NEXT SOFTWARE – DATA STRUCTURES

Fraunhofer
HHI

Motivation

HM Shortcomings

- Complicated data structures
  - Z-index
  - Ambiguous data model (TComDataCU)
- Bad code readability
  - Complicated memory operations are intermixed with general data flow
  - Lack of data and logic encapsulation
- Complicated extensibility
  - Data structures were designed with strict assumptions (e.g. square blocks)
  - How many ideas were discarded because of erroneous implementation?

Fraunhofer HHI

Goals

NextSoftware design principles

- Simple unambiguous data model
  - Modern OO-design, yet fast and sleek
- Global pixel addressing
  - Elimination of Z-index usage
  - Elimination of 2-level (CTU→Z-Index) signal addressing
- Encapsulation of trivial operations (e.g. memory operations)
  - Allows for better readability of general flow
- Better code readability
  - Allowing for easier extensibility

HM – Model

- TComDataCU
  - Stores coding information
  - Might represent a CU or CTU (containing multiple CU's)
  - Basically a map of image position to coding information
- TComTU
  - An object allowing for easier TU structure navigation
  - Contains no actual data, but rather wraps around *TComDataCU*
- TComPicYuv
  - Stores the video signals

NextSoftware – Model

Navigation

- Size, Position, Area (Position + Size)
  - Represent basic 2D navigation information
- CompArea
  - *Area* in a given component (block)
- UnitArea
  - Represents an area in a multi-channel signal
  - A set of blocks describing a composition of co-located components

Fraunhofer HHI

NextSoftware – Model

Signal Storage

- AreaBuf
  - Describes the memory layout of a 2D signal in linear memory
  - Contains simple operations (copy, fill etc.)
- UnitAreaBuf
  - Describes the memory layout a multi-component 2D signal in linear memory
  - Contains simple operations (copy, fill etc.)
- PelStorage
  - A *UnitAreaBuf* which also allocates its own memory

