How to use

1. Test script brief introduction

Git Repos:

https://sqbu-github.cisco.com/huashi/OpenH264MemCheckTool

Test Script:

For more detail about scripts, please refer to .2~4

- **run_ParseMemCheckLog.sh**
  parse memory allocate/free/leak size for one test case
- **run_MememAnalyseForOneYUV.sh**
  test all cases for one YUV and output memory/FPS statistic info
- **run_Main.sh**
  test all YUVs for all cases and output all memory/FPS statistic info
- Other scripts as test tool called by above script
  `run_GetYUVList.sh`
  `run_GetYUVPath.sh`
  `run_ParseYUVInfo.sh`

Test case in configuration file:

Using default setting in below cfg files which copied from openh264/testbin/

- Welsenc.cfg
- Layer2.cfg

And combine with below parameters in run_MememAnalyseForOneYUV.sh

```
SliceMode=(0  1  2 3)
SliceNum=(1 4  4 0 )
ThreadNum=(1 2 3 4)
SliceSize=(1500 600)
```

Test codec:

- Origin design
  `Codec version info, refer to Codec_OriginDesign/Codec_Info.txt`

- New design
  `Codec version info, refer to Codec_NewDesign/Codec_Info.txt`

- New design with reallocate step ¼ MaxSliceNumOld
  `Codec version info, refer to Codec_NewDesign_Step_1-4/Codec_Info.txt`

- New design with reallocate step 1/3 MaxSliceNumOld
  `Codec version info, refer to Codec_NewDesign_Step_1-3/Codec_Info.txt`

Test data:

- Overall data structure
  `TestSpace ="TestData/${Codec}/${Arch}"`
  Example:
  ```
  TestData/Codec_OriginDesign/x86
  TestData/Codec_OriginDesign/x64
  TestData/Codec_NewDesign/x86
  TestData/Codec_NewDesign/x64
  TestData/Codec_NewDesign_Step_1-3/x86
  TestData/Codec_NewDesign_Step_1-3/x64
  TestData/Codec_NewDesign_Step_1-4/x86
  TestData/Codec_NewDesign_Step_1-4/x64
  ```

- Test data file generate by **encoder** for one case
  `TestMemLogFile="${TestSpace}/enc_mem_check_point_${YUVName}_${SliceMode[$i]}_${SliceNum[$i]}_${iThrdNum}_${iSlcSize}.txt"`

- Test data file generate by **run_ParseMemCheckLog.sh** for one case
  `TestAnalyseResult="${TestSpace}/MemAnalyseResut_${YUVName}_${SliceMode[$i]}_${SliceNum[$i]}_${iThrdNum}_${iSlcSize}.txt"`

- Test data file for one YUV by **run_MememAnalyseForOneYUV.sh**
  `TestReport="${TestSpace}/MemReport_For_${YUVName}.csv"`

- Test data file for all YUVs by **run_Main.sh**
  `TestReportForAllYUV="${TestSpace}/MemReport_For_AllYUVs.csv"`

Test data example:

For **new design** with **x64** encoder,
`TestSpace= TestData/Codec_NewDesign/x64/`

case is: slice mode = 1, slice num = 4,
thread num = 3, target bit rate = 600
YUV is : Zhuling_1280x720.yuv

- memory file generated by encoder, rename and remove to:
  `TestData/Codec_NewDesign/x64/enc_mem_check_point_Zhuling_1280x720.yuv_1_4_3_600.txt`

- Memory analyse result by **run_ParseMemCheckLog.sh** is:
  `TestData/Codec_NewDesign/x64/MemAnalyseResut_Zhuling_1280x720.yuv_1_4_3_600.txt`

- Test report by **run_MememAnalyseForOneYUV.sh** for Zhuling_1280x720.yuv is:
  `TestData/Codec_NewDesign/x64/MemReport_For_Zhuling_1280x720.yuv.csv`

- Ater test all YUV by **run_Main.sh**; report for all YUVs is:
  `TestData/Codec_NewDesign/x64/MemReport_For_AllYUVs.csv`

2. Memory analyse for encode one case

run_ParseMemCheckLog.sh

```
#*************************************************************************
#  brief:  1. parse memory allocate/free info based on memory statistic log
#             ./enc_mem_check_point.txt
#
#          2. to generate above log file,
#              need to enable below macro in codec/common/memory_align.h
#              #define MEMORY_CHECK 1
#
#          3. the output may contain the summary of memory allocate/free
#             statistic info during encoding process, and detail info of all
#             buffers which have been allocated/freed during/after'
#             encoding process
#
#  usage:   ./run_ParseMemCheckLog.sh $LogFile $OutFile $Option
#            ----LogFile: should be enc_mem_check_point.txt
#                         or other rename log file
#            ----OutFile:  the final report file for memory statistic info
#            ----Option:
#                       1)MemCheck     :   all detail info
#                       2)OverallCheck :   summary info only
#
#
#*************************************************************************
```

Example:

Analyse new design x64 encoder's memory info, with

Case: slice mode = 1, slice num = 4,
thread num = 3, target bit rate = 600

```
LogFile="TestData/Codec_NewDesign/x64/enc_mem_check_point_Zhuling_1280x720.yuv_1_4_3_600.txt"

OutFile="TestData/Codec_NewDesign/x64/MemAnalyseResut_Zhuling_1280x720.yuv_1_4_3_600.txt"

./run_ParseMemCheckLog.sh ${LogFile} ${OutFile} OverallCheck
./run_ParseMemCheckLog.sh ${LogFile} ${OutFile} MemCheck
```

if using OverallCheck, will only output below summary info

```
TestData/Codec_NewDesign/x64/MemAnalyseResut_Zhuling_1280x720.yuv_1_4_3_600.txt
MemLeakStatus False: AllocatedNum--FreeNum (116--116) :
AllocateSize==FreeSize==LeakSize (25615532==25615532==0)
```

If using OverallCheck, will output detail info about all buffer allocate size, free size, allocate num, free num etc

3. memory analyse for one YUV

run_MememAnalyseForOneYUV.sh

```
#*************************************************************************
#  brief:  Analyse memroy allocate statistic info for one YUV with
#              all test cases
#
#  usage:   ./run_MememAnalyseForOneYUV.sh \$YUVName \$YUVDir \$TestSpace
#            ----YUVName:   Test YUV's name
#            ----YUVDir:    Test YUV's folder, script will search YUV
#                                within this folder
#            ----TestSpace: Test data output folder
#
#*************************************************************************
```

Test case for one YUV in script(you can modify it if you want):

```
SliceMode=(0  1  2 3)
SliceNum=(1 4  4 0 )
ThreadNum=(1 2 3 4)
SliceSize=(1500 600)
```

Example:

Test yuv with new design with arch=x64 encoder,

- **Step 1:**
  copy encoder h264enc from Codec_NewDesign/x64/
- **Step 2:**
  YUVName=" Zhuling_1280x720.yuv"
  YUVDir="../../YUV"
  TestSpace=" TestData/Codec_NewDesign/x64"
- **Step 3:**
  ./run_MememAnalyseForOneYUV.sh ${YUVName} ${YUVDir} ${TestSpace}

Then all test data for Zhuling_1280x720.yuv is under folder TestData/Codec_NewDesign/x64/

Test report file for zhuling is:
TestData/Codec_NewDesign/x64/MemReport_For_Zhuling_1280x720.yuv.csv

4. Memory analyse for all YUVs

run_Main.sh

```
#*************************************************************************
#  brief:  Analyse memroy allocate statistic info for all YUVs with
#              all test cases
#
#  usage:   ./run_Main.sh \$TestYUVListDir
#            ----TestYUVListDir: Test YUV's folder, script will search YUV
#                                in this folder
#
#*************************************************************************
```

Test all YUVs with all test cases using all encoder in encoder set

You can modify below test sets if you want

```
TestYUVList=("CiscoVT2people_320x192_12fps.yuv"  \
             "xuemei_640x360.yuv" \
             "Zhuling_1280x720.yuv" \
             "desktop_dialog_1920x1080_i420.yuv" \
             "SlideShowFast_SCC_2880x1800.yuv")

TestCodecList=("Codec_NewDesign" \
               "Codec_NewDesign_Step_1-3" \
               "Codec_NewDesign_Step_1-4" \
               "Codec_OriginDesign")

TestCodecArchList=("x64" "x86")
```

And combine with cases in run_MememAnalyseForOneYUV.sh

```
SliceMode=(0  1  2 3)
SliceNum=(1 4  4 0 )
ThreadNum=(1 2 3 4)
SliceSize=(1500 600)
```

Example:

```
./run_Main.sh ../../YUV
```

Then all test data will be generated under all below folders:

```
TestData/Codec_OriginDesign/x86
TestData/Codec_OriginDesign/x64
TestData/Codec_NewDesign/x86
TestData/Codec_NewDesign/x64
TestData/Codec_NewDesign_Step_1-3/x86
TestData/Codec_NewDesign_Step_1-3/x64
TestData/Codec_NewDesign_Step_1-4/x86
TestData/Codec_NewDesign_Step_1-4/x64
```

5. Example for final Test Data

All test data for all test encoder all arch wit all test YUVs for all cases:

https://sqbu-github.cisco.com/huashi/OpenH264MemCheckTool/tree/master/TestData

Final Test report for all YUVs:

https://sqbu-github.cisco.com/huashi/OpenH264MemCheckTool/blob/master/NewDesignTestData.xlsx

example for SlideShowFast_SCC_2880x1800.yuv

						Origin design		New design		Delta(%)	
SlCM	SlcN	PicW	PicH	ThrdN	SlcSz	AllocSize	FPS	AllocSize	FPS	deltaSiz	deltaFPS(%)
3	0	2880	1800	1	1500	99424308	85.17	99171165	88.36	-0.25	3.75
3	0	2880	1800	1	600	99738905	80.77	100121689	86.73	0.38	7.38
3	0	2880	1800	2	1500	162263844	123.39	137297154	122.54	-15.39	-0.69
3	0	2880	1800	2	600	162263844	124.79	199850984	127.57	23.16	2.23
3	0	2880	1800	3	1500	170984608	151.67	149679510	160.58	-12.46	5.87
3	0	2880	1800	3	600	170984608	157.59	244683892	156.55	43.10	-0.66
3	0	2880	1800	4	1500	178796354	165.30	168923895	175.24	-5.52	6.01
3	0	2880	1800	4	600	178796354	166.49	221484277	180.50	23.88	8.41

-15.39% means that new design decrease overall allocate size;
43.10% means that new design increase overall allocate size due to reallocate module which is not allowed in origin design

Final test summary:

New design most of time save the overall memory size,
FPS is the same with origin disign