INTERNATIONAL STANDARD

ISO/IEC 23002-4

Second edition
2014-04-15

AMENDMENT 1
2014-11-15

Information technology — MPEG video technologies —

Part 4:
Video tool library

AMENDMENT 1: Graphics tool library (GTL) for the reconfigurable multimedia coding (RMC) framework

Technologies de l'information — Technologies vidéo MPEG —

Partie 4: Bibliothèque d'outils vidéo

AMENDEMENT 1: Bibliothèque d'outils graphiques (GTL) pour le cadre de codage multimédia reconfigurable (RMC)

Reference number
ISO/IEC 23002-4:2014/Amd.1:2014(E)

© ISO/IEC 2014

Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2014

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Documents
Document Sharing Hub

© ISO/IEC 2014 – All rights reserved

Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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).

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Documents
Document Sharing Hub

© ISO/IEC 2014 – All rights reserved

Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

Documents
Document Sharing Hub

Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

Information technology — MPEG video technologies —

Part 4:
Video Tool Library

AMENDMENT 1: Graphics tool library (GTL) for the reconfigurable multimedia coding (RMC) framework

In 4.1 "FU Interfaces", before Table 1, add the following text:

- In several FU diagrams, the ports are named with a trailing "_i" for the input port type and with a trailing "_o" for the output port type.
- Some FU diagrams contains as well the Finite State Machine diagram. The following conventions apply: INPUT - the action of reading a token or a set of tokens from the input port, OUTPUT - the action of writing the token or a set of tokens to an output port.
- "Parameter" is set at network configuration stage (cannot be changed during the process) and it is characteristic for each FU
- Token RANGE: describes the mathematical interval for the token value

  Examples:
  Token RANGE: { 0, 1} – binary value
  Token RANGE: [0 .. N], $value \in [0, N\ ]$ real values, closed interval

- All the FUs require the data to be in little-endian format.

In 4.2 "FU IDs", complete Table 2 with the following lines:

Note: update the FU table..

ID	FU Name
107	Algo_Parser_SC3DMC
108	Algo_InverseQuantization1D
109	Algo_InverseQuantizationND
110	Algo_InversePrediction1D
111	Algo_InversePredictionND
112	Algo_ED_AD_StaticBit
113	Algo_ED_AD_AdaptiveBit
114	Algo_ED_VLD
115	Algo_ED_BitPrecision
116	Algo_ED_AD
117	Algo_ED_AD_EG
118	Algo_ContextModeling

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

119	Algo_ContextModeling_SVA_nType
120	Algo_ContextModeling_SVA_Indexes
121	Algo_ContextModeling_SVA_Vertex_Attribute
122	Algo_ED_4bitsD
123	Algo_ED_ FixedLength
124	Algo_LookUpTable1D
125	Algo_DecodeConnectivity_SVA
126	Algo_DecodeConnectivity_TFAN
127	Algo_ExtractMask_SC3DMC
128	Algo_ExtractFaceDirection_SVA
129	Algo_simpleMath_2op
130	Algo_Connectivity_InversePrediction_SVA
131	Mgnt_Replicate_1_2
132	Mgnt_Replicate_1_4
133	Mgnt_Replicate_1_8
134	Mgnt_MUX_2_1
135	Mgnt_MUX_4_1
136	Mgnt_MUX_8_1
137	Mgnt_DEMUX_1_2
138	Mgnt_DEMUX_1_4
139	Mgnt_DEMUX_1_8
140	Mgnt_ExtractSegment
141	Mgnt_ProviderValue
142	Mgnt_RepeatSegment
143	Mgnt_ExtractBytes
144	Mgnt_ExtractBits
145	Mgnt_Provider1D
146	Mgnt_Provider2D

4.3 "Token Pool":

Add the following rows at the end of the table.

ID & Name	Description
55 BOOLEAN	Token which value is 0 or 1.
56 SIGN	Token which value is 0 or 1.
57 FLAG	Token which value is 0 or 1.
58 UINT_2	Unsigned integer on 2 bits.
59 UINT_4	Unsigned integer on 4 bits.
60 UINT_8	Unsigned integer on 8 bits.
61 UINT_16	Unsigned integer on 16 bits.
62 UINT_32	Unsigned integer on 32 bits.

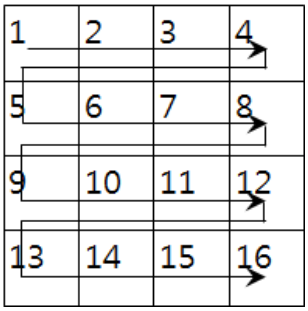
© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

63 UINT_64	Unsigned integer on 64 bits.
64 INT_8	Integer on 8 bits.
65 INT_16	Integer on 16 bits.
66 INT_32	Integer on 32 bits.
67 INT_64	Integer on 64 bits.
68 FLOAT_32	Float on 32 bits.

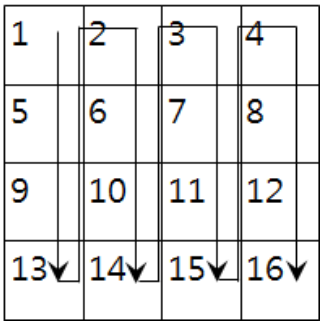
Add 4.4 "Array data order":

4.4 Array data order

- Row based:
  - The data is processed or sent in a sequential order, row by row
    Example:



- Column based
  - The data is processed or sent in a sequential order, column by column
    Example:



Add 4.5 "Input ports":

4.5 Input ports (reset_i, init_i, start_i)

An FU does not have an outside synchronization signal or synchronization mechanism. These ports are used for the purpose of changing the values of the local variables to default or initialization values.

Add 4.6 "FU block diagram notations":

4.6 FU block diagram notations

The notation [EMBED] defines a part of the main FSM schematic that is described as a separate schematic (for complexity reasons). The [EMBED] schematic is an integrated part of the main FSM schematic

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

The notation [MODULE] defines a part of the main FSM schematic that is defined as a separate FU. The module schematic is integrated in the main schematic with the entire FU logic, except the “START” FSM state. The INPUT/OUTPUT states do not read or write values from the ports, they refer to local variables relative to the FU that embeds the other schematic.

Add 4.7 "Conventions":

4.7 Conventions

The significance of the “sign” port values is:

Value	Significance
0	negative
1	positive

The significance of the “flag” values is:

Value	Significance
0	false
1	true

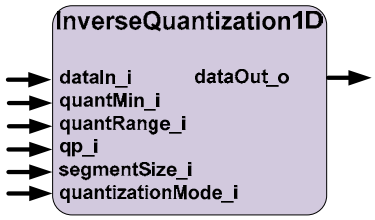
Add 5.2 "General Processing FUs":

5.2 General Processing FUs

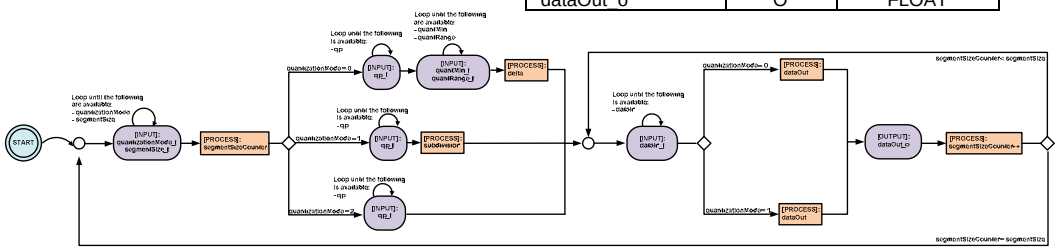
5.2.1 Algo_InverseQuantization1D

FU Name: Algo_InverseQuantization1D

Description



Port Name	Direction (I/O)	Token TYPE
dataIn_i	I	INT8, INT16, INT32, INT64
qp_i	I	UINT_32
quantMin_i	I	FLOAT
quantRange_i	I	FLOAT
segmentSize_i	I	UINT8, UINT16, UINT32, UINT64
quantizationMode_i	I	UINT_2
dataOut_o	O	FLOAT



Inverse Quantization Process:

```
START:
INPUT_SEGMENT_PARAM:
INPUT:
        quantizationMode
        segmentSize
SegmentSizeCounter = 0
IF quantizatioMode = 0
    INPUT:
            qp
            quantMin
            quantRange
    IF ( quantRange > 0.0 )
            delta = ( ( 1 << qp ) - 1 ) / quantRange
```


© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

```
    ELSE
        delta = 1.0
IF quantizationMode = 1
    INPUT:
        qp
INPUT_DATA_IN:
INPUT:
    dataIn
IF quantizatioMode = 0
    PROCESS:
        dataOut = quantMin + (dataIn / delta )
IF quantizatioMode = 1
    PROCESS:
        dataOut =  dataIn / qp
OUTPUT
    dataOut
SegmentSizeCounter ++
IF SegmentSizeCounter < segmentSize
    GOTO INPUT_DATA_IN
ELSE
    GOTO INPUT_SEGMENT_PARAM
```

The following table contains the quantization types index used in the Inverse Quantization 1D FU:

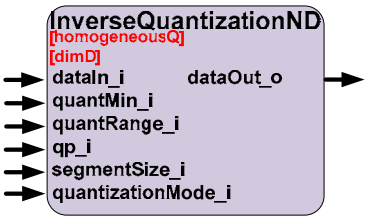
Name of quantization mode	Value
Uniform Quantization	**0**
Uniform Texture Quantization	**1**

The "Inverse Quantization" is an algorithm (step by step procedures) that allows a set of data to be represented with a limited set of values that are associated with its nearest representative.
For a number of “segmentSize” of input data (dataIn), it uses the same set of quantMin, quantRange and quantValue to produce a set of output data (dataOut) of size “segmentSize”