NextSoftware – Model

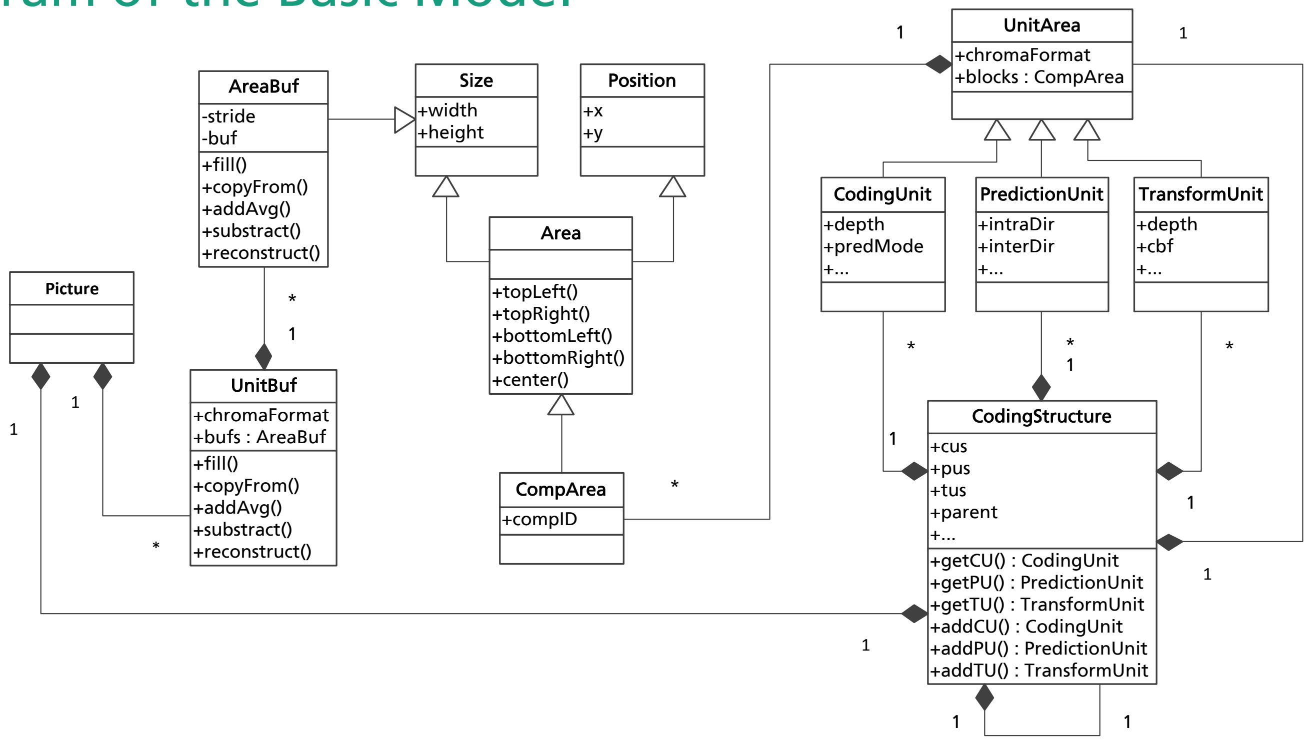
Coding Information

- Picture
  - Contains input and output signals as well as metadata (slice info etc.)
- CodingUnit, PredictionUnit, TransformUnit
  - Single object for a single unit
  - Contain the corresponding information
  - Include the location information (derived from *UnitArea*)
- CodingStructure
  - Manages the *CodingUnit* and co., links them with the picture
  - Contains additional functionalities for top-down RD-search

Fraunhofer HHI

NextSoftware – Model

UML-Diagram of the Basic Model



HM – NextSoftware Model Equivalencies

HM	NextSoftware
Z-index, (CTU)-RS-address, Depth	Position, Size, (Comp)Area, UnitArea
TComDataCU	CodingUnit, PredictionUnit, TransformUnit Operations in CU, PU, TU namespaces CodingStructure
TComTU	Partitioning is governed by Partitioner
TComPicYuv	Picture
TComPic	Picture
TComYuv	UnitAreaBuf

NextSoftware – Detailed Description

CodingStructure Basics

- Contains *CodingUnit* etc. objects and maps them to the picture
- A TComDataCU replacement, but globally allocated
  - Top-level *CodingStructure* contains all CU's, PU's and TU's in the frame
  - Sub-level *CodingStructure* contains a representation of a specific *UnitArea*
- After creation it's empty and needs to be filled
  - addCU/PU/TU methods create and map the specific object
  - getCU/PU/TU fetches the specific objects addressed using global *Position*
- Dynamically allocates the required resources
  - Uses *dynamic_cache* for increased performance

