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

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

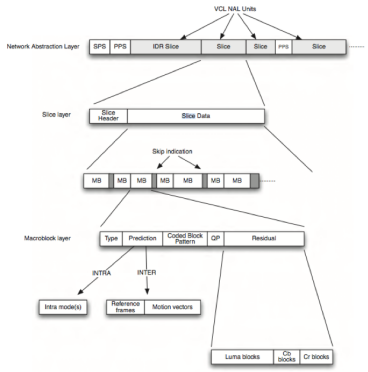

Figure 1. Figure captions are used to describe

The program in implemented mainly in object oriented paradigm and concepts of H.264 are abstracted as objects. The primary 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.

```
def u(self,n) # read n bits as a unsigned integer
def f(self,n) # read n bits as a signed integer
def ue(self) # read bits with variable lenght as unsigned exponential Golomb code
def se(self) # read bits with variable lenght as signed exponential Golomb code
def ce_coeff_token(self, nC) # decode CAVLC coefficient token with nC
def ce_total_zeros(self, tzVlcIndex, maxNumCoeff) # decode number of zeros in coefficients levels
```

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.

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
```

A simplified Unified Modeling Language(UML) diagram of the software architecture is shown as below:

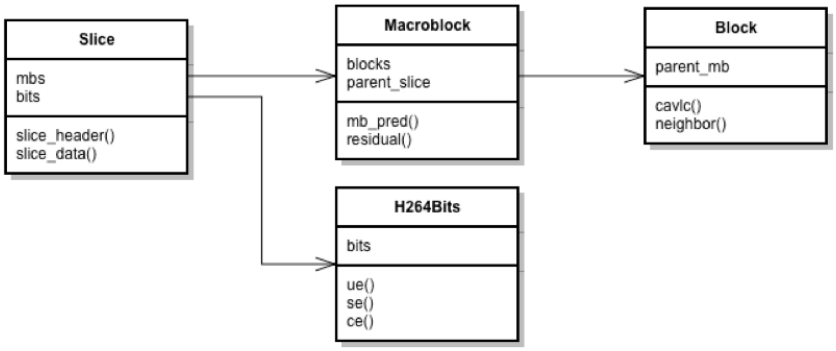

Figure 2. Figure captions are used to describe