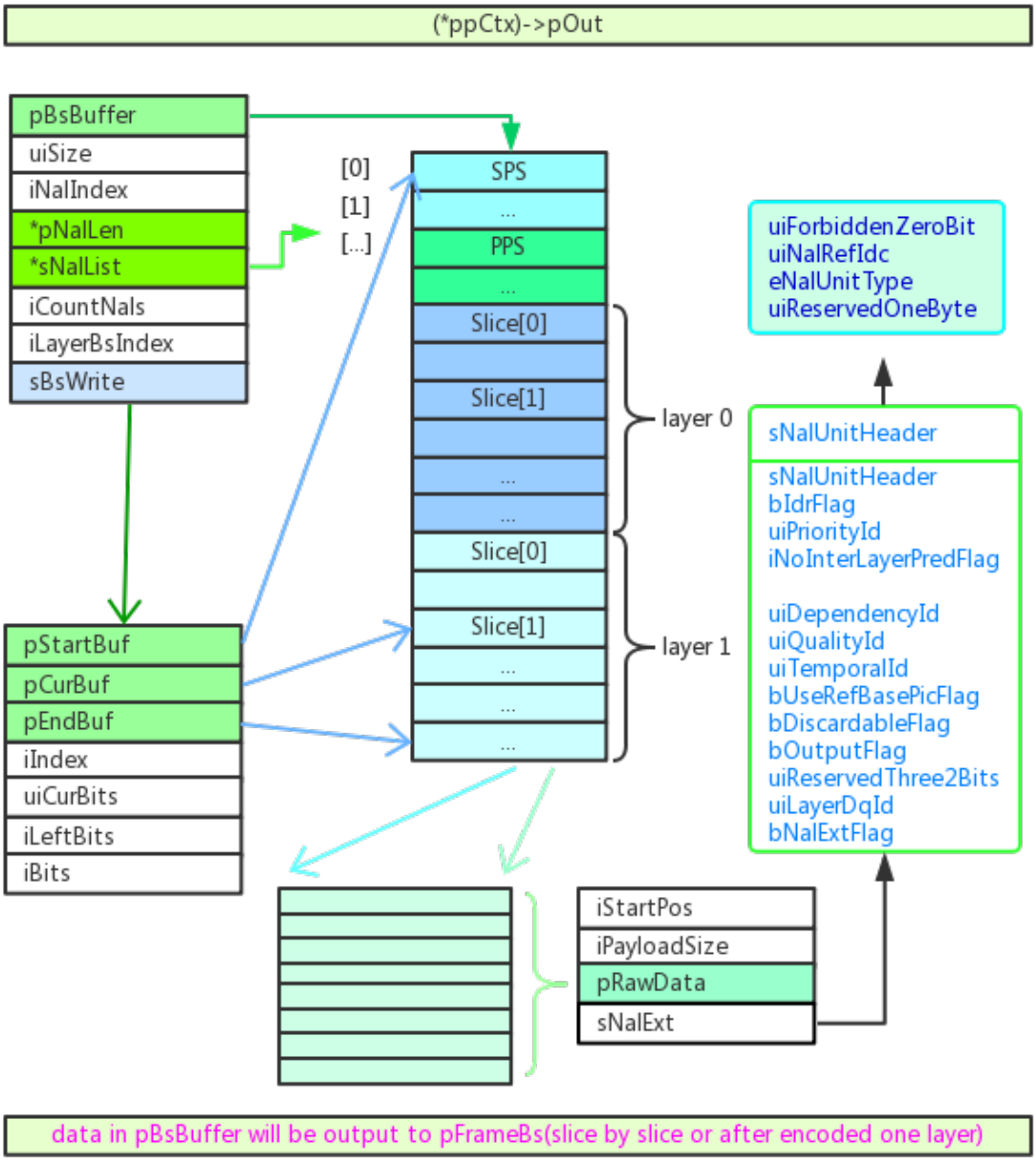
Slice buffer design

Table of Contents

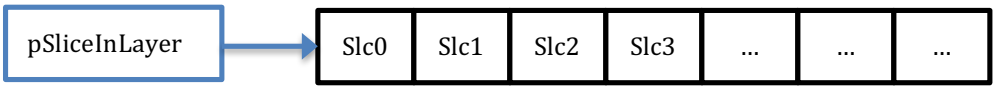


1. Overview for new design

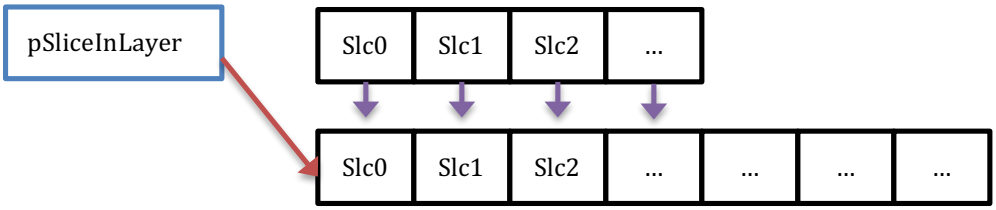
.1. Origin design

SSlice* pSliceInLayer // slice buffer for all slices in layer

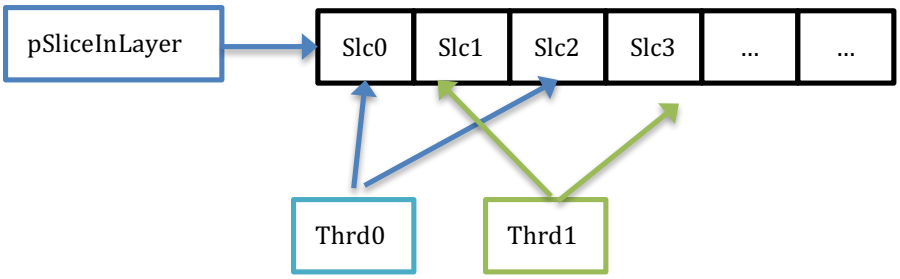
1.1.1. Single thread



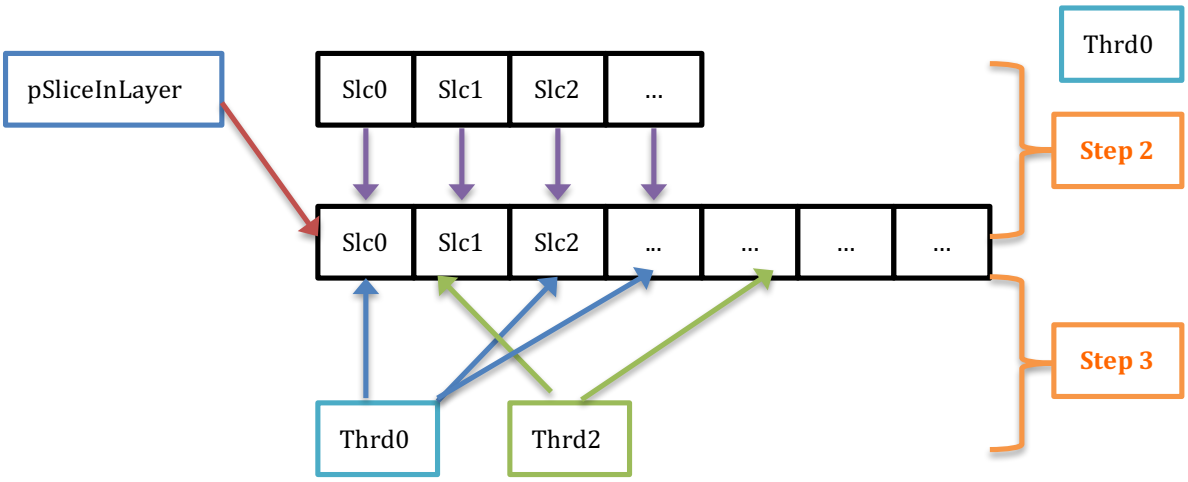
reallocate when current slice index larger than max slice num



1.1.2. Multi thread



reallocate when current slice index larger than max slice num ,
--step 0: thread[0] detect that current slice index larger than max slice num;
--step 1: thread[0] need to wait thread[1] completed current slice encoding task
--step 2: thread[1] stop slice encoding and thread[0] reallocate slice buffer,
--step 3: thread[0]/thread[1] start to encode new slice

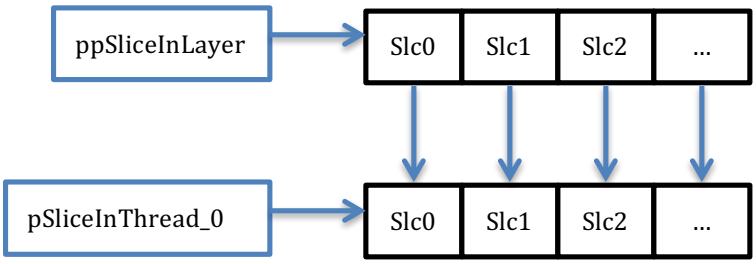


i.

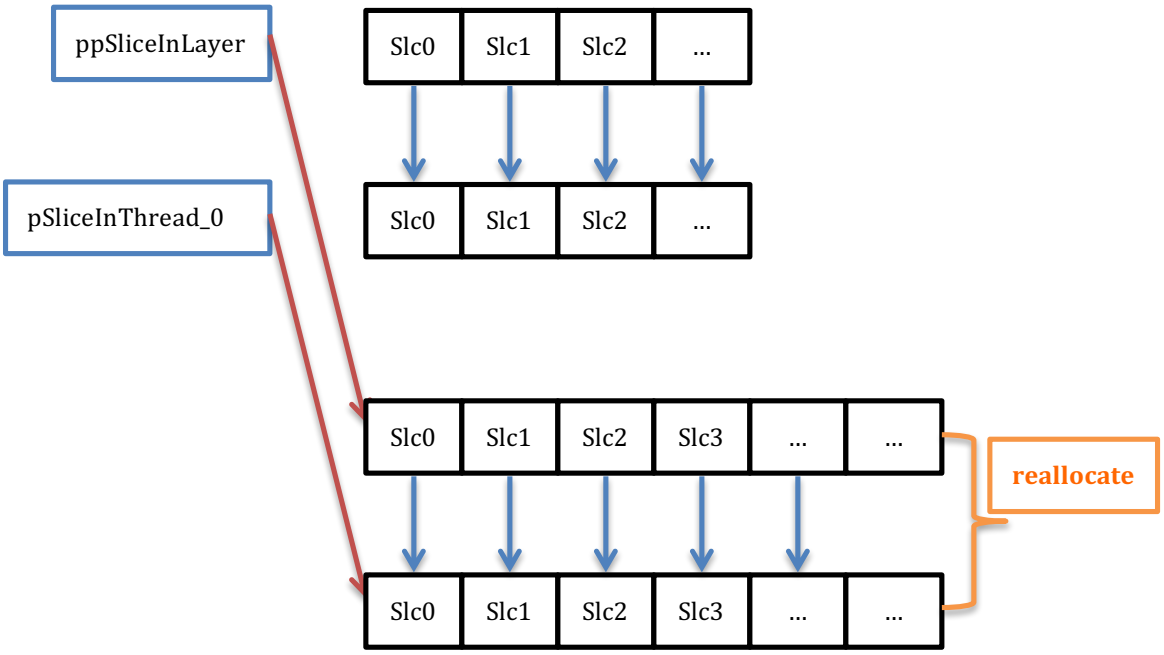
1.2 New design in review

```
SSlice** ppSliceInLayer;    // point to actual slice buffer
typedef struct TagSliceThreadInfo {
    SSlice*                 pSliceInThread;// slice buffer for multi thread,
                                // pSliceInThread[0] for single thread
    int32_t                 iMaxSliceNum;
     int32_t                 iCodedSliceNum;
}SSliceThreadInfo;
```

