ and a CL band at resolution r, where these values consist of the breakpoint type t, precision P, and tick-point k, as introduced in 7.5. Input values from the CL band can only have types $t = 3$ (vertex) or $t = 0$ (no signalled break, or induction block). Output values from synthesis stage r preserve decoded information from the CL band at resolution r, but add induced breaks, having types $t = 2$ (direct induction) and $t = 1$ (spatial induction).

Synthesis stage r involves three steps. In the first step, each arc breakpoint from the CL band at resolution r is transferred to the synthesis stage's output, whose rectangular region of support corresponds exactly to that of resolution r, as given by Formula (3). This transfer step discards any CL band breakpoints that do not exist in the output, because the grid-point at the centre of the arc lies outside the region of support for resolution r. This can happen, because the CL band includes a whole number of 2×2 cells, some of which might only partially intersect the resolution's region of support.

In the second step, LL breakpoints at resolution $r-1$ are used to directly induce breaks at resolution r, merging these into the synthesis stage's output, wherever they do not conflict with breaks transferred in the first step. This step is discussed further in 10.2.

In the third step, the resolution r arc breakpoints produced by the first two steps are used to spatially induce additional breaks at resolution r. This is the spatial induction step and is the one that is most involved. This step is the subject of 10.3 and 10.4.

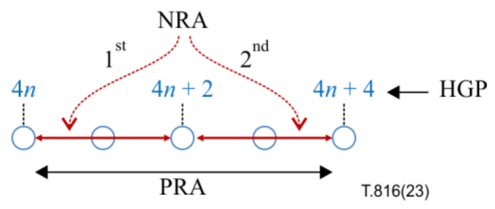
10.2 Direct induction step

10.2.1 Introduction

For each non-root arc within the support of resolution r, there is almost always a corresponding "parent" arc breakpoint that exists within the support of resolution $r-1$. The parent arc breakpoint, if it exists, belongs to the LL_{N_L+r-1} band that enters synthesis stage r, and it has arc length 2^{d+1} when expressed at the full resolution of the tile-component, where $d = N_L + 2 - r$.

NOTE – As defined in 7.4, non-root arcs necessarily have a corresponding parent arc in the next lower resolution, such that the two arcs are parallel and share an end-point. However, that parent arc's breakpoint, corresponding to the grid-point at its centre, might lie outside the rectangular region associated with resolution $r-1$, as given by Formula (3), even though the non-root arc's breakpoint lies within the rectangular region for resolution r.

The non-root arc has arc length 2^d when expressed at the full resolution of the tile-component. Both arcs are co-linear and share one end-point, when projected onto the full resolution of the tile-component. In particular, the non-root arc coincides either with the first or the second half of its parent arc. This relationship is illustrated in Figure 32 for the specific case of horizontal arcs, but the same principals apply to vertical and diagonal arcs.



NRA	non-root arc at resolution r
PRA	parent arc at resolution r−1
HGP	horizontal grid-point ordinates at resolution r

Figure 32 – Relationship between non-root and parent arcs during direct induction in synthesis stage r, shown for the case of horizontal arcs

If the parent arc exists within resolution $r - 1$ and has a break that lies within the span of the non-root arc at resolution r, and the non-root arc has no break of its own in the CL band at resolution r, a directly induced break is added to the non-root arc. The directly induced break is assigned breakpoint type $t_{\text{induced}} = 2$, with tick-point k_{induced} and precision P_{induced} obtained from those of the parent arc according to Formula (9).

If $P_{\text{induced}} = 1$ then the directly induced break that is added to the non-root arc is identified as an ambivalent break.

If the parent arc at resolution $r - 1$ contains a break with precision $P_{\text{parent}} \leq 1$, then each of the two non-root arcs at resolution r are assigned an indefinite break with $t_{\text{induced}} = 2$ and precision $P_{\text{induced}} = 0$.

