a]Fanyu Ran b]Yang Zhou c]Yifei Zhou [a]Student No.8759122, University of Ottawa [b]Student No.8657223, University of Ottawa [c]Student No.8635051, University of Ottawa

The software architecture design is guided by the syntax of H.264 bitstream which is shown as below.

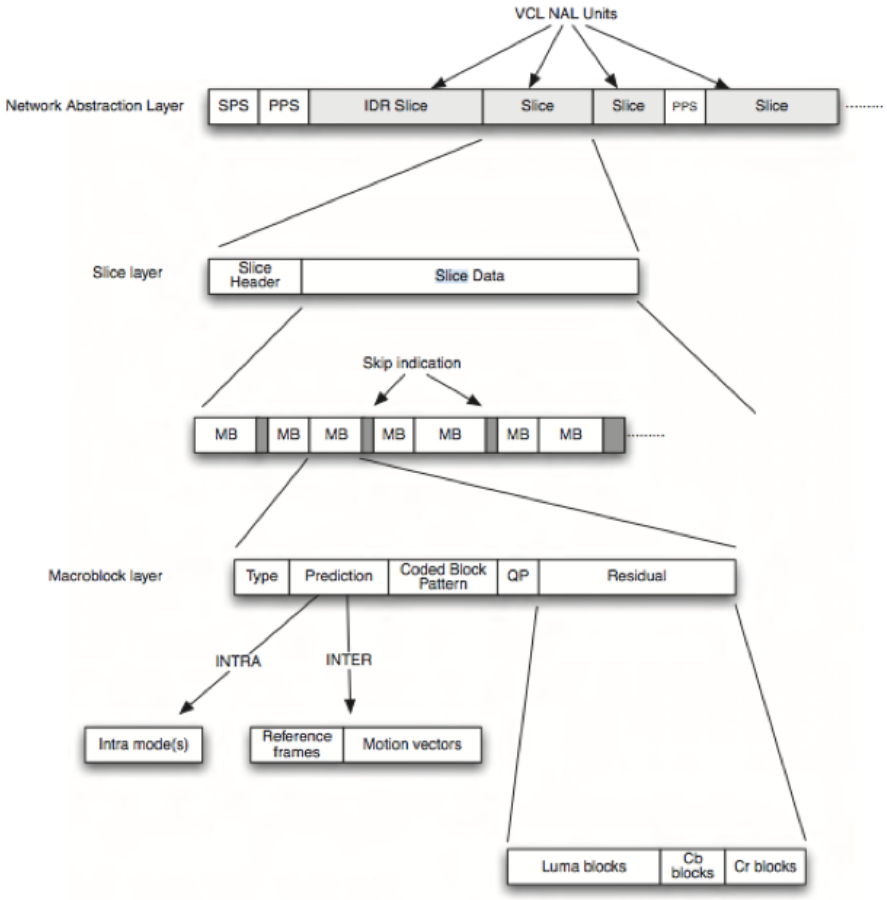


Figure 1. H.264 bitstream syntax

The project is implemented mainly in object oriented paradigm. Primary concepts of H.264 are abstracted as objects. A simplified Unified Modeling Language(UML) diagram of the software architecture is shown as below:

The functionalities of these objects are described here.

## 0.1 H264Bits

H264Bits object stores raw bitstream data and entropy decoding methods. Each slice object contains a reference to the singleton H264Bits object, so the binary data can be decoded and transfered to its associated object.

## 0.2 SPS

SPS object contains sequence parameter set related parsing logic and result. The parsing methods read parameters from bitstream and store the results as instance variables of the object. So that these parameters can be shared later. There's usually only one SPS object in a sequence.

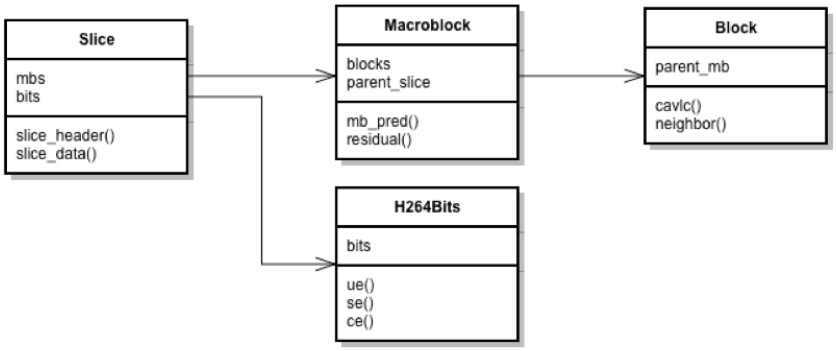


Figure 2. Software architecture UML diagram

## 0.3 PPS

PPS object contains picture parameter set related parsing logic and result. This object has behavior similar to SPS except that there can be multiple PPS objects in a sequence.

## 0.4 Slice

Slice object contains slice header parsing logic, slice related parameters and macroblock objects. Slice-related paramters from slice header segments are parsed and stored here. Arbitrary Macroblock objects are contained in this object which is identical to H.264's layer logic.

## 0.5 Macroblock

Macroblock object contains macroblock parsing logic and Luma/Chroma block objects Macroblock-related parameters are decoded and CAVLC entropy decoding process is invoked here. The entropy decoded results are stored as Block objects inside this object.

## 0.6 Block

Block object contains Luma/Chroma block's transform coefficients and related decoding methods. The object also has block id related information stored in it.

Each object also contains its parent object pointer to share data between each other. So accessing sequence parameter set from a luma block object can be done like this:

```
one_luma_block.mb.slice.sps.some_parameter
```

There are another two modules for Intra frame decoding: idct and intra_pred. These modules are collections of functions for inverse DCT (and dequantization) and intra prediction.

## 0.7 IDCT

This module contains amount of functions to perform dequantization and inverse DCT on block coefficients levels. Related argument tables are also stored here as module attributes.

## 0.8 Intra Prediction

This module contains amount of functions to generate intra prediction for blocks.