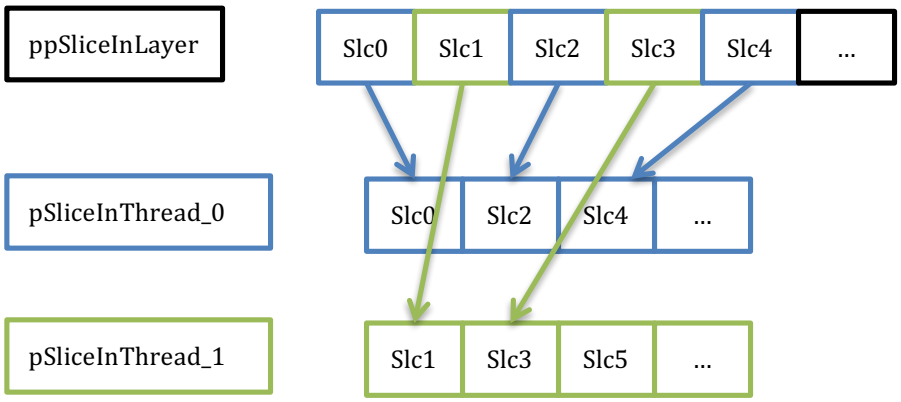
1.2.1 Single thread



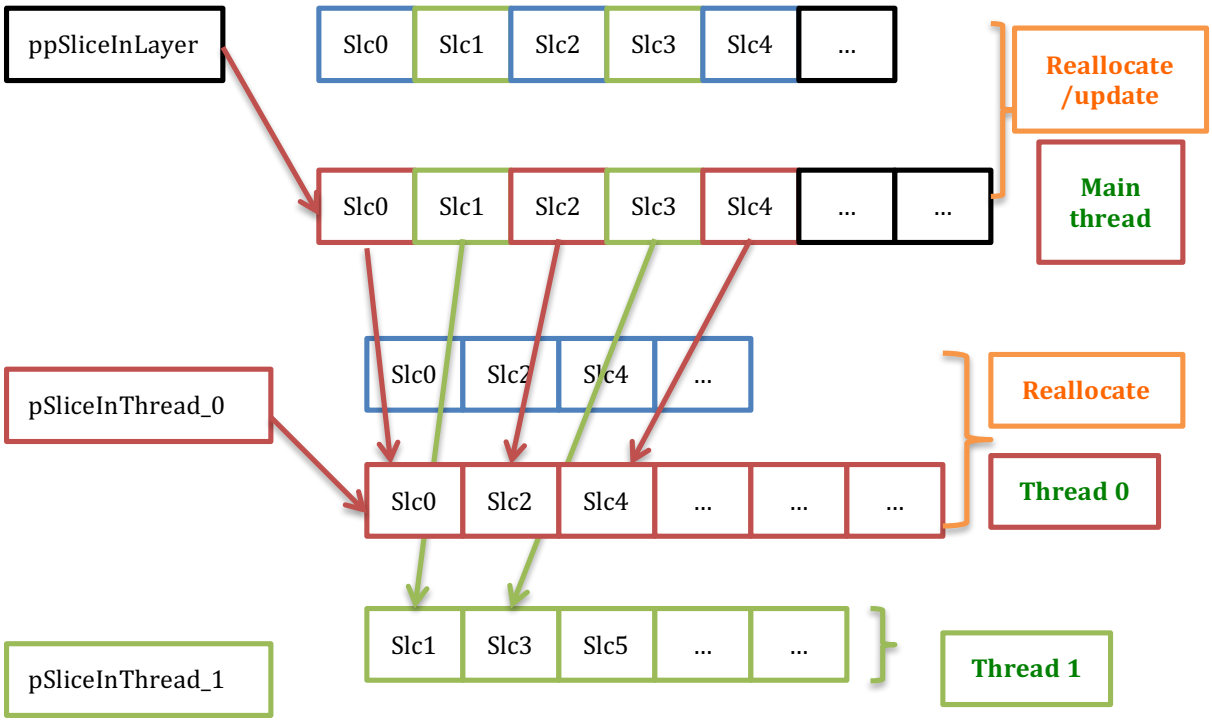
reallocate when current slice index larger than max slice num



1.2.2. Multi-thread



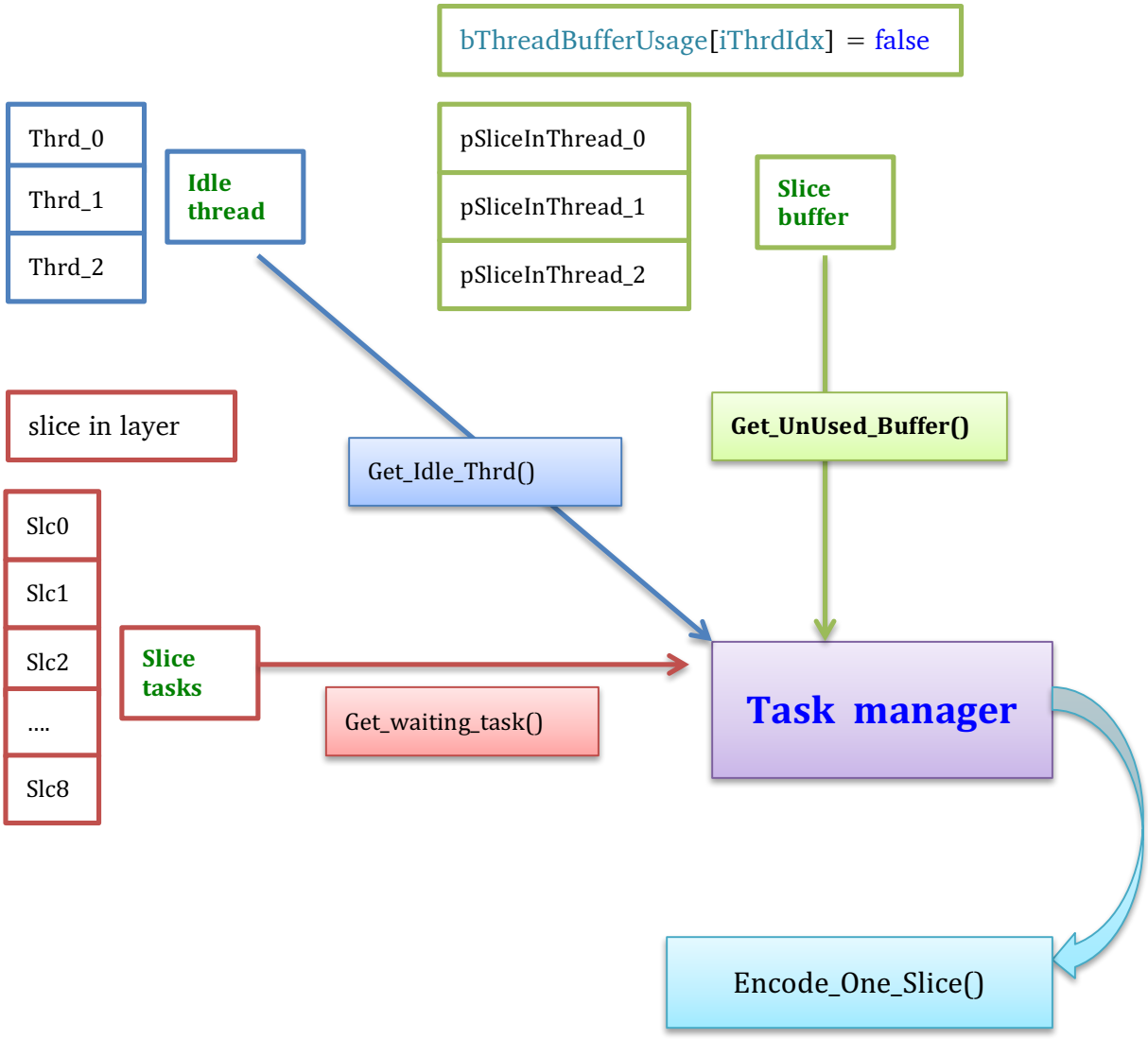
for reallocate, each thread will do it independently, and will update ppSliceInLayer by ***main thread*** when all slices in layer are encoded.



2. SLICE BUFFER AND THREAD

2.1 Before encoding one layer

the status of slice buffer and thread :
example:
thread: 3 threads
slices: 9 slices in layer



Query thread buffer function:

```
int32_t CWelsSliceEncodingTask::QueryEmptyThread (bool* pThreadBsBufferUsage) {
  for (int32_t k = 0; k < MAX_THREADS_NUM; k++) {
    if (pThreadBsBufferUsage[k] == false) {
      pThreadBsBufferUsage[k] = true;
      return k;
    }
  }
  return -1;
}
```