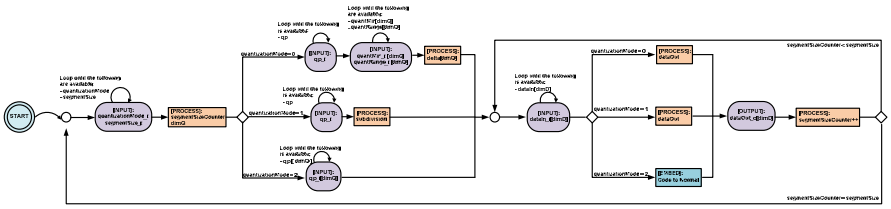
ISO Standards using the FU	ISO/IEC 14496-16:2011
Profiles@levels supported	

5.2.2 Algo_InverseQuantizationND

FU Name	Algo_InverseQuantizationND
Description	



Port Name	Direction (I/O)	Token TYPE
dataIn_i	I	INT8, INT16, INT32, INT64
qp_i	I	UINT_32
quantMin_i	I	FLOAT
quantRange_i	I	FLOAT
segmentSize_i	I	UINT8, UINT16, UINT32, UINT64
quantizationMode_i	I	UINT_2
dataOut_o	O	FLOAT



© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

Inverse Quantization Process:

$$dimQ = \begin{cases} dimD, \ if \ homogeneousQ = 0 \\ 1, \ if \ homogeneousQ = 1 \end{cases}$$

```
START:
INPUT_SEGMENT_PARAM
INPUT:
        quantizationMode
        segmentSize
SegmentSizeCounter = 0
IF quantizatioMode = 0
    INPUT:
        qp
        quantMin [ dimQ ]
        quantRange [ dimQ ]
    WHILE dimQ_counter < dimQ
        IF ( quantRange [ dimQ_counter ] > 0.0 )
            delta [ dimQ_counter ] = ( ( 1 << qp ) - 1 ) / quantRange [ dimQ_counter ]
        ELSE
            delta [ dimQ_counter ] = 1.0;
        dimQ_counter++
IF quantizationMode = 1
    INPUT:
        Qp [ dimQ ]
IF quantizationMode = 2
    INPUT:
        qp
    PROCESS:
        Subdivision = (qp  - 3) / 2

INPUT_DATA_IN:
INPUT:
    dataIn [ dimQ ]
IF quantizatioMode = 0
    PROCESS:
        dataOut = quantMin [ segmentSizeCounter%dimD ] + (dataIn / delta [segmentSizeCounter%dimD])
IF quantizatioMode = 1
    PROCESS: EMBED Code 2 Normal
IF quantizatioMode = 2
    PROCESS:
        dataOut =  dataIn / qp [ segmentSizeCounter % dimD ]
OUTPUT
    dataOut
SegmentSizeCounter ++
IF SegmentSizeCounter < segmentSize
    GOTO INPUT_DATA_IN
ELSE
    GOTO INPUT_SEGMENT_PARAM


EMBED: Code 2 Normal

    Mask = ( 1 << ( 2 * subdivision) ) - 1
    tricode = data & mask;

    // Find y coordinate by solving 2nd degree equation
    factor = 1 << subdivision
    y = factor - sqrt ( (factor^2) - tricode )
    tricode = tricode + ( y * ( y - (2 * factor) ) )
    x = tricode / 2
    upsideDown = tricode % 2

    // Calculate coordinates for all vertices in triangle
    v1x = x + upsideDown
    v1y = y + upsideDown
    v2x = x + 1
    v2y = y
    v3x = x
    v3y = y + 1

    // Calculate coordinates of barycenter
```


© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

```
invMaxCoord = 1 / factor
normal [ 0 ] = (v1x + v2x + v3x) * invMaxCoord
normal [ 1 ] = (v1y + v2y + v3y) * invMaxCoord
normal [ 2 ] = 3 - normal [ 0 ] - normal [ 1 ]

// Flip component signs if necessary
octantCode = ( data >> 2 * subdivision ) & 0x7
if  (octantCode & 0x4)
      normal [ 0 ] = (-1) * normal [ 0 ]
if (octantCode & 0x2)
      normal [ 1 ] = (-1) * normal [ 1 ]
if (octantCode & 0x1)
      normal [ 2 ] = (-1) * normal [ 2 ]

invNorm:= 1 / sqrt ( (normal[ 0 ])^2 + (normal [ 1 ])^2 + (normal [ 2 ])^2 );

//Write the 3 output values
normal [ 0 ] = normal [ 0 ] * invNorm
normal [ 1 ] = normal [ 1 ] * invNorm
normal [ 2 ] = normal [ 2 ] * invNorm

dataOut = normal
```

The following table contains the quantization types index used in the Inverse Quantization ND FU:

Name of quantization mode	Value
Uniform Quantization	**0**
Normal Quantization	**1**
Uniform Texture Quantization	**2**

The "Inverse Quantization" is an algorithm (step by step procedures) that allows a set of data to be represented with a limited set of values that are associated with its nearest representative.

For a number of “segmentSize” x “dimD” of input data (dataIn), it uses the same set of quantMin, quantRange and quantValue of size “dimD” to produce a set of output data (dataOut) of size segmentSize” x “dimD”.

For each set of size “dimD” of input data (dataIn) it uses the corresponding value of quantMin, quantRange and quantValue.

ISO Standards using the FU	ISO/IEC 14496-16:2011
Profiles@levels supported	

Parameter

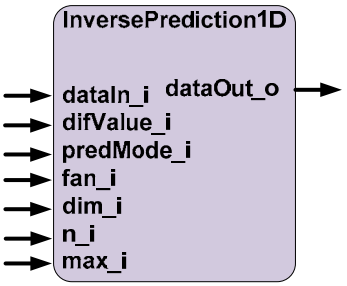
Name	Description	Type / Range
dimD	Describes the number of tokens of type dataIn_i that are consumed at each firing. This parameter is set at the network configuration level.	Type: Integer Range: [1 .. 2^5]
homogeneousQ	Describes the number of tokens of type quantRange_i, quantMin_i and quantValue_i that are necessary for the inverse quantization process. This parameter is set at the network configuration level. The number of tokens is equal to dimD if this parameter is 0 and the number of tokens is equal to 1 if this parameter is 1	Type: Boolean Range: {0,1}

© ISO/IEC 2014 – All rights reserved

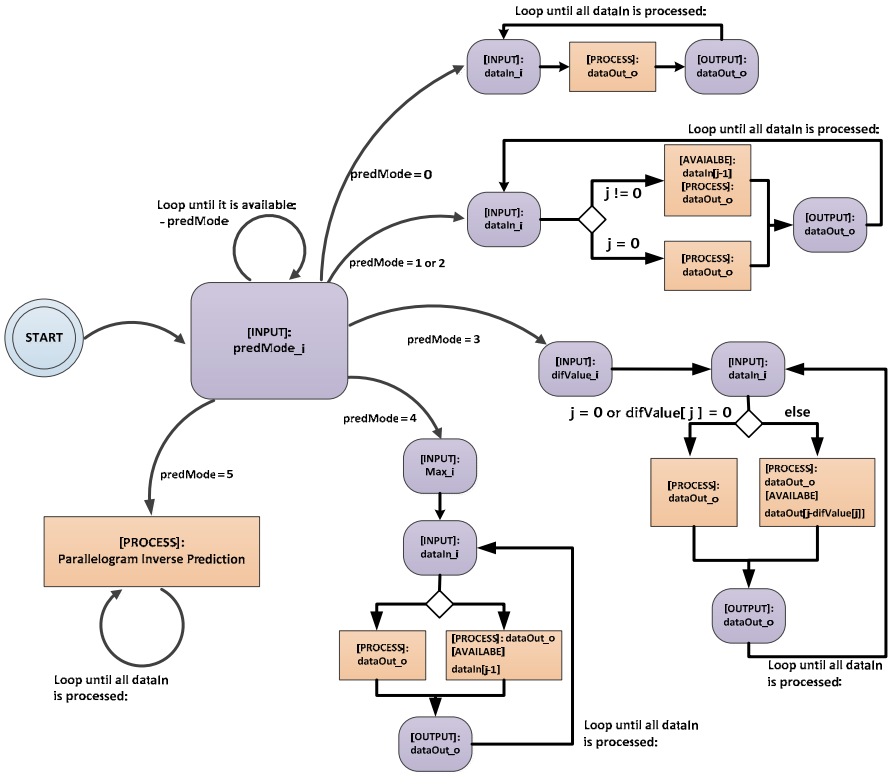
Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

5.2.3 Algo_InversePrediction1D

FU Name	Algo_InversePrediction1D
Description	This FU allows five modes of inverse prediction. Not all the ports are used for each prediction mode. This FU can handle one input token at a time.