10.2.2 Extrapolation Qualifier for TriBPT direct induction

For the TriBPT arrangement, each directly induced breakpoint at a resolution r is associated with extrapolation qualifiers comprised of a pair of binary flags GRAD1_FLAG$_{induced}$ and GRAD2_FLAG$_{induced}$ which are utilized during the breakpoint-dependent transform stage. As explained in 7.5, the two bit extrapolation qualifier e_b is equated to these flags, such that $e_{b,0}$ = GRAD1_FLAG $_{induced}$ and $e_{b,1}$ = GRAD2_FLAG $_{induced}$. The process of determining the binary flags for each directly induced break is defined below.

Starting with a vertex at resolution $r - 1$, flags GRAD1_FLAG $_{parent}$ and GRAD2_FLAG $_{parent}$ are initialized to 0 (FALSE). The flag values GRAD1_FLAG $_{induced}$ and GRAD2_FLAG $_{induced}$ associated with the corresponding directly induced break at the finer resolution level r is determined in accordance with the following procedure.

For the parent arc, if the breakpoint is located in the second half of the arc at resolution $r - 1$ (i.e., right or bottom half of the parent arc) then

GRAD1_FLAG $_{induced}$ = 1 (TRUE)

GRAD2_FLAG $_{induced}$ = GRAD2_FLAG $_{parent}$

Otherwise, if the breakpoint on the parent arc is located in the first half of the arc at resolution $r - 1$ (i.e., left or top half of the parent arc) then

GRAD1_FLAG $_{induced}$ = GRAD1_FLAG $_{parent}$

GRAD2_FLAG $_{induced}$ = 1 (TRUE).

As direct induction proceeds to subsequent finer resolutions, the flags are determined at each resolution following the same procedure as that described above with the current *induced* arc at resolution r becoming the *parent* arc of the subsequent *induced* arc at the finer level $r + 1$.

10.3 QuadBPT spatial induction

10.3.1 Introduction

Spatial induction applies only to root arcs. As shown in Figure 3, for the QuadBPT arrangement, each intersecting pair of horizontal and vertical root arcs lies within a 2-span, surrounded on all sides by non-root arcs. QuadBPT spatial induction is based on breaks found on these surrounding non-root arcs alone. Breaks on the surrounding non-root arcs can correspond to decoded vertices at resolution r, direct induction at resolution r, or spatial induction at resolution $r-1$. It can happen that some (or even all) of these surrounding non-root arcs do not actually exist within the rectangular support of resolution r, as given by Formula (3); any such non-root arc that does not exist is taken to have no break (i.e., $t=0$ and $P=0$) for the purpose of spatial induction.

It is useful to label the non-root arcs L (left), R (right), T (top) and B (bottom), designating them as perimeter arcs. It is also helpful to label the vertical and horizontal root arcs V and H, respectively. All of these arcs have length 2^d when assessed at the top resolution of the tile-component, and any break on arc b is located on one of the tick-points identified in Formula (6), characterized by a tick-point index k_b in the range $0 \leq k_b < 2^{d+F_B}$. The breakpoint parameters for all quantities involved in spatial induction are denoted (t_L, P_L, k_L), (t_R, P_R, k_R), (t_T, P_T, k_T), (t_B, P_B, k_B), (t_V, P_V, k_V) and (t_H, P_H, k_H), where the arc-identifying subscripts here correspond to the perimeter labels introduced above.