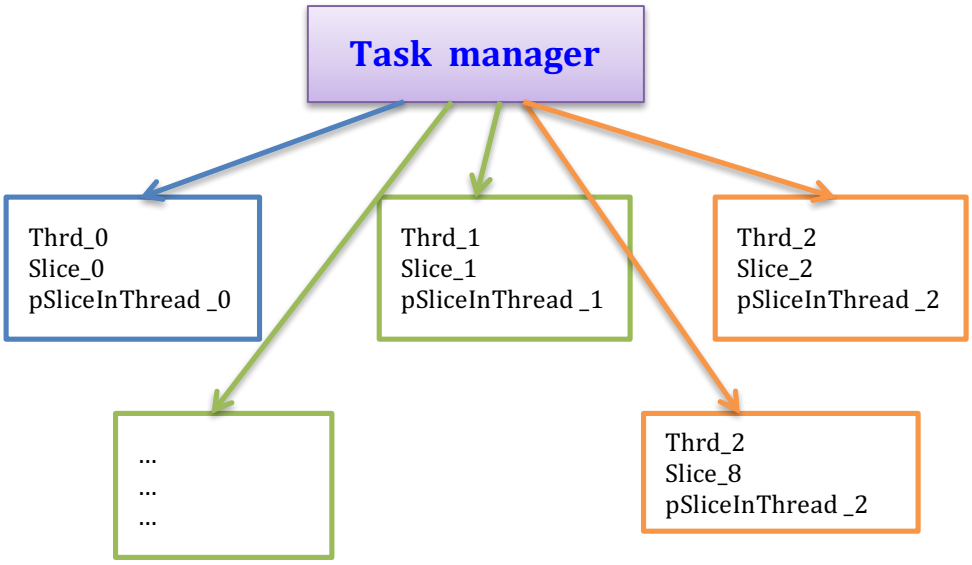
Example:

k=1, means that encode slice task using pSliceInThread_0 as slice buffer

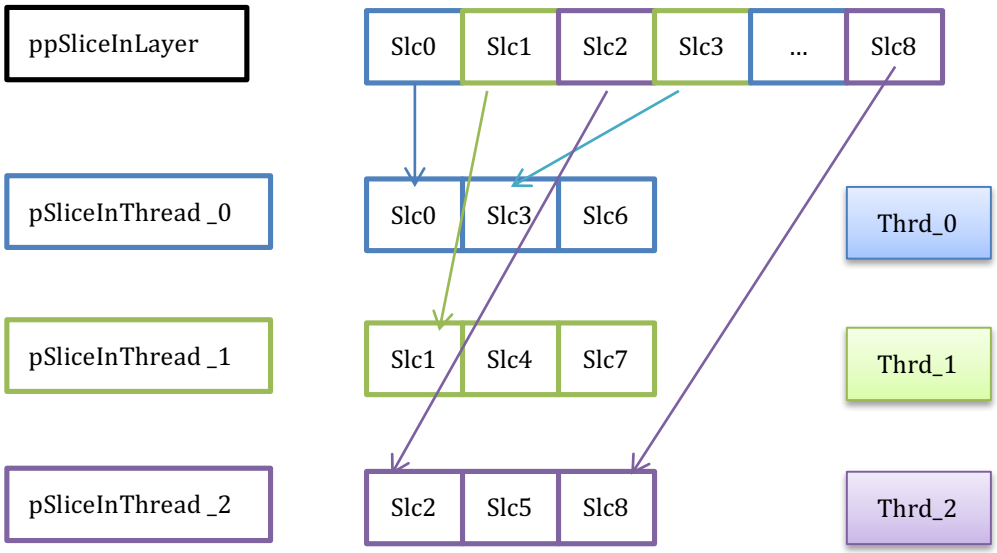
Query IDLE thread for slice task function:

```
CWelsTaskThread*   CWelsThreadPool::GetIdleThread() {
  CWelsAutoLock cLock (m_cLockIdleTasks);

  if (m_cIdleThreads->size() == 0) {
    return NULL;
  }

  CWelsTaskThread* pThread = m_cIdleThreads->begin();
  m_cIdleThreads->pop_front();
  return pThread;
}
```

2.2. Normal case for thread index and slice buffer index



Final map for thread index and slice buffer index, slice index

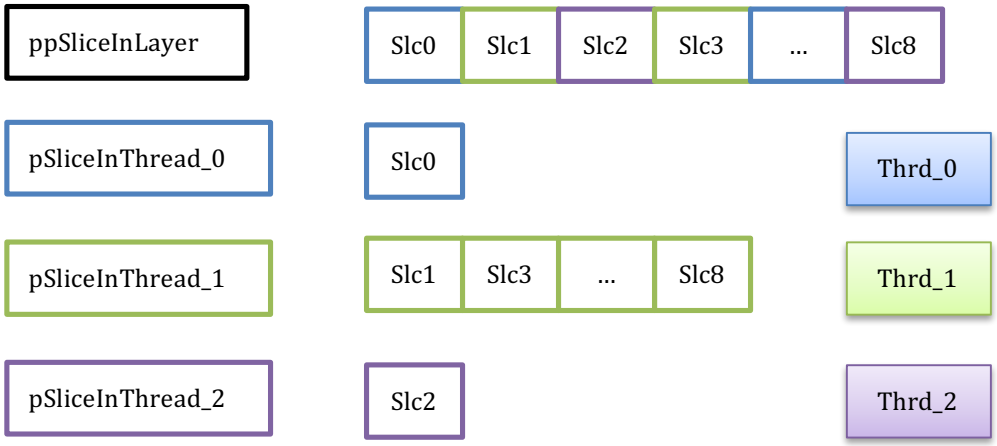


2.3. Slice num not the same among threads

case 1:

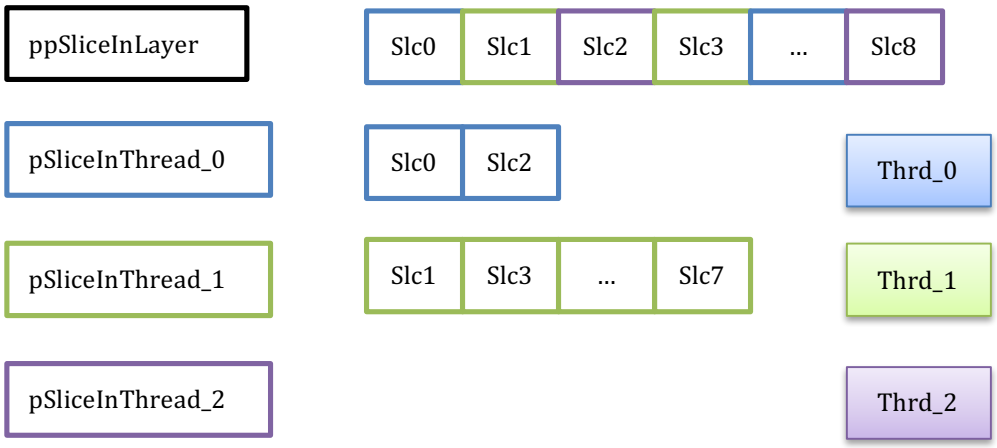
case 2:

encode time for Slc0 and Slc_2 are longer than Slc1, Slc3~Slc8



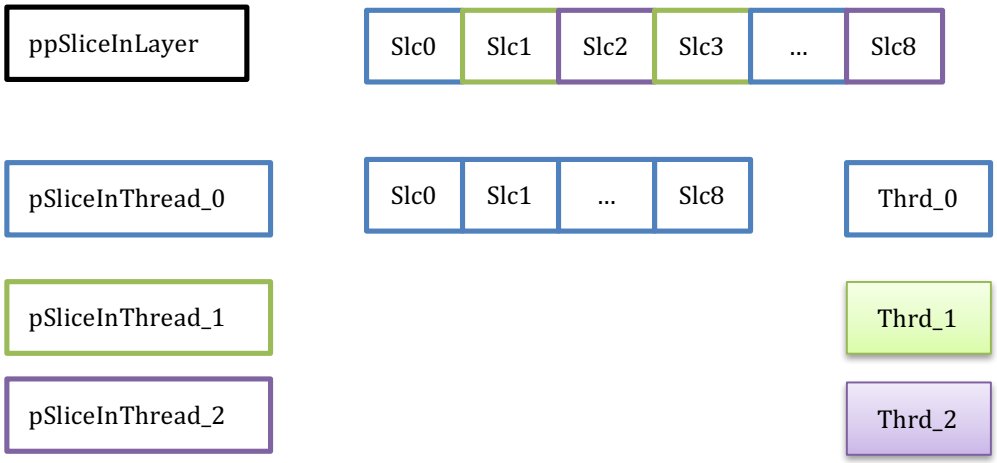
Corner case 1, no encoded slice for one thread:

Thread_0 and Thread_1 are fast enough and always get CPU resource,
Thread_2 is under waiting status/IDLE status.



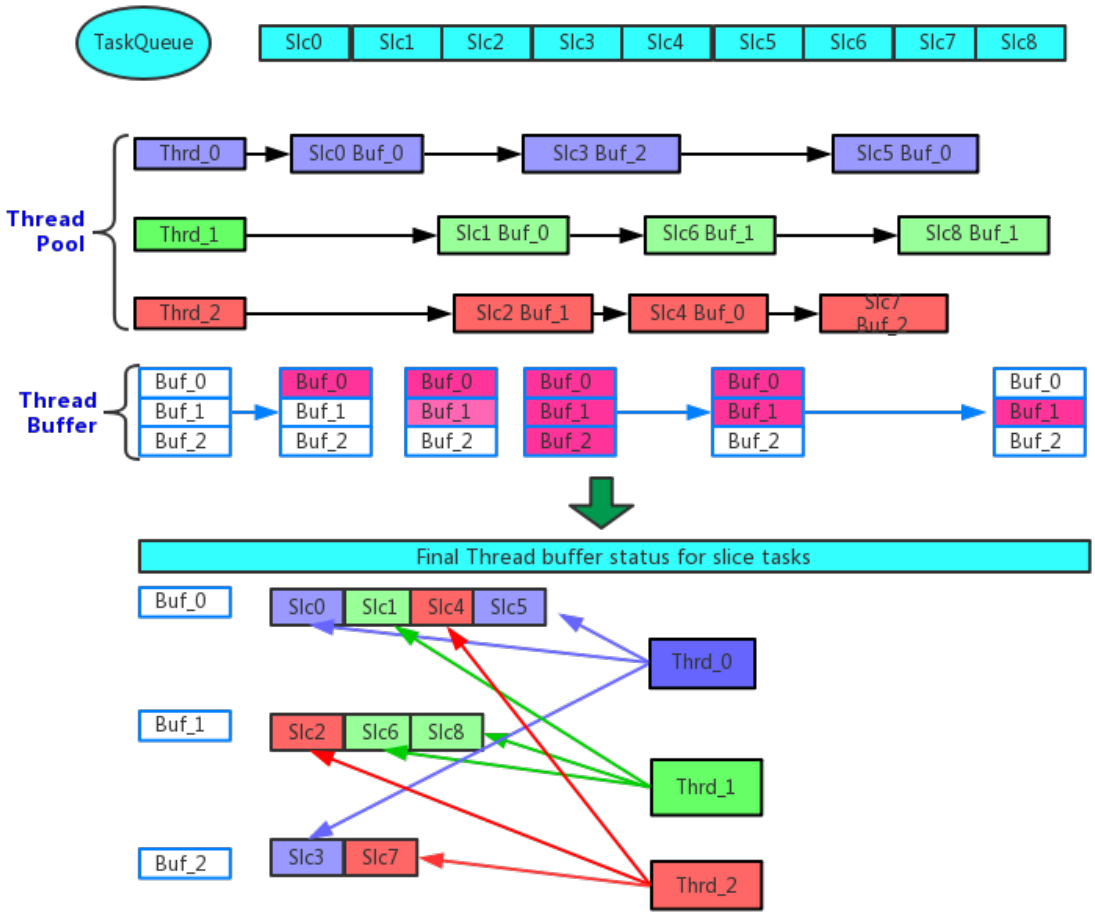
Corner case 2, all slices encoded by one thread :

Thread_0 is fast enough and always get CPU resource,
Thread_1 and Thread_2 are under waiting status/IDLE status.