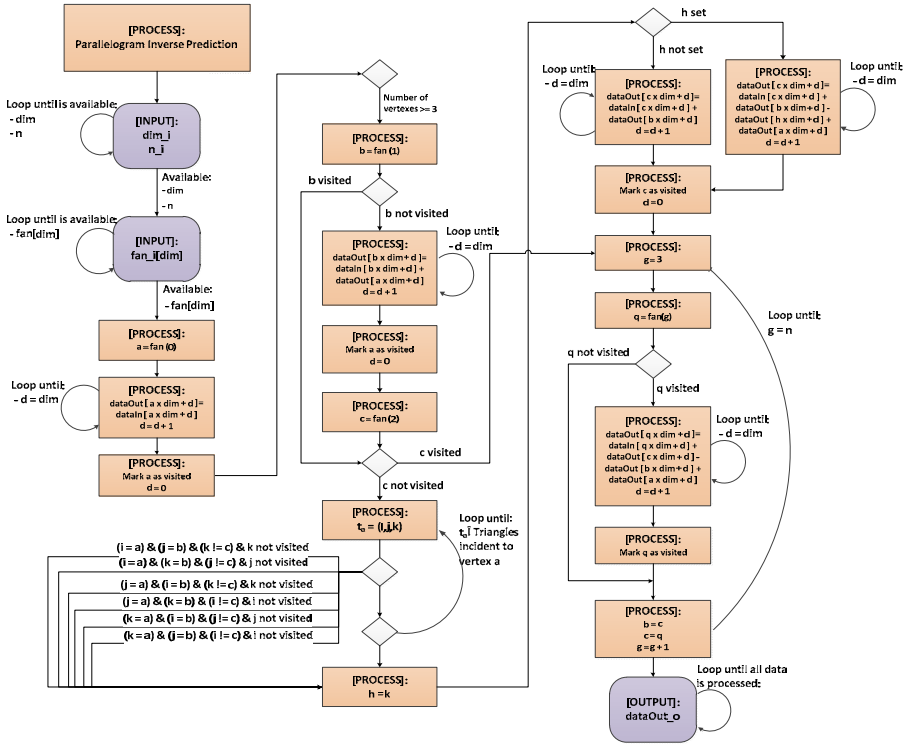


Port Name	Direction (I/O)	Token TYPE
dataIn_i	I	INT8, INT16, INT32, INT64
difValue_i	I	INT8, INT16, INT32, INT64
predMode_i	I	UINT_4
fan_i	I	INT8, INT16, INT32, INT64
dim_i	I	UINT8, UINT16, UINT32, UINT64
n_i	I	INT8, INT16, INT32, INT64
max_i	I	UINT8, UINT16, UINT32, UINT64
dataOut_o	O	INT8, INT16, INT32, INT64



Process Parallelogram Inverse Prediction Schematic (FSM):

© ISO/IEC 2014 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014



Inverse Prediction Process:

Switch (predMode)
{

Case 0: NP – No Prediction

$$dataOut[j] = dataIn[j], \quad \forall j \in \{0..N-1\}.$$

Case 1: Diff – Differential Prediction

$$dataOut[j] = \begin{cases} DataIn[j], \ \forall j = 0 \\ DataIn[j] + DataOut[j-1], \ \forall j \in \{1..N-1\} \end{cases}$$

Case 2: XOR – based prediction

$$dataOut[j] = \begin{cases} dataIn[j], \ \forall j = 0 \\ dataIn[j] \otimes dataOut[j-1], \ \forall j \in \{1..N-1\} \end{cases}$$

Case 3: Adaptive Prediction

$$dataOut[j] = \begin{cases} dataIn[j], if\ difValue[j] = 0\ or\ j = 0 \\ dataIn[j] + dataOut[j - difValue[j]], otherwise \end{cases}$$

Case 4: Circular Differential Prediction

$$dataOut[j] = \begin{cases} dataIn[j], \ \forall j = 0 \\ d\ , if\ dataIn[j] < dataIn[j-1]\ , where \\ -d, otherwise \end{cases}$$

$$d = \begin{cases} dataIn[j-1] + M_d - dataIn[j], if\ dataIn[j] > dataIn[j-1] \\ dataIn[j] + M_d - dataIn[j-1], otherwise \end{cases}$$

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

Case 5: Parallelogram Inverse Prediction

```
a = fan ( 0 )
if a not visited
    d = 0
    WHILE d < dim
        dataOut [ a x dim + d ] = dataIn [ a x dim + d ]
        d = d + 1
    Mark a as visited
If number of vertexes > 3
    b = fan ( 1 )
    if b not visited
        d = 0
        WHILE d < dim
            dataOut [ b x dim + d ] = dataIn [ b x dim + d ] + dataOut  [ a x dim + d ]
            d = d + 1
        Mark b as visited
    c = fan ( 2 )
    if c not visited
        init h, t_a
        WHILE t_a = (I,j,k) ∈Triangles incident to vertex a
            If ( i=a & j=b & k≠c and k not visited ) h = k, break
            If ( i=a & k=b & j≠c and j not visited ) h = k, break
            If ( j=a & i=b & k≠c and k not visited ) h = k, break
            If ( j=a & k=b & i≠c and i not visited ) h = k, break
            If ( k=a & i=b & j≠c and j not visited ) h = k, break
            If ( k=a & b=b & i≠c and i not visited ) h = k, break
        If h not set
            d = 0
            WHILE d < dim
                dataOut [ c x dim + d ]=dataIn [ c x dim + d ]+ dataOut [ b x dim + d ]
                d = d + 1
        else
            d = 0
            WHILE d < dim
            dataOut [ c x dim + d ] =
                                dataIn [ c x dim + d ] + dataOut [ b x dim + d ]
                                – dataOut [ h x dim + d ] + dataOut [ a x dim + d ]
            d = d + 1
        Mark c as visited
    g = 3, d = 0
    WHILE g < fanSize
        q = fan ( g )
        if q not visited
            WHILE d < dim
                dataOut [ q x dim + d ] =
                        dataIn [ q x dim + d ] + dataOut [ c x dim + d ]
                        - dataOut [ b x dim + d ] + dataOut [ a x dim + d ]
                d = d + 1
            b = c
            c = q
```

The following table contains the prediction types index used in the Inverse Prediction 1D FU:

Name of prediction mode	Value
No Prediction	**0**
Differential Prediction	**1**
XOR based prediction	**2**

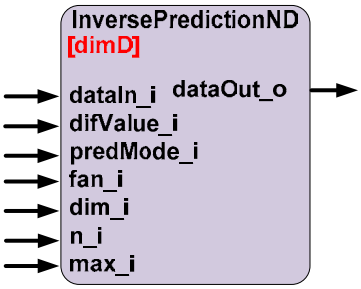
© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

	Adaptive Differential Prediction **3** Circular Differential Prediction **4** Parallelogram Inverse Prediction **5** The detailed description of the inverse parallelogram prediction is described in Annex F.
ISO Standards using the FU	ISO/IEC 14496-16:2011
Profiles@levels supported	

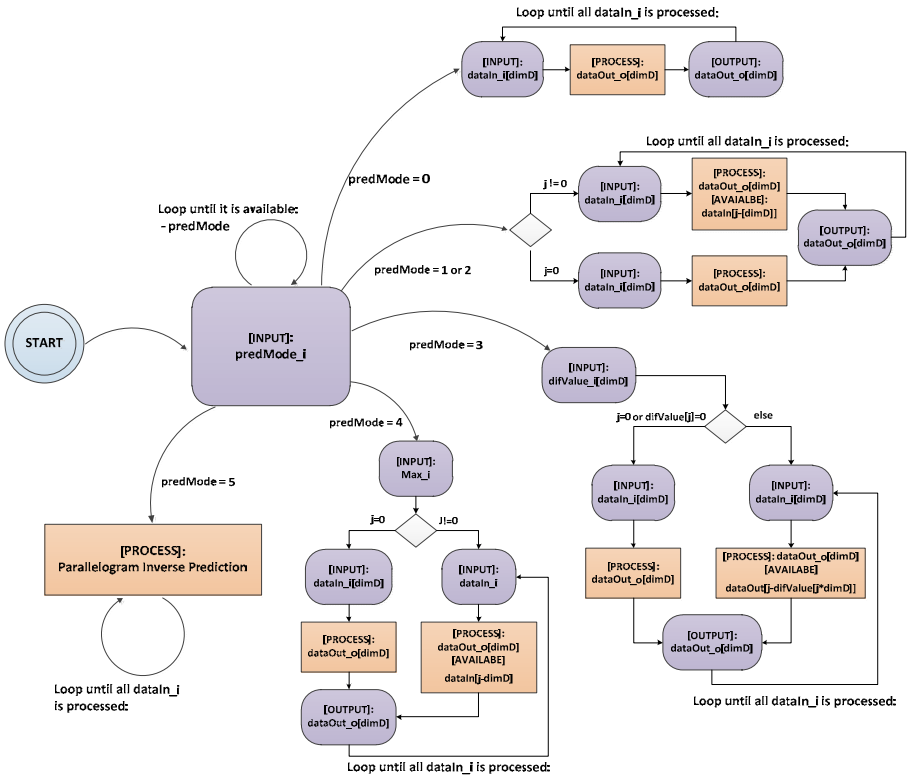
Parameter		
Name	**Description**	**Range**