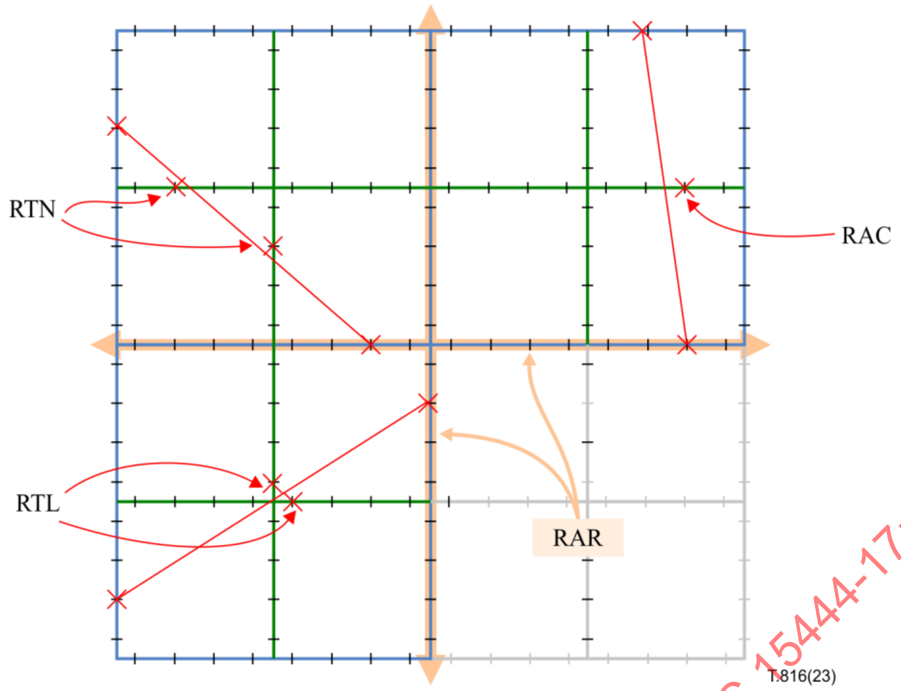
Spatial induction to the V and H root arcs shall take place only if the following two conditions are both satisfied:

1. Exactly 2 of the four perimeter arcs have breaks – i.e., two of t_L, t_R, t_T and t_B are non-zero.
2. Neither of the root arcs has its own vertex or induction block; equivalently, $P_V = 0$ and $P_H = 0$ prior to spatial induction.

Subject to the above conditions, labelling the two perimeter arcs with breaks as B1 and B2, if either of these is indefinite (i.e., if $P_{B1} \cdot P_{B2} = 0$), then both root arcs are assigned indefinite breaks – i.e., $t_H = t_V = 1$ and $P_H = P_V = 0$. If either of the breaks B1 or B2 is ambivalent, that is $\min\{P_{B1}, P_{B2}\} = 1$, then both root arcs are assigned ambivalent breaks – i.e., $t_H = t_V = 1$ and $P_H = P_V = 1$. Otherwise, spatial induction is performed by generating a line segment between the tick-point locations associated with the perimeter breaks, as given by k_{B1} and k_{B2}, and finding its intersection, if any, with each of the V and H root arcs, rounding the resulting location to the nearest tick point. Writing R for the generic label of a root arc V or H, if an intersection exists with root arc R, it is assigned break-type $t_R = 1$, break-precision $P_R = \min\{P_{B1}, P_{B2}\}$ and tick-point index $k_R = f_{\text{quad}}(\text{B1,B2,R}, k_{B1}, k_{B2})$, where f_{quad} is the QuadBPT intersection evaluation function, that depends on the locations and geometry of the three arcs B1, B2 and R, along with the perimeter break tick-points k_{B1} and k_{B2}.

The function f_{quad} rounds the intersection between the line segment induced by the two perimeter breaks and the root arc R to the nearest available tick-point k_R. If the line segment induced by the two perimeter breaks is equidistant from two tick points, the induced break location is obtained by rounding away from the centre of the root arc. This is consistent with the way in which decoded vertices are mapped to breakpoints, as given by Formula (8).