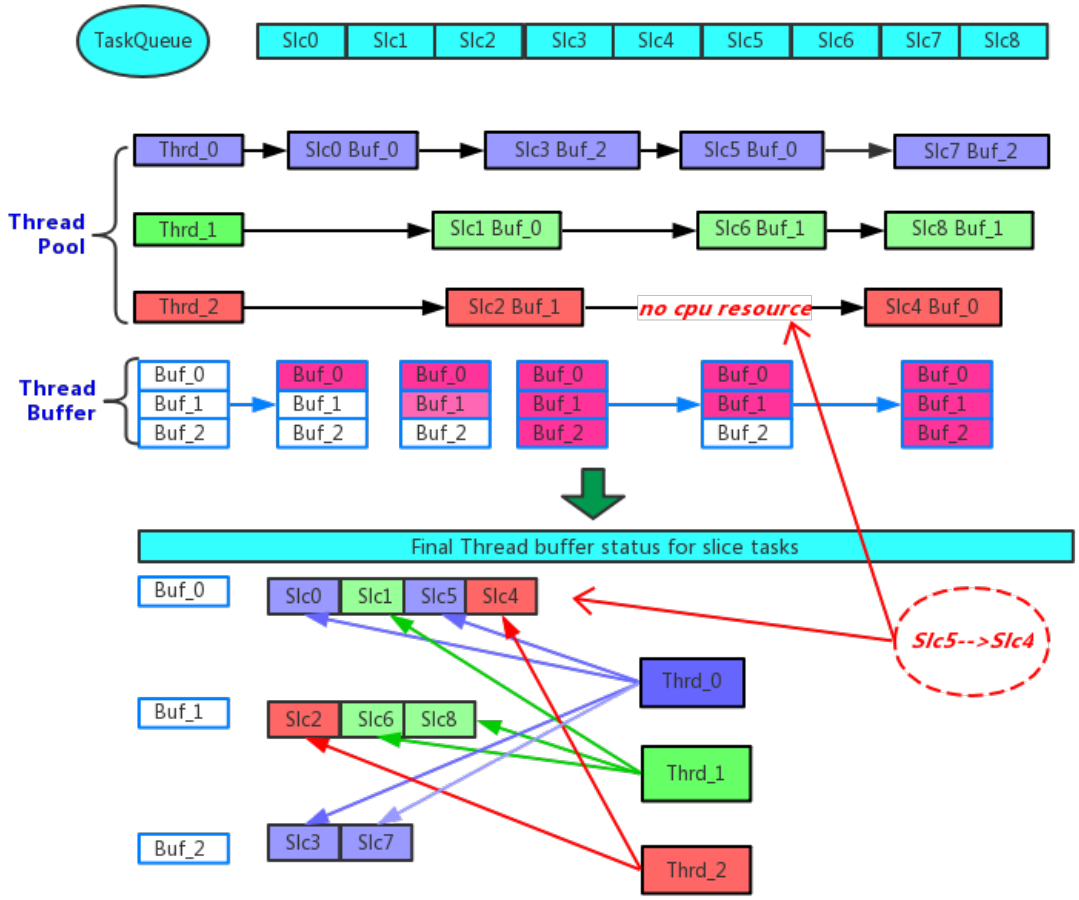


2.4. Different thread index in the same slice buffer

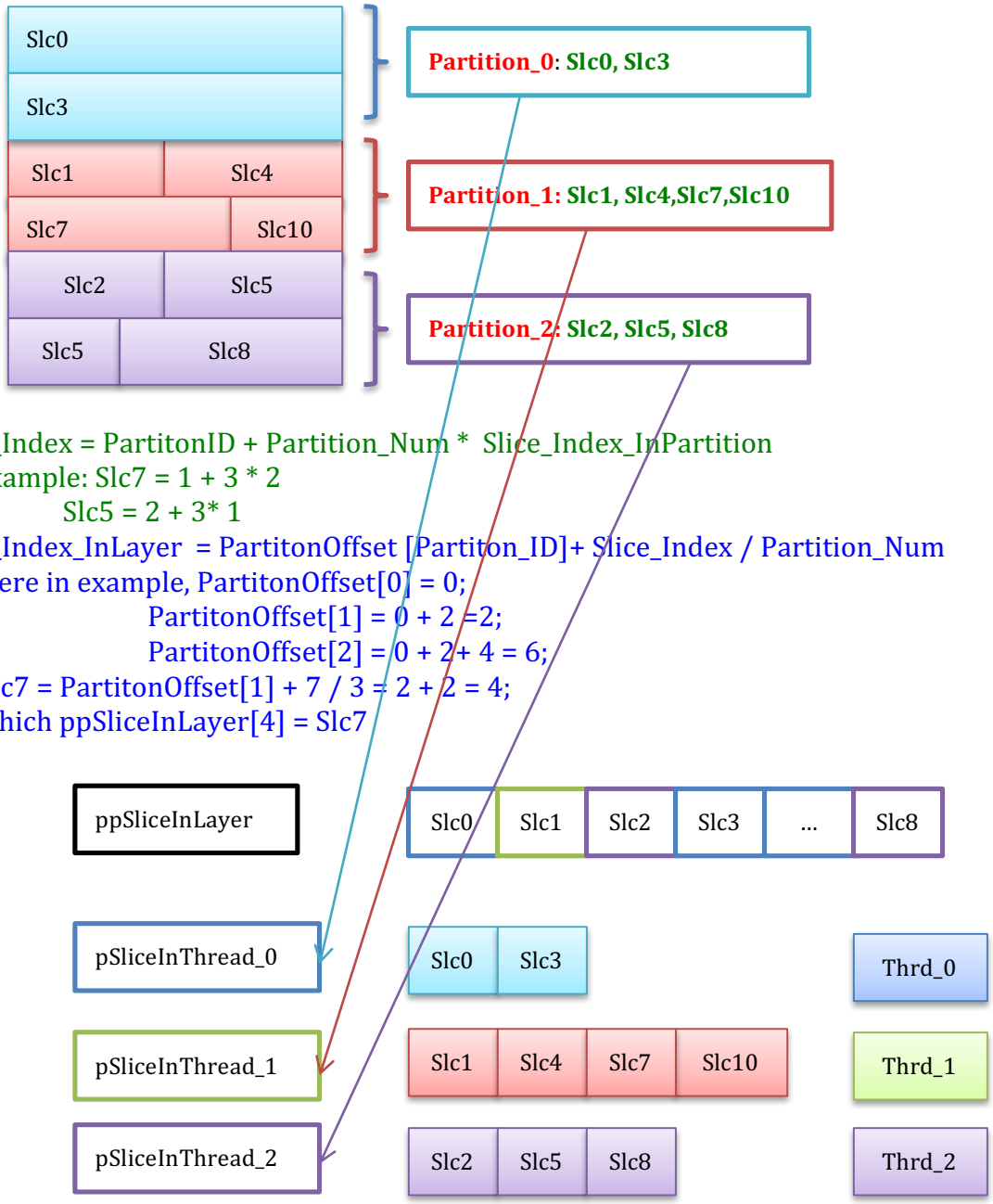
Timing diagram for Thread—SlcTask—ThreadBuffer



2.5. Slice index out-of-order case

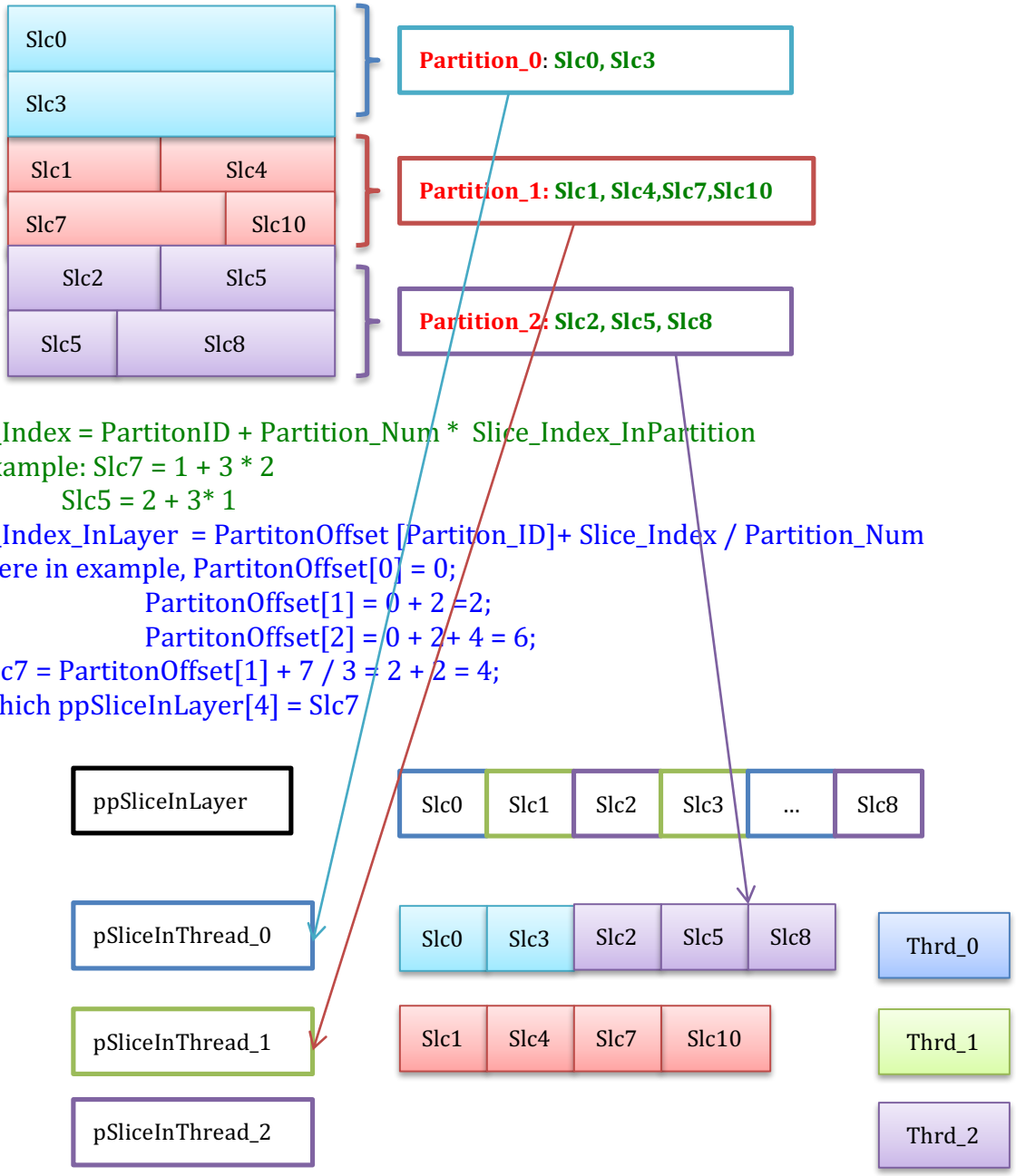


2.6. Dynamic slice mode case1



2.7. Dynamic slice mode case2

Thread_0 encoded two partitions
while no partition for Thread_2



Slice_Index = PartitonID + Partition_Num * Slice_Index_InPartition

Example: Slc7 = 1 + 3 * 2

Slc5 = 2 + 3* 1

Slice_Index_InLayer = PartitonOffset [Partiton_ID]+ Slice_Index / Partition_Num

Here in example, PartitonOffset[0] = 0;