5.2.4 Algo_InversePredictionND

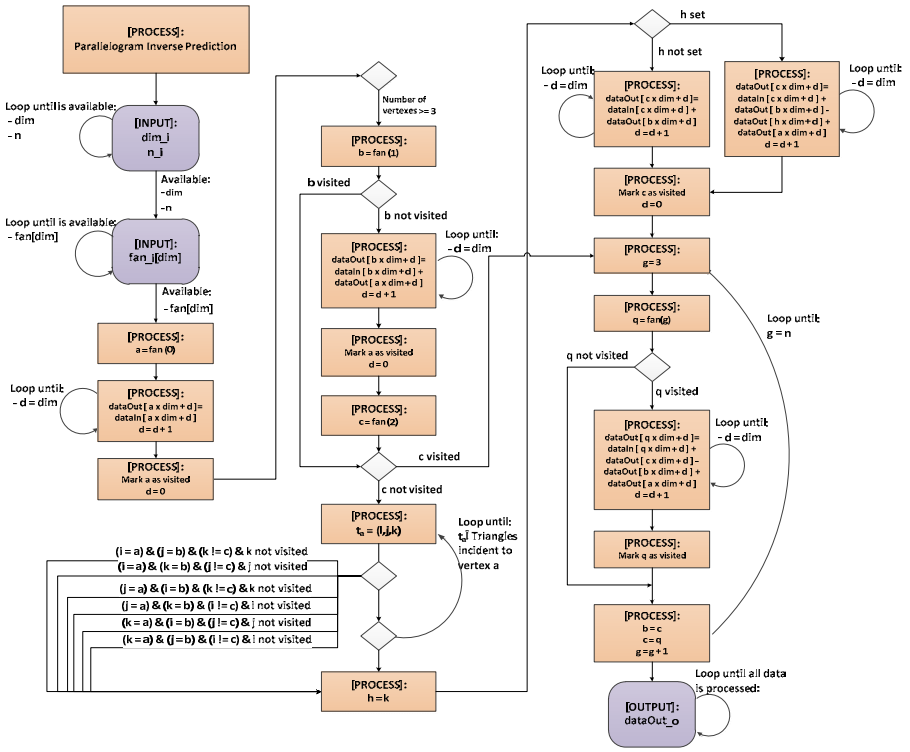
FU Name	Algo_InversePredictionND
Description	This FU allows five modes of inverse prediction. Not all the ports are used for each prediction mode. This FU can handle a number of dimD input tokens at a time.  Port Name / Direction (I/O) / Token RANGE dataIn_i / I / INT8, INT16, INT32, INT64 difValue_i / I / INT8, INT16, INT32, INT64 predMode_i / I / UINT_4 fan_i / I / INT8, INT16, INT32, INT64 dim_i / I / UINT8, UINT16, UINT32, UINT64 n_i / I / INT8, INT16, INT32, INT64 max_i / I / UINT8, UINT16, UINT32, UINT64 dataOut_o / O / INT8, INT16, INT32, INT64 Process Schematic (FSM):

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014



Process Parallelogram Inverse Prediction Schematic (FSM):



Inverse Prediction ND Process:

Switch(predMode)

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

{

Case 0: NP – No Prediction

$$dataOut[j] = dataIn\,[j],\ \ \forall j \in \{0\,..(N\ x\ dimD)-1\}$$

Case 1: Diff – Differential Prediction

$$dataOut[j] = \begin{cases} dataIn[j],\ \forall j = 0 \\ dataIn[j] + dataOut[j - dimD],\ \ \forall j \in \{1\,..(N\ x\ dimD)-1\} \end{cases}$$

Case 2: XOR – based prediction

$$dataOut[j] = \begin{cases} dataIn[j],\ \forall j = 0 \\ dataIn[j] \otimes dataOut[j - dimD],\ \ \forall j \in \{1\,..(N\ x\ dimD)-1\} \end{cases}$$

Case 3: Adaptive Prediction

$$dataOut[j] = \begin{cases} dataIn[j],\ \text{if } difValue[j] = 0 \text{ or } j = 0 \\ dataIn[j] + dataOut[j - difValue[j] \otimes dimD],\ \text{otherwise} \end{cases}$$

Case 4: Circular Differential Prediction

$$dataOut[j]\ = \begin{cases} DataIn[j],\ \forall j = 0 \\ d\,,\ \text{if } dataIn[j]\ < dataIn[j - dimD]\ \text{, where} \\ -d\,,\text{otherwise} \end{cases}$$

$$d = \begin{cases} dataIn[j - dimD] + M_d - dataIn[j],\ \text{if } dataIn[j] > dataIn[j - dimD] \\ dataIn[j] + M_d - dataIn[j - dimD],\ \text{otherwise} \end{cases}$$

}

Case 5: Parallelogram Inverse Prediction

```
a = fan ( 0 )
if a not visited
    d = 0
    WHILE d < dim
        dataOut [ a x dim + d ] = dataIn [ a x dim + d ]
        d = d + 1
    Mark a as visited
If number of vertexes > 3
    b = fan ( 1 )
    if b not visited
        d = 0
        WHILE d < dim
            dataOut [ b x dim + d ] = dataIn [ b x dim + d ] + dataOut  [ a x dim + d ]
```

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

```
                d = d + 1
            Mark b as visited
        c = fan ( 2 )
        if c not visited
            init h, t_a
            WHILE t_a = (I,j,k) ∈Triangles incident to vertex a
                If ( i=a & j=b & k≠c and k not visited ) h = k, break
                If ( i=a & k=b & j≠c and j not visited ) h = k, break
                If ( j=a & i=b & k≠c and k not visited ) h = k, break
                If ( j=a & k=b & i≠c and i not visited ) h = k, break
                If ( k=a & i=b & j≠c and j not visited ) h = k, break
                If ( k=a & b=b & i≠c and i not visited ) h = k, break
            If h not set
                d = 0
                WHILE d < dim
                    dataOut [ c x dim + d ]=dataIn [ c x dim + d ]+ dataOut [ b x dim + d ]
                    d = d + 1
            else
                d = 0
                WHILE d < dim
                dataOut [ c x dim + d ] =
                                dataIn [ c x dim + d ] + dataOut [ b x dim + d ]
                                – dataOut [ h x dim + d ] + dataOut [ a x dim + d ]
                d = d + 1
            Mark c as visited
        g = 3, d = 0
        WHILE g < fanSize
            q = fan ( g )
            if q not visited
                WHILE d < dim
                    dataOut [ q x dim + d ] =
                            dataIn [ q x dim + d ] + dataOut [ c x dim + d ]
                            - dataOut [ b x dim + d ] + dataOut [ a x dim + d ]
                    d = d + 1
                b = c
                c = q
```

The following table contains the prediction types index used in the Inverse Prediction ND FU:

Name of prediction mode	Value
No Prediction	**0**
Differential Prediction	**1**
XOR based prediction	**2**
Adaptive Differential Prediction	**3**
Circular Differential Prediction	**4**
Parallelogram Inverse Prediction	**5**

The detailed description of the inverse parallelogram prediction is described in Annex F.

ISO Standards using the FU	ISO/IEC 14496-16:2011
Profiles@levels supported	

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

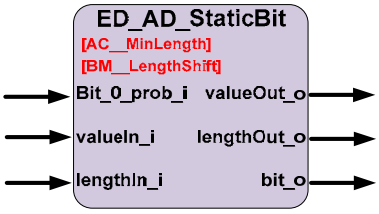
Parameter		
Name	**Description**	**Range**
dimD	Describes the number of tokens of type dataIn_i that are consumed at each firing. This parameter is set at the network configuration level.	Type: Integer Range: [1 .. 2^5]

5.2.5 Algo_ED_AD_StaticBit

FU Name: Algo_ED_AD_StaticBit

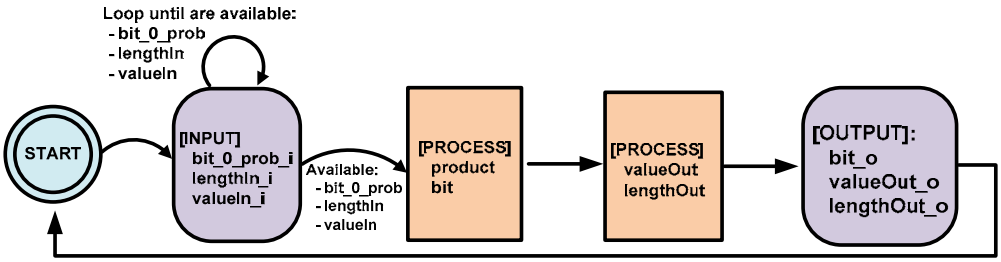
Description:

This FU describes the arithmetic decoding process based on a static bit model.



Port Name	Direction (I/O)	Token RANGE
Bit_0_prob_i	I	UINT8, UINT16, UINT32, UINT64
valueIn_i	I	UINT8, UINT16, UINT32, UINT64
length_i	I	UINT8, UINT16, UINT32, UINT64
valueOut_o	O	UINT8, UINT16, UINT32, UINT64
lengthOut_o	O	UINT8, UINT16, UINT32, UINT64
bit_o	O	UINT8, UINT16, UINT32, UINT64

Process Schematic (FSM):



ED_AD_StaticBit Process:

```
START
INPUT:
    Bit_0_prob_i
    lengthIn_i
    valueIn_i
product = bit_0_prob * ( lengthIn >> BM_LengthShift )
bit = ( valueIn >= product )
if ( bit == 0 )
    valueOut = valueIn
    lengthOut = product
else
    valueOut = valueIn – product
    lengthOut = lengthIn – product
OUTPUT:
valueOut_o
lengthOut_o
GOTO START
```

ISO Standards using the FU: ISO/IEC 14496-16:2011

Profiles@levels supported:

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

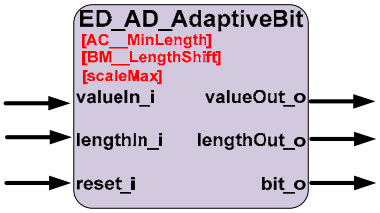
Parameter		
Name	**Description**	**Range**
AC__MinLength	Describes the threshold for renormalization. This parameter is set at the network configuration level.	Type:Unsigned Integer Range: [1 .. 2^{32}]
BM__LengthShift	Describes the length bits discarded before multiplication. This parameter is set at the network configuration level.	Type:Unsigned Integer Range: [1 .. 2^{5}]

5.2.6 Alog_ED_AD_AdaptiveBit

FU Name: Algo_ED_AD_AdaptiveBit

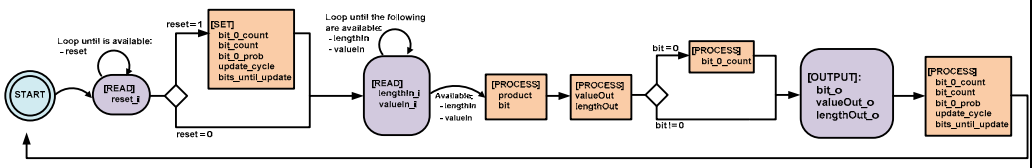
Description

This FU describes the arithmetic decoding process based on a adaptive bit model as presented in [3,4].