It remains to resolve the rounding ambiguity that arises when the line segment between perimeter breaks passes exactly through the centres of the two root arcs. This is only possible when inducing breaks from two horizontal perimeter breaks or from two vertical perimeter breaks that are located on opposite sides of their arc centres. To ensure that the spatially induced breakpoints have absolute geometric meaning, that can be preserved under horizontal or vertical geometric flips, transposition and rotation by multiples of 90°, this ambiguity is resolved by rounding towards the nearest root arcs at the next lower resolution, regardless of whether or not they actually exist. Equivalently, this last case is resolved by rounding towards the centre of the 4-span to which the V and H root arcs belong. These rounding policies are illustrated in Figure 33.



RTN	Round to nearest
RAC	Round away from centre
RTL	Round towards lower resolution root arcs
RAR	Root arcs at resolution r−1

Figure 33 – Rounding policies used in QuadBPT spatial induction, showing different induction examples in three of the four 2-spans within a 4-span

10.3.2 Image boundary handling for QuadBPT

For the QuadBPT case, a set of implied breaks are assumed for arcs that intersect the right and bottom image boundaries. As described in 7.1, the rectangular region of image samples, is defined by (x_o, y_o) for the upper left hand corner and $(x_l - 1, y_l - 1)$ for the lower right hand corner. Implied breaks are placed on arcs that intersect the right vertical boundary at $x_l - 1$ or the bottom horizontal boundary at $y_l - 1$.

Possible break positions for implied breaks are denoted by $VrtBndry\_breakposition$ and $HrzBndry\_breakposition$ corresponding to the right-vertical boundary and the bottom-horizontal boundary respectively, and are specified below.

$$VrtBndry\_breakposition = (x_l - 1) + 2^{-(F_B+1)}$$

$$HrzBndry\_breakposition = (y_l - 1) + 2^{-(F_B+1)}$$

NOTE 1 – These implied breaks are never explicitly communicated but assumed to exist and utilized to adapt the transform for arcs that intersect the right and bottom image boundaries.

NOTE 2 – The implied break positions $VrtBndry\_breakposition$ and $HrzBndry\_breakposition$ are located outside the image region at one tick point away from the image boundary intersect.

As an initialization step, for all resolution levels r, an arc centred at location $\boldsymbol{p}$ that crosses the right or bottom image boundary is set to type $t_b(\boldsymbol{p}) = 0$.

The set of implied breaks include implied vertices on arcs at the coarsest resolution level $r = N_L$, located at positions $VrtBndry\_breakposition$ or $HrzBndry\_breakposition$, depending on the image boundary that an arc intersects.

At finer resolution levels $r < N_L$, breaks on non-root arcs are determined by direct induction.

Root arcs are subject to the spatial induction procedure. For a vertical root arc that intersects the horizontal image boundary, if spatial induction results in a break that has a location that is different to $HrzBndry\_breakposition$ then an implied vertex is placed on the horizontal root arc of the 2-span that intersect the image boundary. Similarly, for a horizontal root arc that intersects the vertical image boundary, if spatial induction results in a break that has a location that is different to $VrtBndry\_breakposition$ then implied vertices are placed on root arcs of the 2-span that intersect the image boundaries.

NOTE 3 – spatial induction, induces breaks onto root arcs. If an induced break does not correctly model the image boundary then spatial induction is abandoned and implied vertices are placed on root arcs that intersect the image boundary. The need for such implied vertices on root arcs is most obvious at 2-span arrangements that contain the bottom right corner of the image region.

10.4 TriBPT spatial induction

10.4.1 TriBPT induction modes and corner notation

Spatial induction applies only to root arcs and Figure 3 shows the three root arcs for the TriBPT-LR and TriBPT-RL configurations within each triangle of a 4-span. At a given resolution r, non-root arcs are sub-arcs of arcs at resolution $r-1$ while root arcs are new arcs that emerge at the current resolution (i.e they have no parent arcs at resolution $r-1$). Spatial induction for TriBPT components can be categorized into 4-arc and 3-arc induction modes. The 4-arc mode involves two non-root source arcs, whose breaks induce new breaks on two root arcs. The 3-arc mode involves one non-root source arc, with one root source arc, whose breaks induce a break on one root arc. Both modes are specified below after introducing important notation. In all cases, source breaks on non-root arcs correspond to vertices decoded at resolution r or breaks that are directly induced from breaks at resolution $r-1$, which can have been vertices, directly induced or spatially induced breaks. Where an induction source break belongs to a root arc (3-arc mode), it corresponds to a decoded vertex at resolution r.