PartitonOffset[1] = 0 + 2 =2;

PartitonOffset[2] = 0 + 2+ 4 = 6;

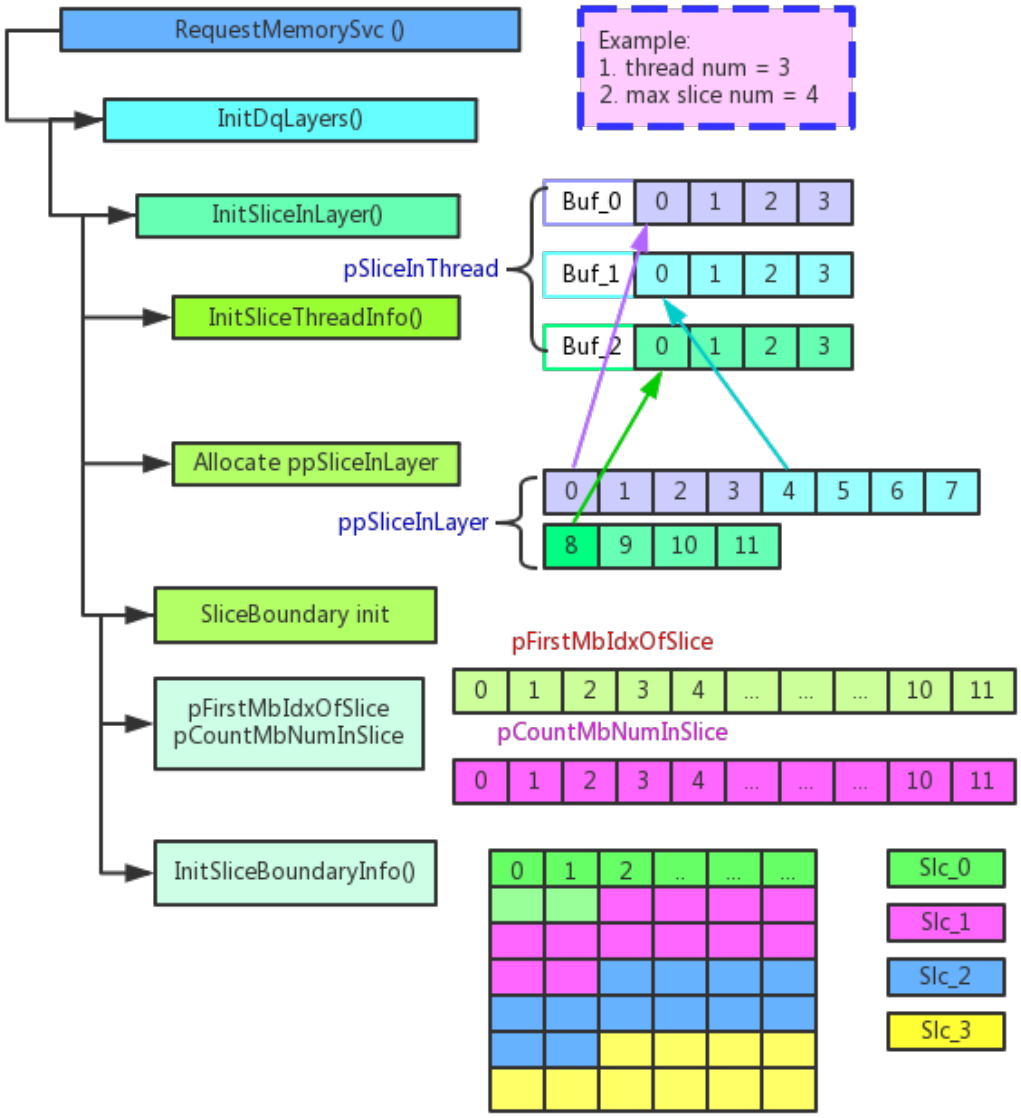
Slc7 = PartitonOffset[1] + 7 / 3 = 2 + 2 = 4;

Which ppSliceInLayer[4] = Slc7

3. Slice Buffer Init/Update/Reorder

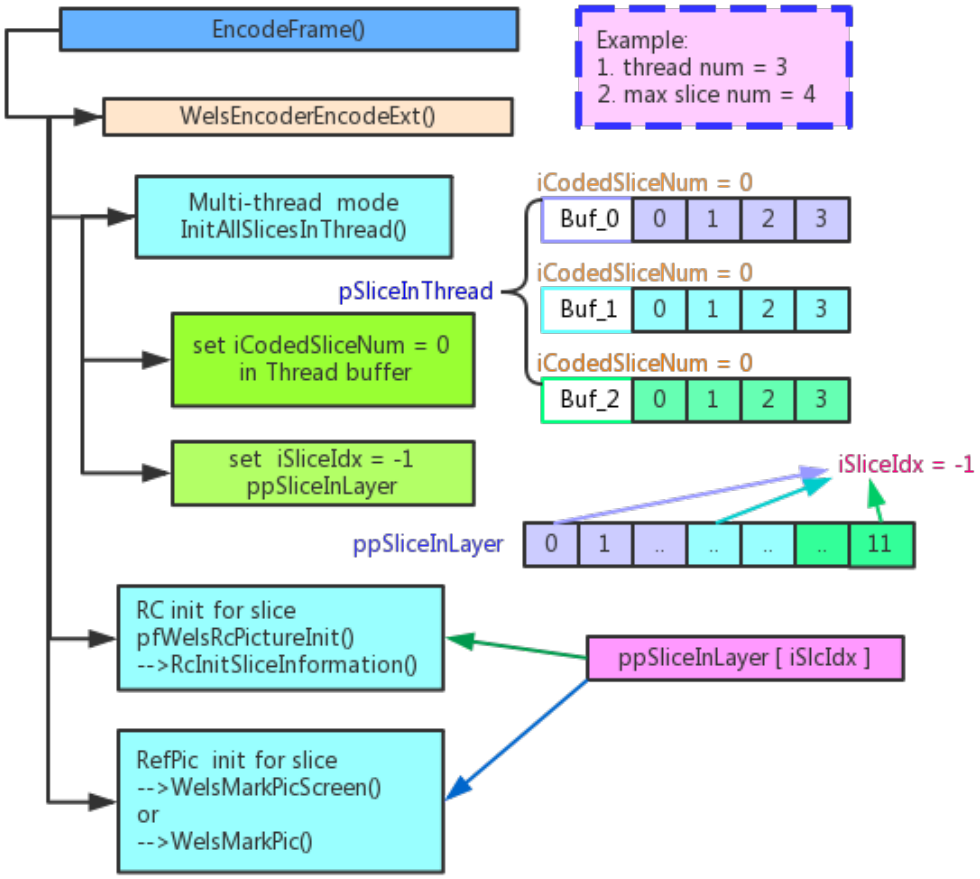
3.1. Allocate and initial before encode first IDR

- Allocate pSliceInThread buffer
- Allocate ppSliceInLayer buffer
- Allocate pFirstMbIdxOfSlice and pCountMbNumInSlice buffer
  Init slice boundary info



3.2 Slice buffer info before encoder one layer

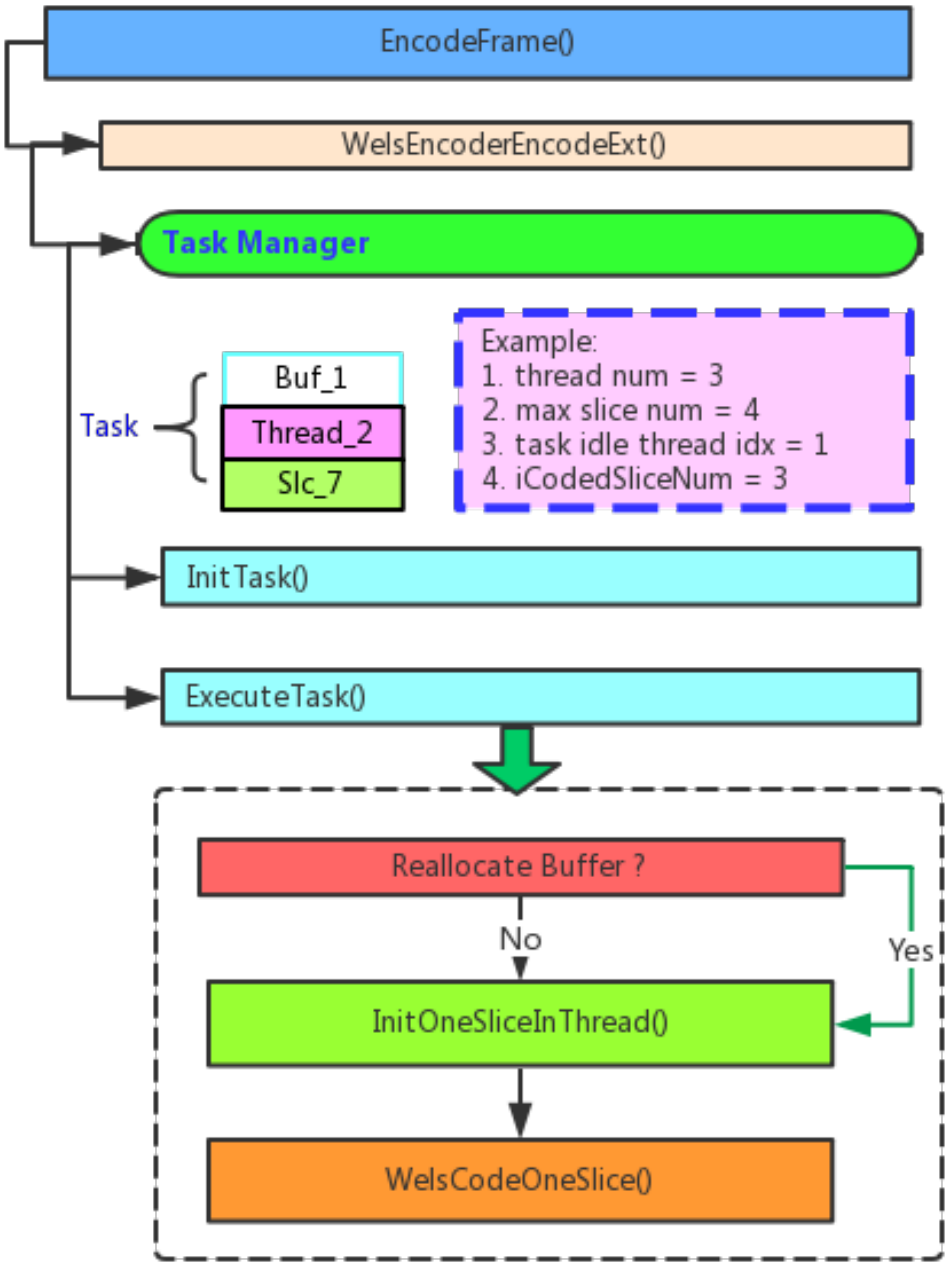
- Set iCodedSliceNum = 0 for all thread buffer
- Set iSliceIdx = -1 for all slice buffer (will set to actual slice idx during encoding one slice)
- Init RC info for all slice level parameters
- Init reference pic info for all slices using ppSliceInLayer[iSlcIdx]