Port Name	Direction (I/O)	Token RANGE
valueIn_i	I	UINT8, UINT16, UINT32, UINT64
length_i	I	UINT8, UINT16, UINT32, UINT64
reset_i	I	BOOLEAN
valueOut_o	O	UINT8, UINT16, UINT32, UINT64
lengthOut_o	O	UINT8, UINT16, UINT32, UINT64
bit_o	O	UINT8, UINT16, UINT32, UINT64

Process Schematic (FSM):



ED_AD_AdaptiveBit Process:

```
START
If reset = 1
    bit_0_count = 1;
    bit_count   = 2;
    bit_0_prob  = 1U << (BM__LengthShift - 1);
    update_cycle = bits_until_update = 4;

product = bit_0_prob * ( lengthIn >> BM_LengthShift )
bit = ( valueIn >= product )

if ( bit == 0 )
    valueOut = valueIn
    lengthOut = product
    ++bit_0_count
else
    valueOut = valueIn – product
    lengthOut = lengthIn – product

if (bit_count += update_cycle) > BM__MaxCount
    bit_count = (bit_count + 1) >> 1
    bit_0_count = (bit_0_count + 1) >> 1
    if bit_0_count = bit_count
        ++bit_count
scale = scaleMax / bit_count;
bit_0_prob = (bit_0_count * scale) >> (31 - BM__LengthShift);
update_cycle = (5 * update_cycle) >> 2;
```


© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

	if update_cycle > 64
	update_cycle = 64;
	bits_until_update = update_cycle;
	GOTO START	

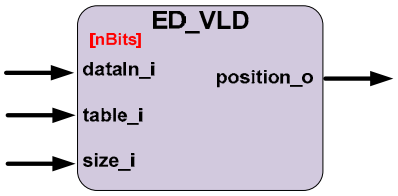
ISO Standards using the FU	ISO/IEC 14496-16:2011	
Profiles@levels supported		
Parameter		
Name	**Description**	**Range**
AC__MinLength	Describes the threshold for renormalization. This parameter is set at the network configuration level.	Type:Unsigned Integer Range: [1 .. 2^{32}]
BM__LengthShift	Describes the length bits discarded before multiplication. This parameter is set at the network configuration level.	Type:Unsigned Integer Range: [1 .. 2^{5}]
scaleMax	Describes the max value to compute the scaled bit 0 probability. This parameter is set at the network configuration level.	Type:Unsigned Integer Range: [1 .. 2^{32}]

5.2.7 Algo_ED_VLD

FU Name: Algo_ED_VLD

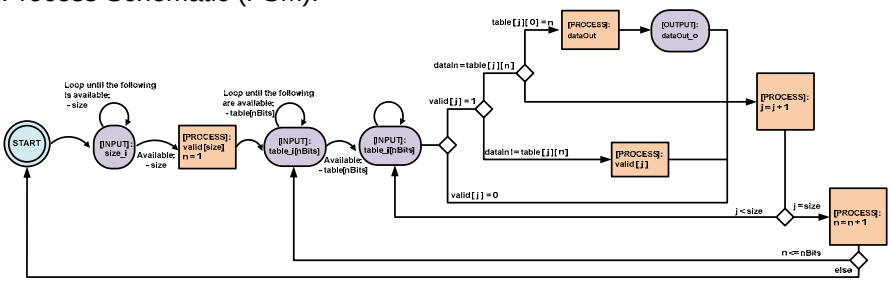
Description

This FU describes the Variable Length Decoding process.



Port Name	Direction (I/O)	Token RANGE
dataIn_i	I	UINT8, UINT16, UINT32, UINT64
table_i	I	UINT8, UINT16, UINT32, UINT64
size_i	I	UINT8, UINT16, UINT32, UINT64
position_o	O	UINT8, UINT16, UINT32, UINT64

Process Schematic (FSM):



ED_VLD Process:

```
START
SET valid[size] = 1
SET n = 1
DO
    DO
        If valid[j] = 1
            If dataIn=table[j][n]
                If table[j][0] = n
                    dataOut = j
            else
                valid[j] = 0
        j = j + 1
    WHILE j < size
```

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

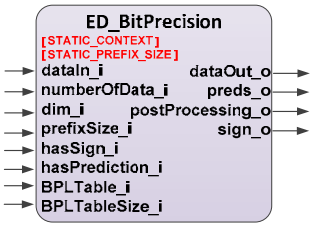
	n = n + 1 WHILE n < nBits GOTO START The matrix tokens have to be send column based.	
ISO Standards using the FU	ISO/IEC 14496-16:2011	
Profiles@levels supported		
Parameter		
Name	**Description**	**Range**
nBits	Describes the length of the bits used for the search algorithm. This parameter is set at the network configuration level.	Type: Integer Range: [1 .. 2^5]

5.2.8 Algo_ED_BitPrecision

FU Name: Algo_ED_BitPrecision

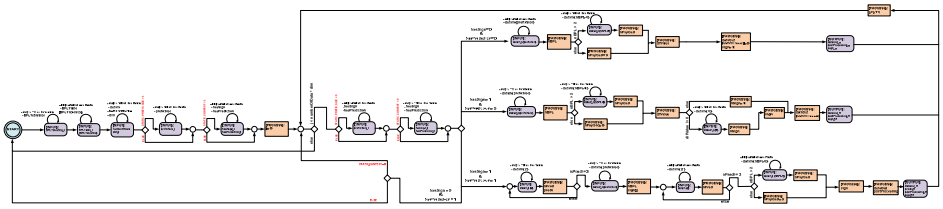
Description

This FU describes the Entropy Decoding Bit Precision processes.



Port Name	Direction (I/O)	Token RANGE
dataIn_i	I	UINT_8
numerOfData_i	I	UINT8, UINT16, UINT32, UINT64
dim_i	I	UINT8, UINT16, UINT32, UINT64
prefixSize_i	I	UINT_8
hasSign_i	I	BOOLEAN
hasPrediction_i	I	BOOLEAN
BPLTable_i	I	UINT8, UINT16, UINT32, UINT64
BPLTableSize_i	I	UINT8, UINT16, UINT32, UINT64
dataOut_o	O	INT8, INT16, INT32, INT64
preds_o	O	INT8, INT16, INT32, INT64
postProcessing_o	O	BOOLEAN
sign_o	O	BOOLEAN

ED_BitPrecision Schematic (FSM):



ED_BitPrecision process:

```
START
INPUT: BPLTableSize_i
INPUT: BPLTable_i [ BPLTableSize ]
INPUT: numberOfData_i
INPUT: dim_i
IF STATIC_CONTEXT = 1
    INPUT: hasSign_i
    INPUT: hasPrediction_i
IF STATIC_PREFIX_SIZE = 1
    INPUT: prefixSize_i
j = 0
```


© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

```
PROCESS_CHUNK:
WHILE j < numberOfData * dim
    IF STATIC_PREFIX_SIZE = 0
        INPUT: prefixSize_i
    IF STATIC_CONTEXT = 0
        INPUT: hasSign_i
        INPUT: hasPrediction_i
    IF hasSign = 0 & hasPrediction = 0
        INPUT: dataIn { prefixSize }
        nBPL = dataIn
        IF nBPL > 2
            INPUT: dataIn { nBPL – 1 }
            nPayLoad = dataIn
        ELSE
            nPayLoad = 0
        difValue = BPLTable [ nBPL ] + nPayLoad
        dataOut = difValue
        postProcessing = 0
        sign = 0
        OUTPUT: dataOut
        OUTPUT: postProcessing_o
        OUTPUT: sign_o
            ELSE IF hasSign = 1 & hasPrediction = 0
                INPUT: dataIn { prefixSize }
                nBPL = dataIn
                if nBPL > 2
                    INPUT: dataIn { nBPL – 1 }
                    nPayLoad = dataIn
                ELSE
                    nPayLoad = 0
                difValue = BPLTable [ nBPL ] + nPayLoad
                IF difValue != 0
                    INPUT: dataIn { 1 }
                    nSign = dataIn
                ELSE
                    nSign = 1
                sign = nSign
                postProcessing = 1
                dataOut = difValue
                OUTPUT: dataOut
                OUTPUT: postProcessing_o
                OUTPUT: sign_o
            ELSE IF hasSign = 1 & hasPrediction = 1
                signDef[2] = { 1,-1 }
                predsOut [ j ] = 0
                DO
                    INPUT: dataIn { 2 }
                    nPred = dataIn
                    preds += nPred
                    WHILE nPred != 3
                INPUT: dataIn { prefixSize }
                nBPL = dataIn
                IF ( nBPL > 2 )
                    INPUT: dataIn { nBPL – 1 }
                    nPayLoad = dataIn
                ELSE
                    nPayLoad = 0
                difValue = BPLTable [ nBPL ] + nPayLoad
                dataOut = difValue
                IF preds != 0
                    postProcessing = 1
                ELSE
                    postProcessing = 0
                IF postProcessing != 0
                    sign = signDef [ difValue % 2 ]
                ELSE
                    sign = 1
                OUTPUT: dataOut_o
                OUTPUT: preds_o
                OUTPUT: postProcessing_o
                OUTPUT: sign_o
            ELSE IF hasSign = 1 & hasPrediction = 0
                IF STATIC_CONTEXT = 0
                    PROCESS_CHUNK
                ELSE
```