Fraunhofer HHI

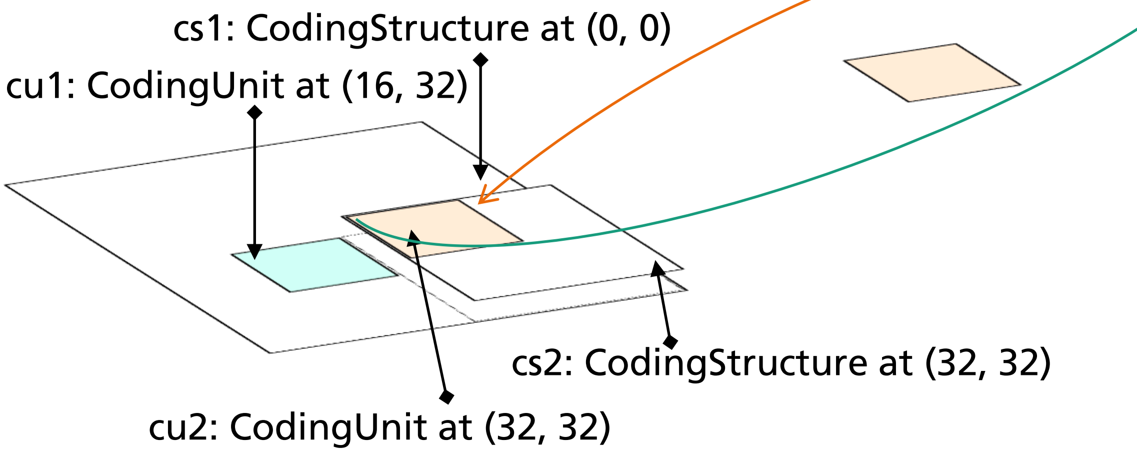
NextSoftware – Detailed Description

RD-Search with CodingStructure

- Designed for Top-Down approach
- Allows for local test encoding with “transparent” global context
  - Follows the well known best-temp scheme with up-propagation
- Hierarchically cascaded
  - A *CodingStructure* is set up to represent a local *UnitArea*
  - Calls outside of this *UnitArea* are forwarded to the parent *CodingStructure*
- Parent nodes are not aware of the children nodes
  - Best candidates need to be propagated to the parents

NextSoftware – Detailed Description

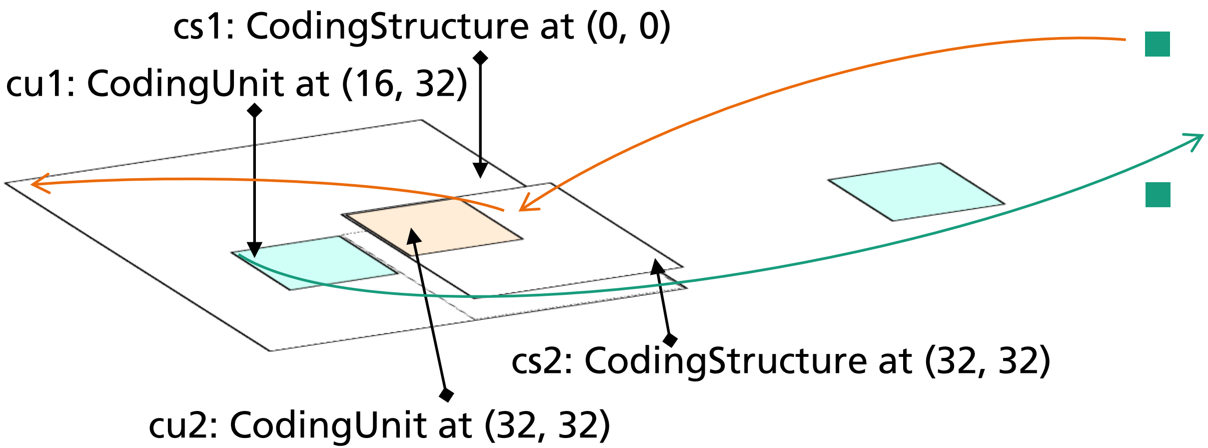
Hierarchical Cascading with CodingStructure



- cs2.getCU({32, 32})
  - returns *cu2*
- cs2.getCU({16, 32})
  - returns *cu1*
- cs1.getCU({32, 32})
  - returns *nullptr*