3.3 Slice buffer init/update/reallocate during encoding one slice

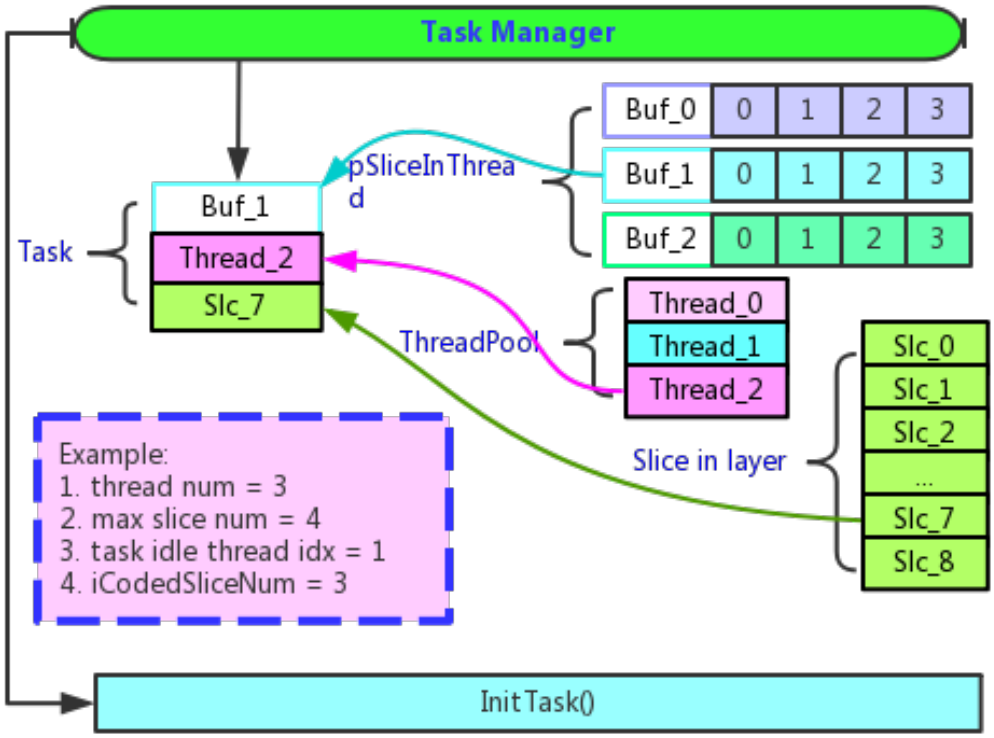
3.3.1 basic flow chart for encoding one slice task

- Task manager: Get idle thread, unused thread buffer, and encoding slice
- Init task: Init slice boundary info etc.
- Excute task: Encode one slice etc



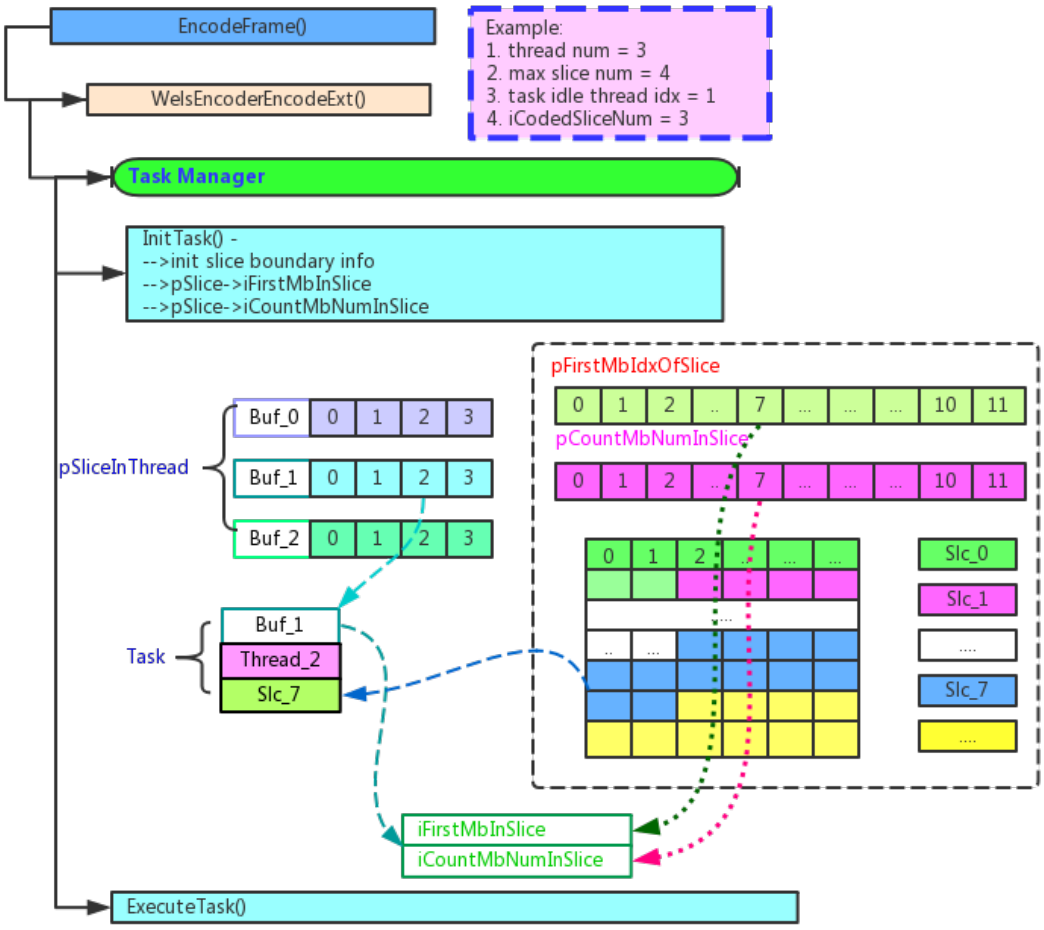
3.3.2 Task manager

- Get unused thread buffer, in this example, choose Buf_1 as slice buffer
- Get Idle thread, thread_2
- Get encode slice index in layer which will be encoded in this task, Slc_7



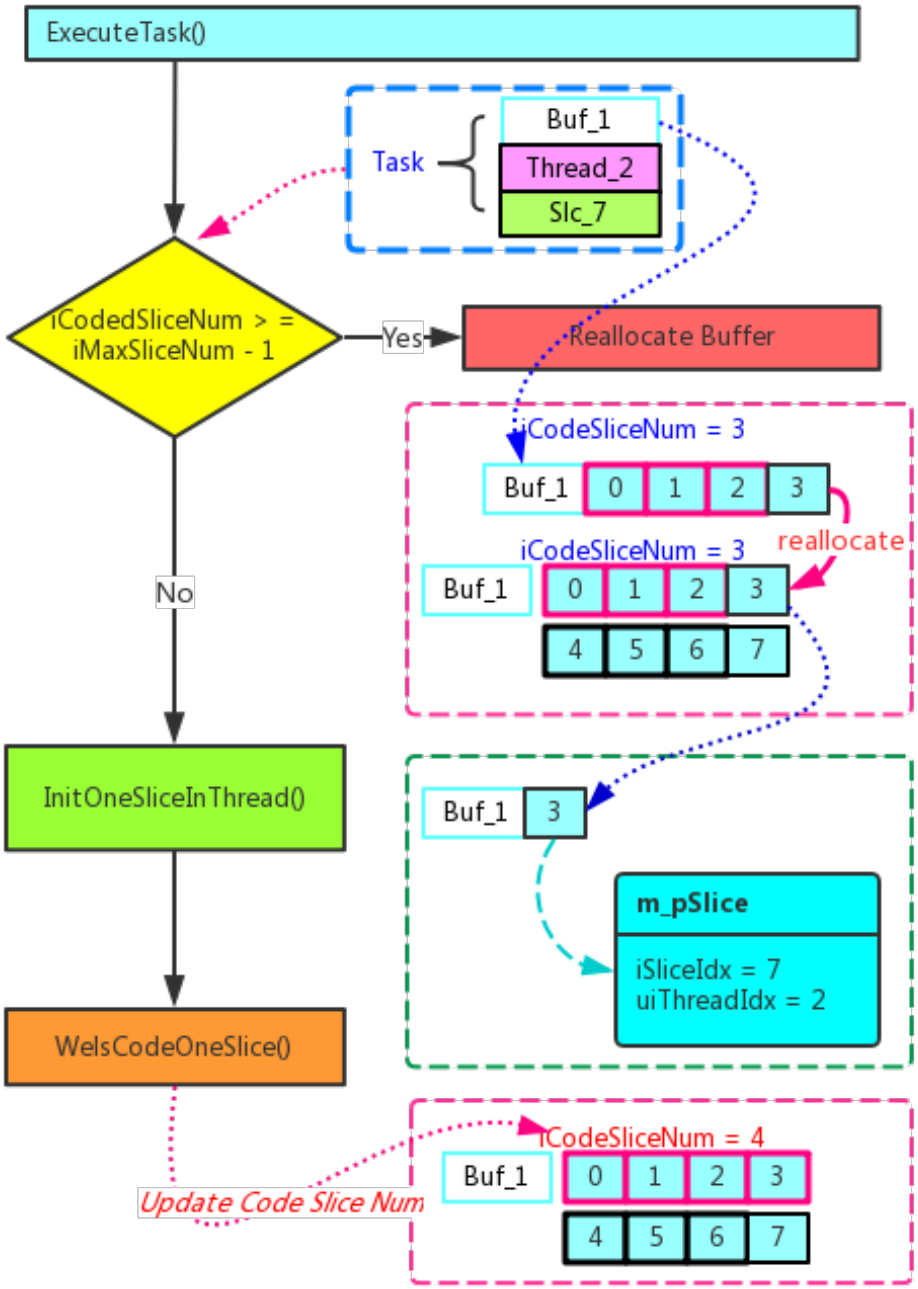
3.3.3. InitTask before encode one slice

- init rc with boundary info
- update next slice boundary(start MB index etc.)