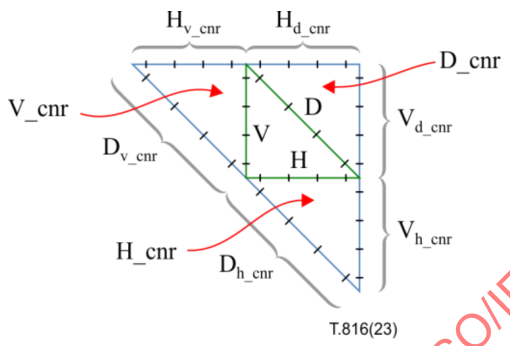


Figure 34 – Arc and corner labels for the TriBPT-LR arrangement

As shown in Figure 34, root arcs are labelled V (vertical), H (horizontal) and D (diagonal). In the 4-span arrangement, the non-root horizontal and diagonal arcs that form the corner adjacent to V are labelled $\mathrm{H_{v\_cnr}}$ and $\mathrm{D_{v\_cnr}}$ respectively. The vertical and diagonal non-root arcs that form the corner adjacent to H are labelled $\mathrm{V_{h\_cnr}}$ and $\mathrm{D_{h\_cnr}}$, respectively, while the horizontal and vertical arcs that form the corner adjacent to D are labelled $\mathrm{H_{d\_cnr}}$ and $\mathrm{V_{d\_cnr}}$. All of these arcs have length 2^d when assessed at the top resolution of the tile-component, and all break positions lie on tick-points in accordance with Formula (6), identified by tick-point indices k_b in the range $0 \le k_b < 2^{d+F_B}$. The breakpoint parameters for all quantities involved in spatial induction are denoted (t_b, P_b, k_b), where the relevant arcs b have the labels $b \in \{\mathrm{V}, \mathrm{H_{v\_cnr}}, \mathrm{D_{v\_cnr}}, \mathrm{H}, \mathrm{V_{h\_cnr}}, \mathrm{D_{h\_cnr}}, \mathrm{D}, \mathrm{H_{d\_cnr}}, \mathrm{V_{d\_cnr}}\}$.

The term $N_{\mathrm{V\_cnr}}$ denotes a count of the number of breaks that are present in the vertical corner's non-root arcs $\mathrm{H_{v\_cnr}}$ and $\mathrm{D_{v\_cnr}}$. Similarly, $N_{\mathrm{H\_cnr}}$ counts the number of breaks present on the horizontal corner's non-root arcs $\mathrm{V_{h\_cnr}}$ and $\mathrm{D_{h\_cnr}}$, and $N_{\mathrm{D\_cnr}}$ counts the number of breaks present on the diagonal corner's non-root arcs $\mathrm{H_{d\_cnr}}$ and $\mathrm{V_{d\_cnr}}$. These counts all lie in the range 0 to 2.

The notation $S_{\mathrm{X\_cnr}}$ is used to label a corner X_cnr, where $X \in \{\mathrm{V, H, D}\}$, as being either a candidate source corner ($S_{\mathrm{X\_cnr}} = 1$) or a non-candidate ($S_{\mathrm{X\_cnr}} = 0$) corner for spatial induction. If $N_{\mathrm{X\_cnr}} = 1$ and the root arc X does not have a vertex or an induction block (i.e., $P_X = 0$) then $S_{\mathrm{X\_cnr}} = 1$; otherwise $S_{\mathrm{X\_cnr}} = 0$. A label of $S_{\mathrm{X\_cnr}} = 1$ signifies that the corner X_cnr, contains a single non-root arc with a breakpoint that can potentially be a source for defining the line segment for spatial induction and that this induction is not impeded by a vertex or induction block in the corresponding root arc X.