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

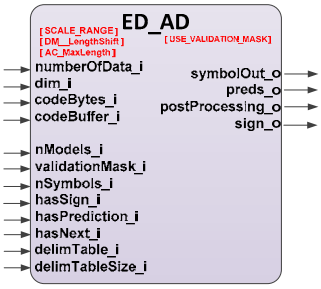
	GOTO START GOTO START An input of size {N} has the meaning of a input of a N-size bit value. Note: The case when both hasSign=1 and hasPrediction=0 is not considered. The behaviour in this case is to return to the START stage (if the parameter STATIC_CONTEXT=1) or to process the next data (if the parameter STATIC_CONTEXT=0).
ISO Standards using the FU	ISO/IEC 14496-16:2011
Profiles@levels supported	

Parameter		
Name	**Description**	**Range**
STATIC_CONTEXT	Describes whether the context data (hasSign and hasPrediction) is read for every processing iteration. If set to 1, the context data is read only once and reused during the process, if set to 0, the context data is read for every processing interation. This parameter is set at the network configuration level.	Type:Boolean Range: {0,1}
STATIC_PREFIX_SIZE	Describes whether the prefix size value is read for every processing iteration. If set to 1, the prefix value is read only once and reused during the process, if set to 0, the prefix value is read for every processing interation. This parameter is set at the network configuration level	Type:Boolean Range: {0,1}

5.2.9 Algo_ED_AD

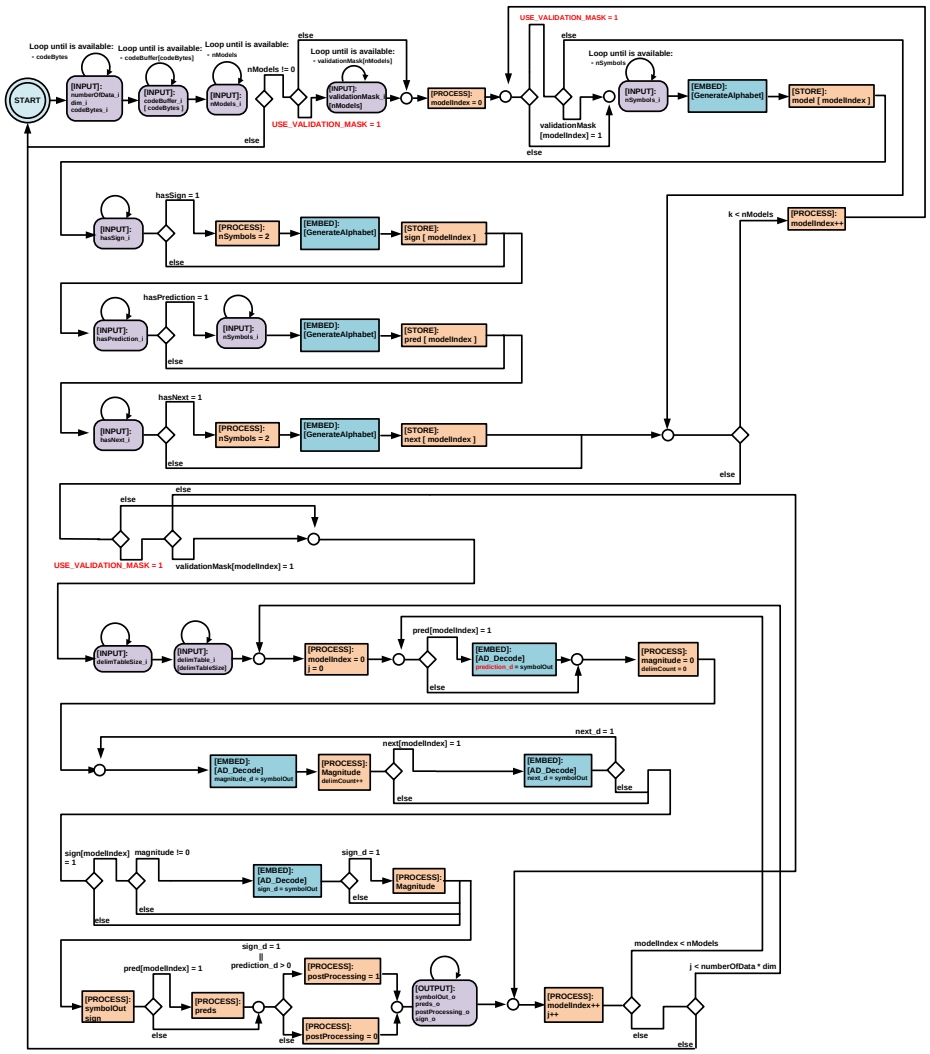
FU Name	Algo_ED_AD
Description	This FU describes the Entropy Decoding Arithmetic Decoding processes.

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014



Port Name	Direction (I/O)	Token RANGE
numberOfData_i	I	UINT8, UINT16, UINT32, UINT64
dim_i	I	UINT8, UINT16, UINT32, UINT64
codeBytes_i	I	UINT8, UINT16, UINT32, UINT64
codeBuffer_i	I	UINT_8
nModels_i	I	UINT_8
validationMask_i	I	BOOLEAN
nSymbols_i	I	UINT8, UINT16, UINT32, UINT64
hasSign_i	I	BOOLEAN
hasPrediction_i	I	BOOLEAN
hasNext_i	I	BOOLEAN
delimTableSize_i	I	UINT_8
delimTable_i	I	UINT8, UINT16, UINT32, UINT64
symbolOut_o	O	INT8, INT16, INT32, INT64
preds_o	O	INT8, INT16, INT32, INT64
postProcessing_o	O	BOOLEAN
sign_o	O	BOOLEAN

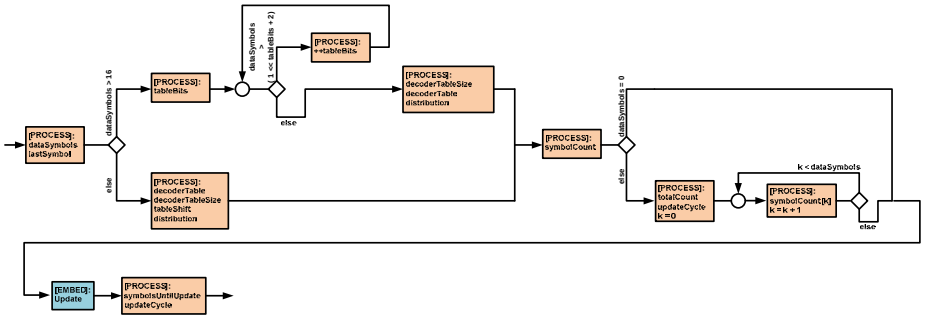
ED_AD Schematic (FSM):



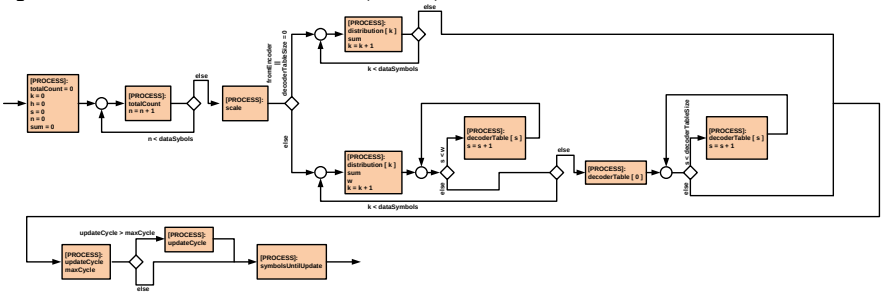
Generate Alphabet Process Schematic (FSM):

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

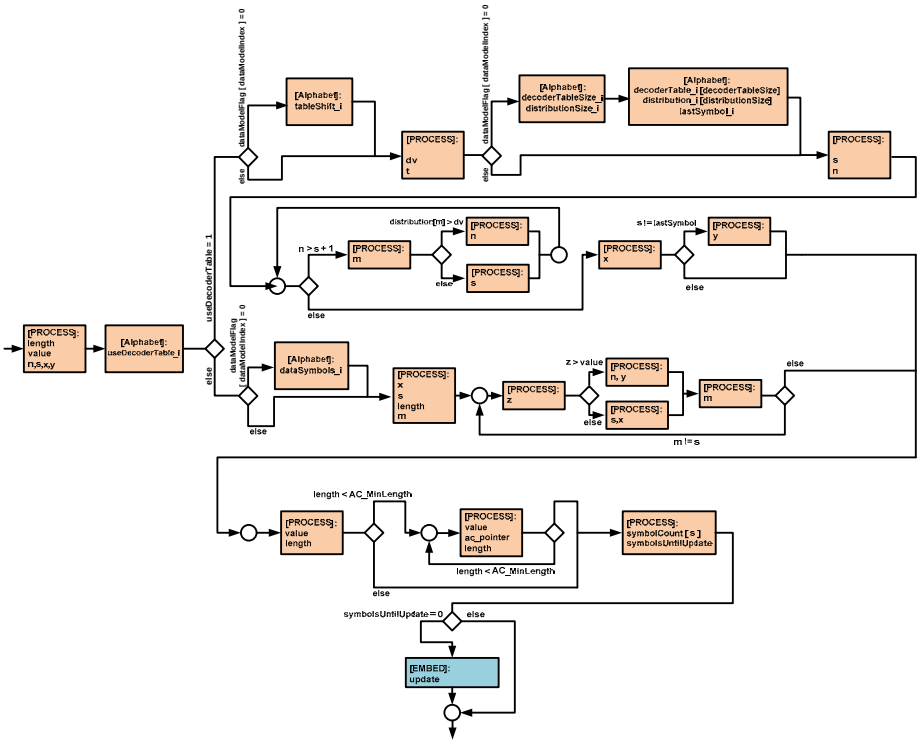


Update - Process Schematic (FSM):