3.3.4 Slice buffer update/reallocate during execute task

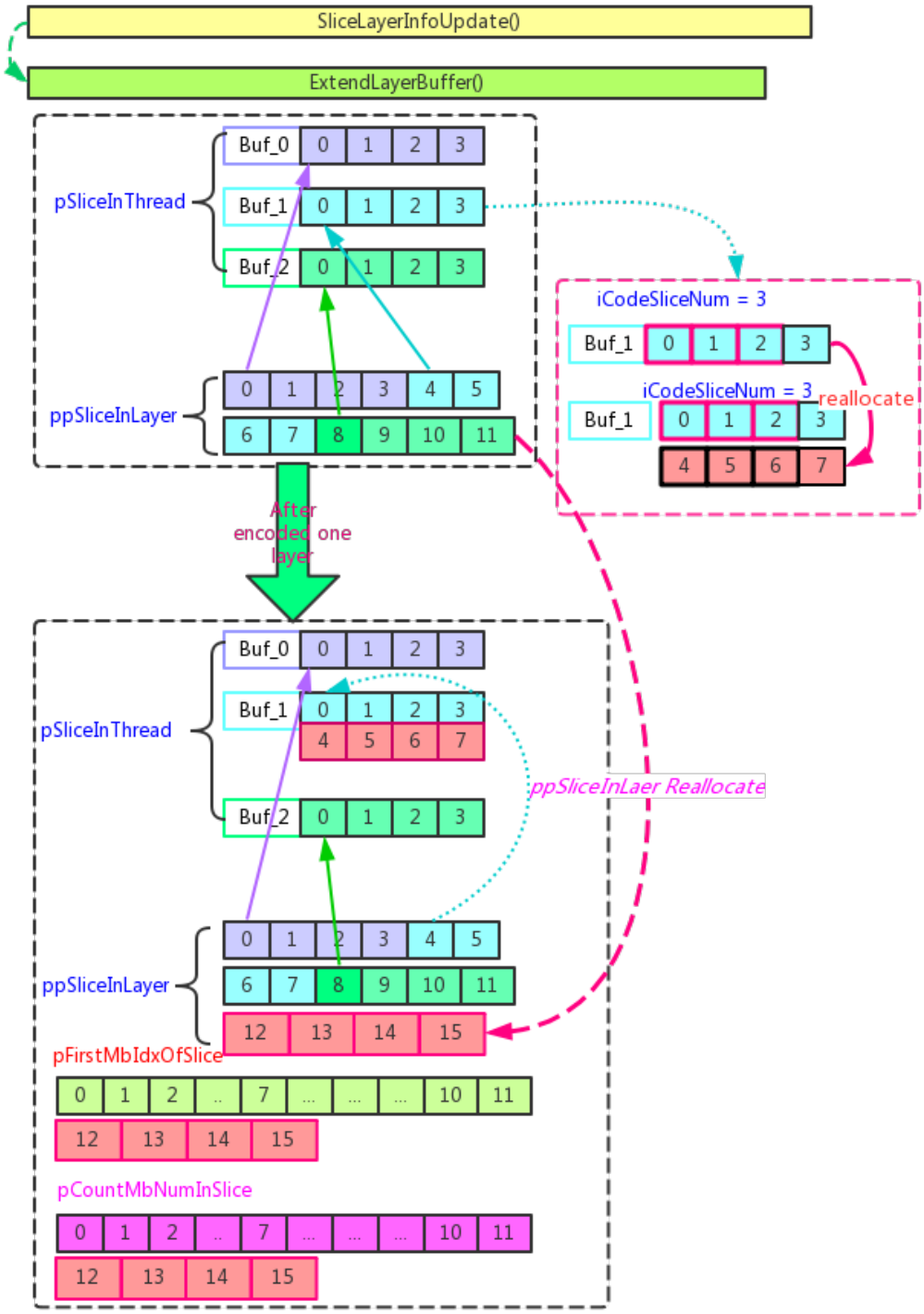
- check whether need to reallocate
- init m_pSlice buffer
- update thread buffer info, iCodeSliceNum ++



3.4. Slice buffer update after encoder one layer

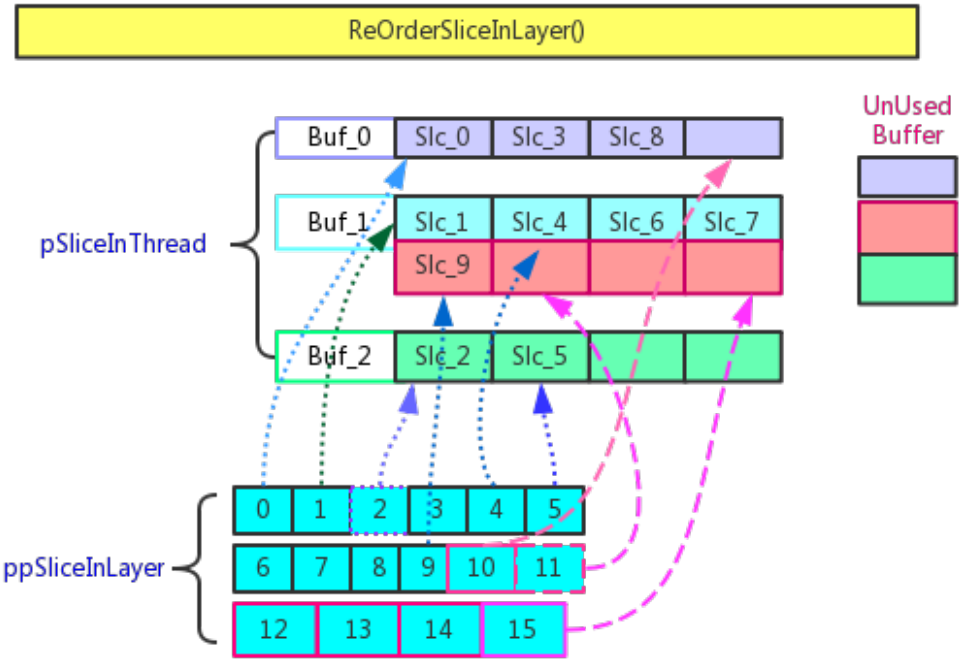
3.4.1. ppSliceInLayer update: extend and reallocate ppSliceInLayer buffer,

ppSliceInLayer buffer num == All thread slice buffer num



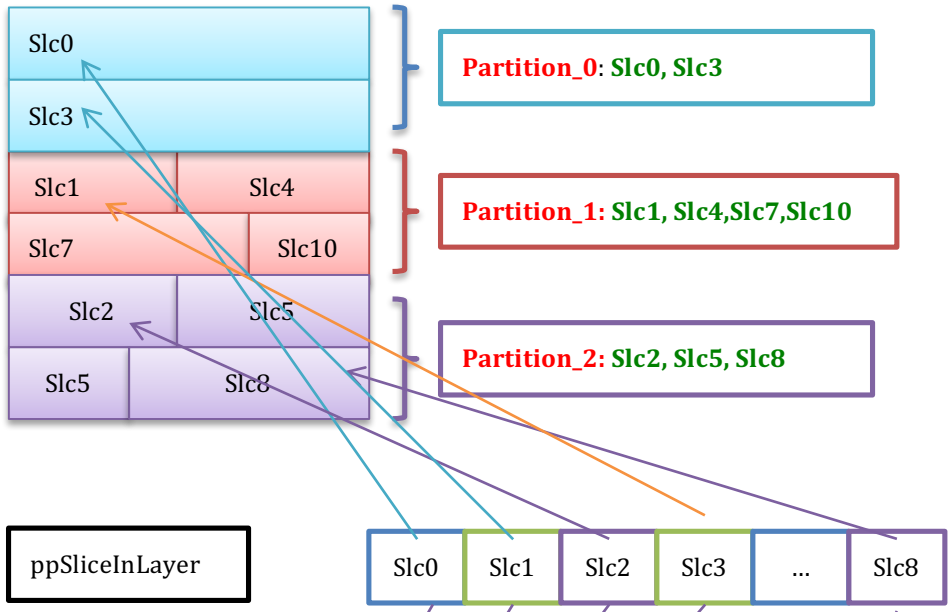
3.4.2. Slice buffer reorder

example 1

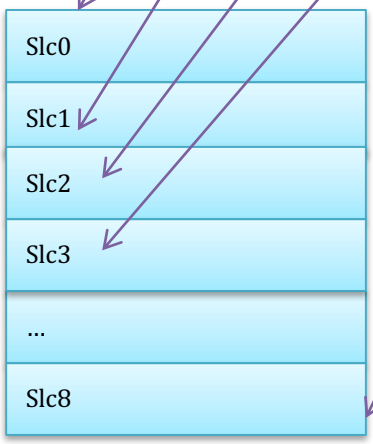


Example 2 Slice index in different slice mode.