10.4.2 4-arc mode

The 4-arc mode involves new breaks on two root arcs being induced by breaks specified on two non-root arcs that are located at different corners. This spatial induction to two out of the three root arcs (i.e., two of V, H and D) shall take place only if the following conditions are all satisfied:

1. None of the three root arcs has its own vertex; equivalently, $P_V = 0$, $P_D = 0$, and $P_H = 0$ prior to spatial induction.
2. Two out of the three corners are labelled as being a source corner for spatial induction; that is $S_{\mathrm{H\_cnr}} + S_{\mathrm{V\_cnr}} + S_{\mathrm{D\_cnr}} = 2$.
3. The non-root arcs with breaks that are associated with the two source corners have different orientations – i.e., the source breaks cannot both lie on vertical arcs, both lie on horizontal arcs or both lie on diagonal arcs.

In accordance with the above conditions, let X_cnr and Y_cnr correspond to the two identified source corners, that is $S_{\mathrm{X\_cnr}} = S_{\mathrm{Y\_cnr}} = 1$ where $\mathrm{X, Y} \in \{\mathrm{V, H, D}\}$, $s.t.$ $\mathrm{X \ne Y}$. The corresponding non-root arcs with breaks that originate from these two source corners are labelled B1$_{\text{x\_cnr}}$ and B2$_{\text{y\_cnr}}$. In the following discussions, the subscript is omitted for ease of notation and these two non-root arcs are referred to as B1 and B2.

Spatial induction is performed by generating a line segment between the tick-point locations associated with the non-root arc breaks, as given by k_{B1} and k_{B2}, and finding its intersection with two of the three root arcs and then rounding the resulting location to the nearest tick point. In keeping with the above notation, X and Y refer to the two root arcs on which breaks points are induced. The two induced breakpoint parameters can therefore be denoted as (t_X, P_X, k_X) and (t_Y, P_Y, k_Y). If either of the breaks on non-root arcs B1 and B2 is indefinite (i.e., if $P_{B1} \cdot P_{B2} = 0$), then the corresponding two root arcs are assigned indefinite breaks – i.e., $t_X = t_Y = 1$ and $P_X = P_Y = 0$. If however $\min\{P_{B1}, P_{B2}\} = 1$, then the corresponding two root arcs are assigned ambivalent breaks – i.e., $t_X = t_Y = 1$ and $P_X = P_Y = 1$. Otherwise, the induced breakpoint parameters are assigned as break-type $t_X = t_Y = 1$, break-precision $P_X = P_Y = \min\{P_{B1}, P_{B2}\}$ and tick-point index $k_X = f_{\text{tri}}(\text{B1,B2,X}, k_{B1}, k_{B2})$ and $k_Y = f_{\text{tri}}(\text{B1,B2,Y}, k_{B1}, k_{B2})$ where f_{tri} is the TriBPT intersection evaluation function, that depends on the locations and geometry of the root arc on which a break is being induced, that is either X or Y, and the two non-root arcs B1, B2 along with the corresponding break tick-points k_{B1} and k_{B2}. The function f_{tri} rounds the intersection between the line segment induced by the two non-root arc breaks and the root arc X (Y) to the nearest available tick-point $k_X(k_Y)$.

Examples of all 9 possible non-root arc pair combinations that can result in inducing breaks on two root arcs are shown in Figure 35 for the TriBPT-LR arrangement.