Generate Alphabet Process:

Arithmetic Decode Adaptive Data - Process Schematic (FSM):



ED_AD process:

START
INPUT: numberOfData_i
INPUT: dim_i
INPUT: codeBytes_i
INPUT: codeBuffer_i[codeBytes]
INPUT: nModels_i
if nModels != 0
    if USE_VALIDATION_MASK = 1
        INPUT: validationMask_i [nModels]
    modelIndex = 0
    k = 0

© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

```
DO
  if USE_VALIDATION_MASK = 1
    if validationMask [ modelIndex ] = 1
        GOTO MODEL_CONFIG
    else
        GOTO NEXT_ MODEL_CONFIG
  else
    GOTO MODEL_CONFIG

  MODEL_CONFIG:
     INPUT: nSymbols_i
     EMBED: Generate Alphabet
     STORE: model [ modelIndex ]
     INPUT: hasSign_i
     if hasSign = 1
        nSymbols = 2
        EMBED: Generate Alphabet
        STORE: sign [ modelIndex ]
     INPUT: hasPrediction_i
     if hasPrediction = 1
        INPUT: nSymbols_i
        EMBED: Generate Alphabet
        STORE: pred [ modelIndex ]
     INPUT: hasNext_i
        nSymbols = 2
        EMBED: Generate Alphabet
        STORE: next [ modelIndex ]
     k = k + 1
     WHILE k < nModels
INPUT: delimTableSize_i
INPUT: delimTable_i [ delimTableSize ]

INIT:
modelIndex = 0
j = 0
DECODE:
    if USE_VALIDATION_MASK = 1
        if validationMask [ modelIndex ] = 1
            GOTO MODEL_PROCESS
        else
            GOTO NEXT_MODEL
    else
        GOTO MODEL_PROCESS

 MODEL_PROCESS:
     if pred[modelIndex] = 1
        EMBED: AD_Decode (prediction_d = symbolOut)
     magnitude = 0
     delimCount = 0
     HAS_NEXT:
        EMBED: AD_DECODE (magnitude_d = symbolOut)
        Magnitude
        delimCount++
        if next [ modelIndex ] = 1
            EMBED: AD_DECODE (next_d = symbolOut)
        if next_d = 1
            GOTO HAS_NEXT
        if sign[modelIndex] = 1
            if magnitude != 0
                EMBED: AD_DECODE (sign_d = symbolCount)
        symbolOut = magnitude
        sign = sign_d
        if pred[modelIndex] = 1
            preds = prediction_d
        if sign_d = 1 || prediction_d > 0
            postProcessing = 1
        else
            postProcessing = 0
        OUTPUT: symbolOut_o
        OUTPUT: preds_o
        OUTPUT: postProcessing_o
        OUTPUT: sign_o

        NEXT_MODEL:
            modelIndex++
```

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

```
                    j++

                if modelIndex < nModels
                        GOTO DECODE
                if j < numberOfData * dim
                        GOTO INIT
GOTO START
```

Generate Alphabet Process:

```
dataSymbols = nSymbols
lastSymbol = nSymbols – 1
if dataSymbols > 16
        tableBits = 3
        WHILE dataSymbols > ( 1 << (tableBits + 2) )
                ++tableBits
        tableSize = 1 << tableBits
        tableShift = DM__LengthShift – tableBits
        distribution [ 2 * dataSymbols + tableSize + 2]
        decoderTable = distribution + 2 * dataSymbols
else
        decoderTable = 0
        tableSize = 0
        tableShift = 0
        distribution [ 2 * dataSymbols ]
symbolCount = distribution + dataSymbols
if dataSymbols != 0
totalCount = 0
updateCycle = dataSymbols
k = 0
WHILE k < dataSymbols
        symbolCount [ k ] = 1
        k = k + 1
EMBED: Update
symbolsUntilUpdate = ( dataSymbols + 6 ) >> 1
updateCycle = ( dataSymbols + 6 ) >> 1
```

Update – Process:

```
totalCount = 0
k = 0
h = 0
s = 0
n = 0
sum = 0
WHILE n < dataSymbols
        totalCount += ( symbolCount [ n ] = (symbolCount [ n ] + 1 ) >> 1 )
scale = SCALE_RANGE / totalCount
if fromEncofer = true | tableSize = 0
        WHILE k < dataSymbols
                distribution [ k ] = ( scale * sum ) >> ( 31 – DM_LengthShift )
                sum += symbolCount [ k ]
                k = k + 1
else
        WHILE k < dataSymbols
                distribution [ k ] = ( scale * sum ) >> ( 31 – DM_LengthShift )
                sum += symbolCount [ k ]
                w = distribution [ k ] >> tableShift
                WHILE  s < w
                        decoderTable [++s] = k – 1
        decoderTable [ 0 ] = 0
        WHILE s <= tableSize
                decoderTable [++s] = dataSymbols – 1
updateCycle = ( 5 * updateCycle ) >> 2
maxCycle = ( dataSymbols + 6 ) << 3
if ( updateCycle > maxCycle )
        updateCycle = maxCycle
symbolsUntilUpdate = updateCycle
```

Arithmetic Decode Adaptive Data Process:

```
codeBytes = 0
WHILE codeBytes <= 0
        INPUT: codeBytes
length = 0
```


© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

```
value = 0
n = 0
s = 0
x = 0
y = 0
if useDecoderTable = 1
    if dataModelFlag [ dataModelIndex ] = 0
        INPUT: tableShift
    dv  = value / ( length >>= DM__LengthShift )
    t = dv >> tableShift
    s = decoderTable [ t ]
    n = decoderTable [ t+1 ] + 1
    WHILE n > s + 1
        m = ( s + n ) >> 1
        if distribution [ m ] > dv
            n = m
        else
            s = m
    x = distribution [ s ] * length
    if s != lastSymbol
        y = distribution [ s + 1] * length
else
    x = 0
    s = 0
    length >>= DM_LengthShift
    m = ( n = dataSymbols ) >> 1
    DO
        z = length * distribution [ m ]
        if z > value
            n = m
            y = z
        else
            s = m
            x = z
        WHILE m = ((s+n) >> 1 ) != s
value -= x
length = y – x
if length < AC__MinLength
DO
    value = (value << 8) | ++ac_pointer
    WHILE (length <<= 8) < AC__MinLength
++symbolCount [ s ]
if  -- symbolsUntilUpdate = 0
    EMBED: Update
```

ISO Standards using the FU	ISO/IEC 14496-16:2011
Profiles@levels supported	

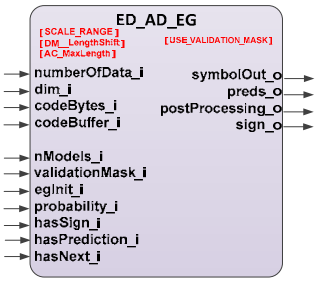
Parameter		
AC__MaxLength	Describes the maximum AC interval. This parameter is set at the network configuration level.	Type:Unsigned Integer Range: [1 .. 2^{32}]
DM__LengthShift	Describes the number of bits discarded before multiplication. This parameter is set at the network configuration level.	Type:Unsigned Integer Range: [1 .. 2^{32}]
SCALE_RANGE	Describes the range for the scale value. This parameter is set at the network configuration level.	Type:Unsigned Integer Range: [1 .. 2^{32}]
USE_ VALIDATION_MASK	Indicates whether to use the validation mask input port or not.(0 – NO, 1 – YES)	Type:Unsigned Integer Range: {0,1}