Dynamic slice mode

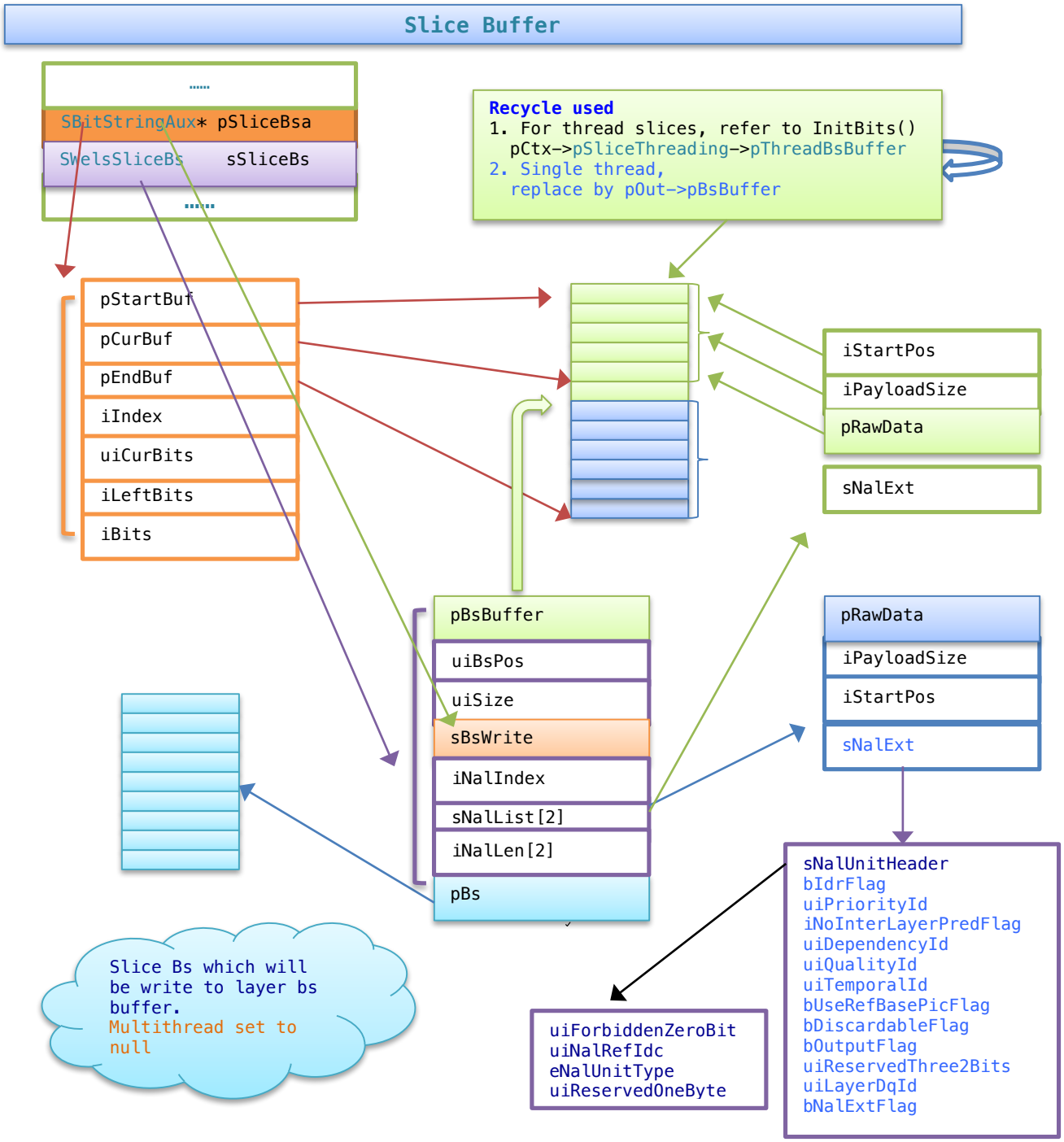


Non-dynamic slice mode

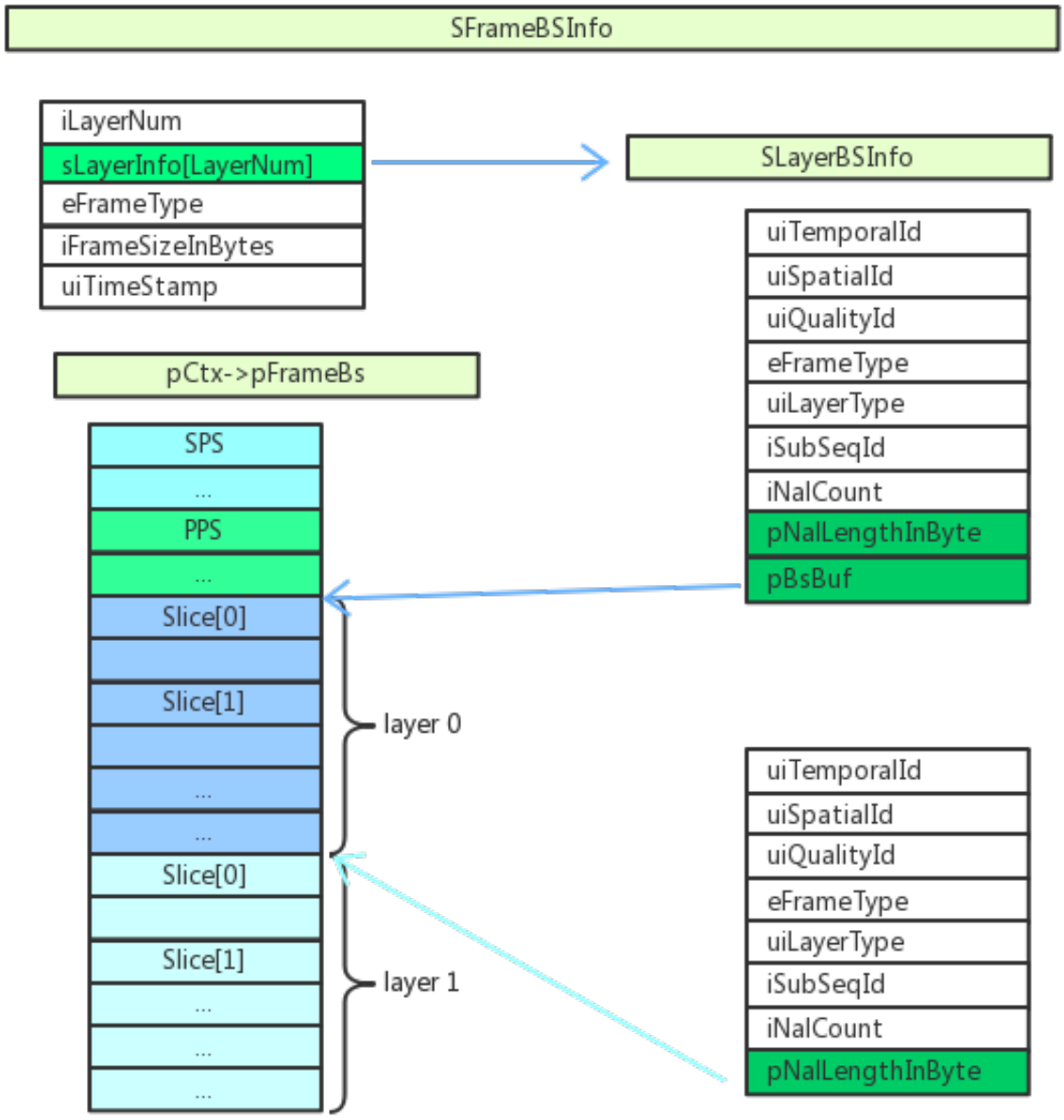


4. Bs buffer design

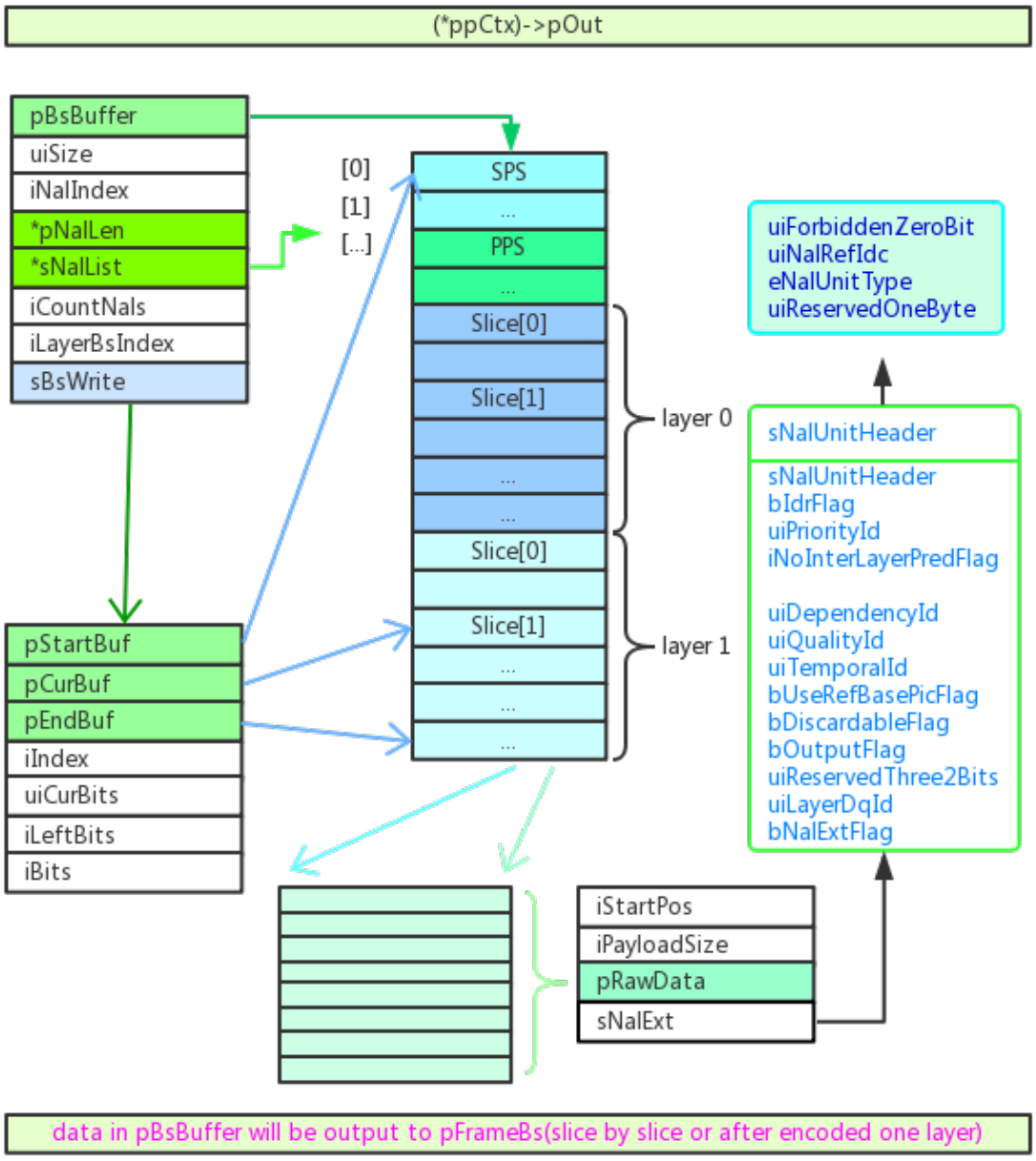
4.1. Slice Bs buffer



4.2. Frame bs buffer



4.3 pCtx-pOut


(*ppCtx)->pOut
pBsBuffer
uiSize
iNalIndex
*pNalLen
*sNalList
iCountNals
iLayerBsIndex
sBsWrite
[0]
[1]
[...]
SPS
...
PPS
...
Slice[0]
Slice[1]
...
...
Slice[0]
Slice[1]
...
...
...
layer 0
layer 1
uiForbiddenZeroBit
uiNalRefIdc
eNalUnitType
uiReservedOneByte
sNalUnitHeader
sNalUnitHeader
bIdrFlag
uiPriorityId
iNoInterLayerPredFlag
uiDependencyId
uiQualityId
uiTemporalId
bUseRefBasePicFlag
bDiscardableFlag
bOutputFlag
uiReservedThree2Bits
uiLayerDqId
bNalExtFlag
pStartBuf
pCurBuf
pEndBuf
iIndex
uiCurBits
iLeftBits
iBits
iStartPos
iPayloadSize
pRawData
sNalExt
data in pBsBuffer will be output to pFrameBs(slice by slice or after encoded one layer)

4.4. dynamic slice, slice boundary strategy