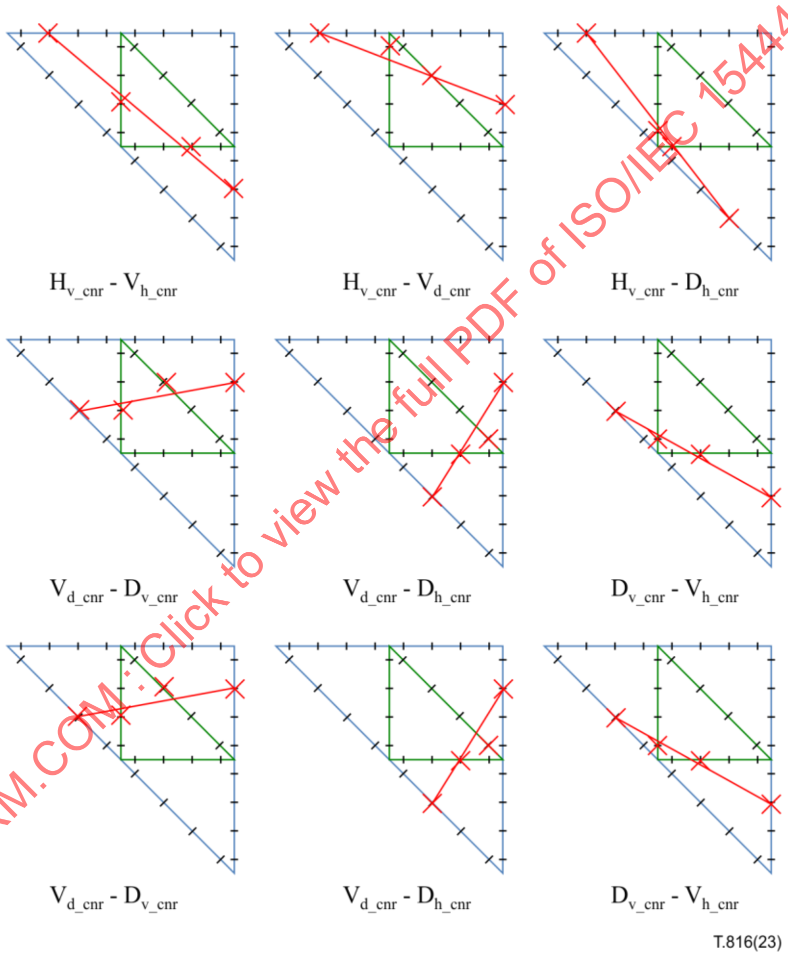

H$_{\text{v\_cnr}}$ - V$_{\text{h\_cnr}}$
H$_{\text{v\_cnr}}$ - V$_{\text{d\_cnr}}$
H$_{\text{v\_cnr}}$ - D$_{\text{h\_cnr}}$
V$_{\text{d\_cnr}}$ - D$_{\text{v\_cnr}}$
V$_{\text{d\_cnr}}$ - D$_{\text{h\_cnr}}$
D$_{\text{v\_cnr}}$ - V$_{\text{h\_cnr}}$
V$_{\text{d\_cnr}}$ - D$_{\text{v\_cnr}}$
V$_{\text{d\_cnr}}$ - D$_{\text{h\_cnr}}$
D$_{\text{v\_cnr}}$ - V$_{\text{h\_cnr}}$
T.816(23)

Figure 35 – 4-arc mode spatial induction examples for the TriBPT-LR arrangement

10.4.3 3-arc mode

The 3-arc mode involves one non-root source arc and one root source arc, whose breaks induce a break on one root arc. This spatial induction to one root arc shall take place only if the following conditions are all satisfied:

1. One of the three root arcs has its own vertex while the remaining two root arcs have no vertex of their own.
2. One out of the three corners are labelled as being a source corner for spatial induction; that is $S_{\text{H\_cnr}} + S_{\text{V\_cnr}} + S_{\text{D\_cnr}} = 1$.

Let R represent the root arc that has its own vertex, with the break parameters given by (t_R, P_R, k_R) where $t_R = 3$. Furthermore let X_cnr correspond to the single identified source corner, that is $S_{\text{X\_cnr}} = 1$, and let B1$_{\text{x\_cnr}}$ correspond to