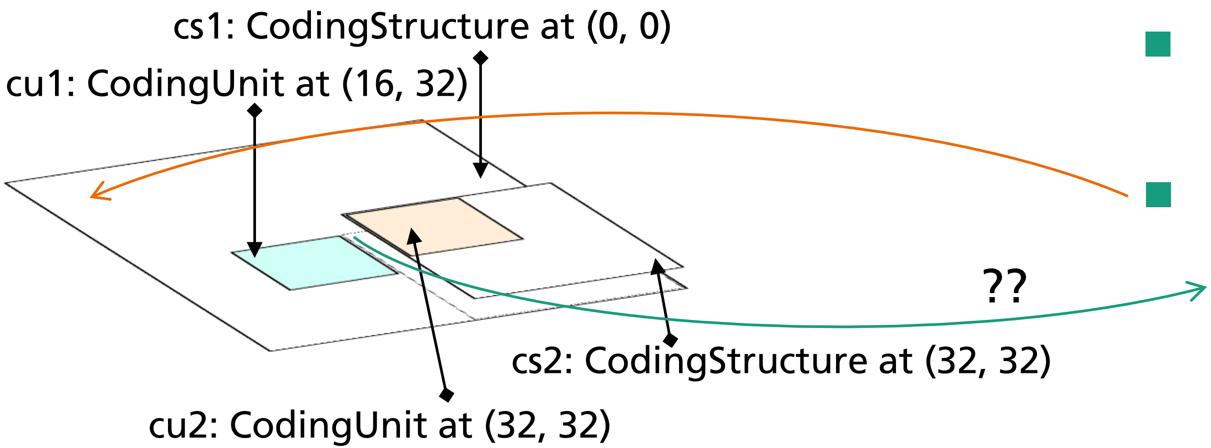
NextSoftware – Detailed Description

Hierarchical Cascading with CodingStructure



- cs2.getCU({32, 32})
  - returns *cu2*
- cs2.getCU({16, 32})
  - returns *cu1*
- cs1.getCU({32, 32})
  - returns *nullptr*

NextSoftware – Detailed Description
Hierarchical Cascading with CodingStructure



- cs2.getCU({32, 32})
  - returns *cu2*
- cs2.getCU({16, 32})
  - returns *cu1*
- cs1.getCU({32, 32})
  - returns *nullptr*

NextSoftware – Detailed Description

Partitioner

- A simple class governing splitting (CU and TU, quad-tree and possibly others)
- Modelled as a stack – new splits are created as levels on the currently processed area
- For HEVC
  - Contains accessors for current split info (*partitioner.curr**)
  - Depth (CU, TU) as well as the actual current *UnitArea*
- For QTBT and further (additionally to HEVC-features)
  - Allows to set split restrictions (e.g. constraint splits at a certain level)
  - Allows to perform split plausibility checks (*canSplit*)

NextSoftware – Detailed Description
Data Ownership

- Each piece of data is owned by some object, which needs to allocate and release it
- *Picture*
  - Owned by *EncLib* or *DecLib*
  - Owns signal buffers, *Slice* objects, *SEI* messages and *TileMap*
- *AreaBuf, UnitBuf*
  - Do not own any data
- *PelStorage*
  - Might own the buffers (depends if *create* or *createFromBuf* used for creation)
  - Owned data is stored in m_origin member

Fraunhofer HHI

NextSoftware – Detailed Description

Data Ownership

- *CodingStructure*
  - Top-Layer: owned by *Picture*
    - Links to signal buffers of *Picture*, does not own them
  - Other (temporary in RD-Search): owned by *EncCu* or *IntraSearch*
    - Contains own signal buffers, owns them
  - Always owns buffers describing the structure and layout (not signal)
  - Owns transformation coefficient buffers
  - Does not own *CodingUnit* etc., only links to them through *dynamic_cache*

NextSoftware – Detailed Description

Data Ownership

- *CodingUnit, PredictionUnit, TransformUnit*
  - Owned by *dynamic_cache* – objects need to be acquired by *get* and freed by *cache*
- *TransformUnit*
  - Does not own transformation coefficient buffers
  - Links to buffers from *CodingStructure*
- *dynamic_cache*
  - Top-Level cache is global (dynamically allocated on runtime and freed on exit)
  - RD-search cache is owned by *EncCu* and *IntraSearch*

Fraunhofer HHI

NextSoftware – Code Snippets

Iterate over all TUs in a CU (inter decompression example)

- Iterating over PUs works similar with *traversePUs(…)* call
- *firstTU* and *firstPU* can be used as fast accessors if the CU only has one TU/PU

```
// call xDecodeInterTU for all TUs in the CU described by cu
for( auto& currTU : CU::traverseTUs( cu ) )
{
    xDecodeInterTU( currTU, compID );
}
```


Fraunhofer HHI

NextSoftware – Code Snippets

Iterate over all CUs in a specific area (CTU decompression example)

```
// iterate over all CUs which are contained in the area described by ctuArea
for( auto &currCU : cs.traverseCUs( ctuArea ) )
{
    // decompress CU
    xDecompress( currCU );
}
```

NextSoftare – Common Problems

Where is the motion information stored?

