Some Processes and Questions

• Compatibility between SHM and Visual Studio 2019

I spent quite some time figuring out how to build SHM on my Visual Studio setup. Initially, I was using an older SHM version that could only be built with Visual Studio 2010, which led to multiple compatibility errors when I tried building it on Visual Studio 2019.
Eventually, I found that **SHM-12.4** supports **Visual Studio 2017**, and I realized that I could install the **VS2017 build tools** within Visual Studio 2019. After doing so, I successfully built SHM using the 2017 toolset inside my VS2019 environment.

• Building the customized `Two_layer_Corgi.cfg` file

In the original configuration parameters you shared, I noticed that:

```
InputBitDepth0 = 8
InputBitDepth1 = 10
```

I assume that for the high-resolution image/video, you used **10 bits**, and for the low-resolution one, **8 bits**.
Since I am working with a very simple image, I used **8 bits for both layers**. I know that FFmpeg can output YUV420 images with 10 bit, but I did not enable that option this time. If you think it is important for consistency or testing accuracy, I can easily change it to 10 bit output.

• IntraPeriod0 / IntraPeriod1 settings

In your original configuration, `IntraPeriod0` was set to **48**, which makes sense for video sequences.
However, since I only have a single frame (a static image), every frame should technically be an intra frame. So, I initially set `IntraPeriod0` and `IntraPeriod1` to **1**, but that caused an error during encoding.
Interestingly, when I changed them to **-1**, the process worked without errors. I'm not entirely sure **why -1 is accepted here while 1 is not**.

• Decoding question

When I first tried to decode both layers simultaneously:

```
TAppDecoder.exe -b "E:\Dropbox\Test\app\SHM\Corgi_bitstream.bin" -ls 2 -o0
"E:\Dropbox\Test\app\SHM\Corgi_decoded_0.yuv" -o1
"E:\Dropbox\Test\app\SHM\Corgi_decoded_1.yuv"
```

the decoded **layer 0** `Corgi_decoded_0.yuv` was **empty** (size = 0).

Later, I decoded the two layers separately:

```
TAppDecoder.exe -b "E:\Dropbox\Test\app\SHM\Corgi_bitstream.bin" -ls 2 -lid
0 -o0 "E:\Dropbox\Test\app\SHM\Corgi_decoded_0.yuv"
```

```
TAppDecoder.exe -b "E:\Dropbox\Test\app\SHM\Corgi_bitstream.bin" -ls 2 -lid
1 -o1 "E:\Dropbox\Test\app\SHM\Corgi_decoded_1.yuv"
```

and this time both outputs were correct.
I’m not sure why decoding both layers together results in an empty base-layer output, while decoding them separately works fine.