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

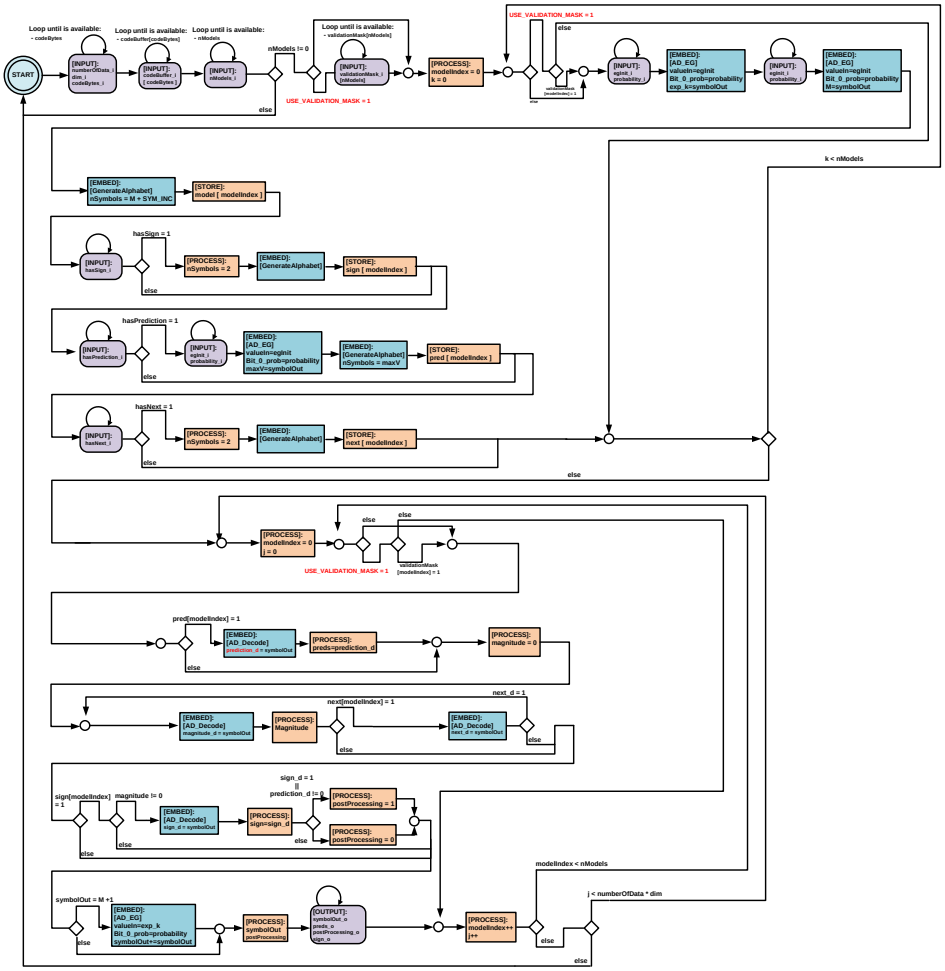
5.2.10 Algo_ED_AD_EG

FU Name	Algo_ED_AD_EG
Description	This FU describes the Entropy Decoding Arithmetic Decoding Exponential Golomb processes.



Port Name	Direction (I/O)	Token RANGE
numberOfData_i	I	UINT8, UINT16, UINT32, UINT64
dim_i	I	UINT8, UINT16, UINT32, UINT64
codeBytes_i	I	UINT8, UINT16, UINT32, UINT64
codeBuffer_i	I	UINT_8
nModels_i	I	UINT_8
validationMask_i	I	BOOLEAN
egInit_i	I	UINT8, UINT16, UINT32, UINT64
probability_i	I	UINT8, UINT16, UINT32, UINT64
hasSign_i	I	BOOLEAN
hasPrediction_i	I	BOOLEAN
hasNext_i	I	BOOLEAN
symbolOut_o	O	INT8, INT16, INT32, INT64
preds_o	O	INT8, INT16, INT32, INT64
postProcessing_o	O	BOOLEAN
sign_o	O	BOOLEAN

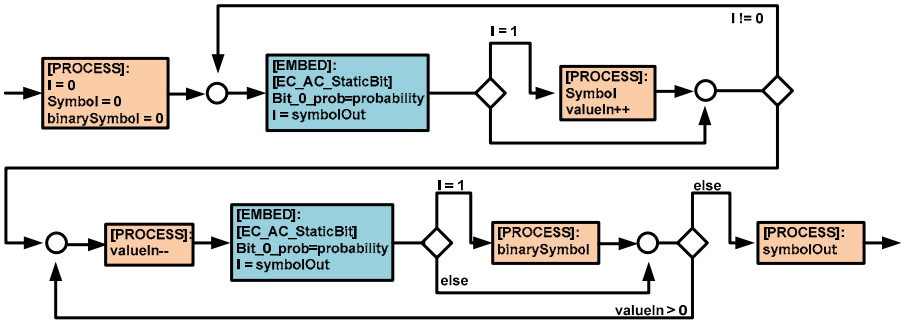
ED_AD_EG Schematic (FSM):



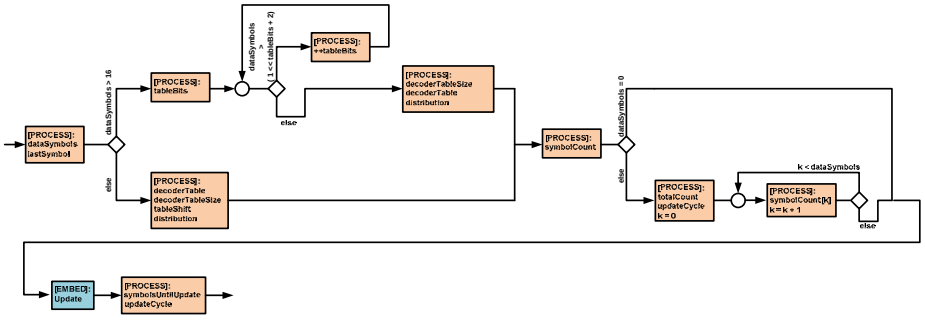
© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014

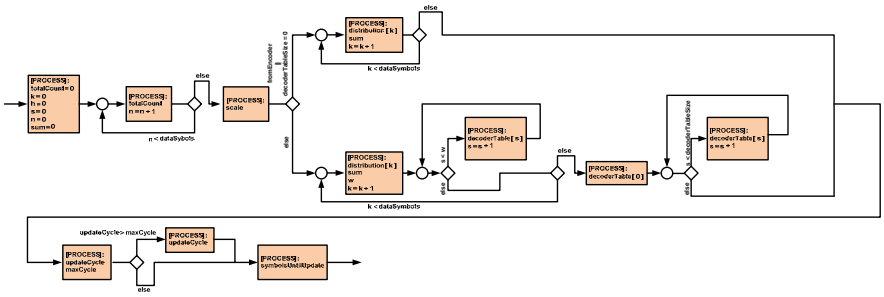
ED_AD_EG Process Schematic (FSM):



Generate Alphabet Process Schematic (FSM):



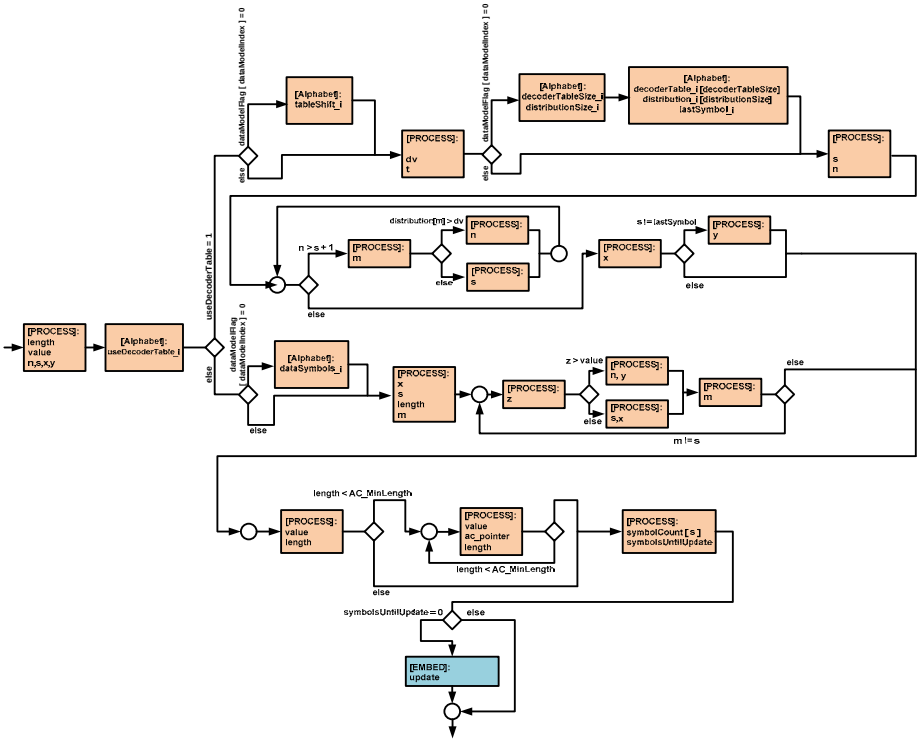
Update - Process Schematic (FSM):



Arithmetic Decode Adaptive Data - Process Schematic (FSM):

© ISO/IEC 2014 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014



ED_AD_EG process:

```
START
INPUT: numberOfData_i
INPUT: dim_i
INPUT: codeBytes_i
INPUT: delimTableSize_i
INPUT: codeBuffer_i[codeBytes]
INPUT: delimTable_i[delimTableSize]
INPUT: nModels_i
if USE_VALIDATION_MASK = 1
    INPUT: validationMask[nModels]
if nModels != 0
    modelIndex = 0
    k = 0
     INPUT: egInit_i
     INPUT: probability_i
     EMBED: AD_EG (valueIn=egInit, Bit_0_prob=probability,exp_k=symbolOut)
     INPUT: egInit_i
     INPUT: probability_i
     EMBED: AD_EG (valueIn=egInit, Bit_0_prob=probability,exp_k=symbolOut)
    DO
        if USE_VALIDATION_MASK = 1
            if validationMask [nModels] = 1
                EMBED: Generate Alphabet (nSymbols=M+SYM_INC)
                STORE: model [ modelIndex ]
                INPUT: hasSign_i
                if hasSign = 1
                    nSymbols = 2
                    EMBED: Generate Alphabet
                    STORE: sign [ modelIndex ]
                INPUT: hasPrediction_i
                if hasPrediction = 1
                    INPUT: egInit_i
                    INPUT: probability_i
                    EMBED: AD_EG (valueIn=egInit,
Bit_0_prob=probability,maxV=symbolOut)
                    EMBED: Generate Alphabet (nSymbols=maxV)
                    STORE: pred [ modelIndex ]
                INPUT: hasNext_i
                    nSymbols = 2
                    EMBED: Generate Alphabet
                    STORE: next [ modelIndex ]
                k = k + 1
```


© ISO/IEC 2014 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 23002-4_2014_Amd 1_2014