- Initially with HEVC, minimal resolution for Motion Vector storing was the PU
- New prosed tools for h.266 standard break this convention (e.g. VCEG-AZ10)
  - Sub-PU resolution for motion vector information is needed
  - Storing sub-PUs as PUs would break the logical purpose of a PU
  - Solution: additional buffer for sub-PU resolved motion information
  - Example:

```
Mv mvL0 = cs.getPU( pos )->mv[0];           // obsolete low-res call
Mv mvL0 = cs.getMotionInfo( pos ).mv[0]; // new next-style high-res call
```

- Old call still allowed, might provide sub-resolution results
- *PU::spanMotionInfo* sets up the buffer

Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute, HHI

WE PUT SCIENCE INTO ACTION.

Contact:

Adam Wieckowski
adam.wieckowski@hhi.fraunhofer.de
-833

Einsteinufer 37
10587 Berlin

NextSoftware – Reference

Size, Position, Area

- Size
  - width, height : UInt – describe the size of a rectangle
- Position
  - x, y: Int – describe the 2D position of a point
- Area : Size, Position
  - a rectangle of a specific *Size* located at a specific *Position*
- CompArea : Area
  - an *Area* within a specific component (*compID*) of a multi-component signal

NextSoftware – Reference

UnitArea, CodingUnit, PredictionUnit, TransformUnit

- UnitArea
  - blocks [0..N-1]: *CompArea*
  - a multi-component compound consisting of N blocks
- CodingUnit : UnitArea
  - Describes how the area described by this *UnitArea* is coded
- PredictionUnit : UnitArea
  - Describes how the prediction signal for this *UnitArea* is to be generated
- TransformUnit : UnitArea
  - Describes how the transformation coding for this *UnitArea* is to be applied

NextSoftware – Reference

AreaBuf

- AreaBuf<T> (defined for *Pel* and *TCoeff* as *PelAreaBuf* and *CoeffAreaBuf*)
  - at(x, y) – returns the value of the signal at position (x,y)
  - bufAt(x, y) – returns the raw pointer to the buffer at position (x,y)
  - subBuf(x, y, w, h) – returns a *AreaBuf* describing an area offset by (x,y) of size (w,h)
  - fill(val) – fills the specified area with the defined value
  - copyFrom(other) – copies the contents from the other area
  - substract, addAvg, reconstruct, removeHighFreq – methods replacing the functionalities of TComYuv
- UnitBuf<T> (similar interface to *AreaBuf*)
  - bufs [0..N-1]: AreaBuf – contains the signal descriptions for different components

Fraunhofer HHI

NextSoftware – Reference

CodingStructure Basics

- CodingStructure
  - area: UnitArea – describes which area in the picture the *CodingStructure* spans
  - addCU(UnitArea) – creates and locates a *CodingUnit* spanning the *UnitArea*
  - getCU(Position) – returns the *CodingUnit* located at the specified *Position*
    (analogue interfaces exist for *TransformUnit* and *PredictionUnit*)
  - setDecomp(CompArea) – sets the specified *CompArea* as reconstructed
  - setDecomp(UnitArea) – analogue multi-channel operation
  - isDecomp(Position) – tells if the reco. signal for the *Position* has been generated

NextSoftware – Reference

RD-Search with CodingStructure

- CodingStructure
  - initStructData(…) – clears all currently contained data (signals and coding info)
  - initSubStructure(…) – links a new *CodingStructure* at the bottom of the hierarchy
  - useSubStructure(…) – copies the coding data from a sub-structure
  - copyStructure(…) – copies to coding data from another structure, does not rely on a parent-child binding