

# Concept and Code Description of X264 on Cell

## 1 Concept

My project is to utilize the power of CELL BE, a multi-core architecture, as much as possible to handle the difficulties of real-time H.264 encoding. Figure 1 is the basic parallel structure of our encoder. Every SPU is responsible for encoding a strip of macroblocks in a frame, which is column-divided. This structure is very attractive due to balanced overload of every SPU and the number of all working SPU can be easily and adaptively adjusted according to the complexity of encoding.

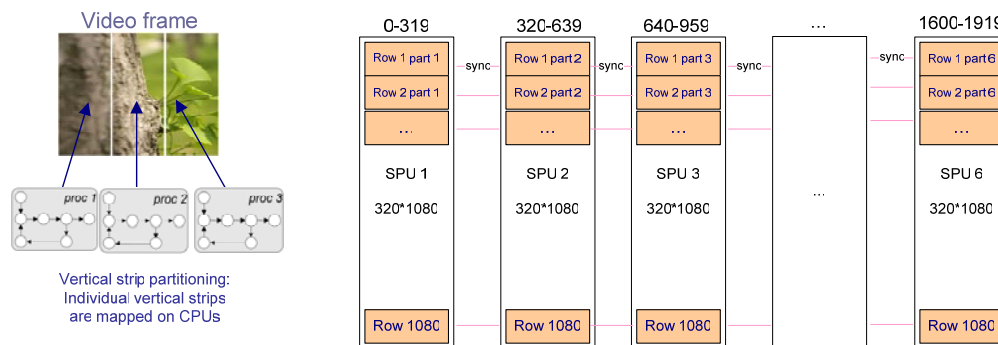


Figure 1 Parallel structure of our encoder

## 2 Code Structure

Currently, I've created a basic parallel code structure of X264, and implemented and optimized Loop Filter module on Cell platform. Since extremely slow speeds to run X264 executable on simulator, I can only test and ensure the validity of input bitstream of QCIF format. Considering the symmetrical nature of every SPU in this parallel structure, I strived to optimize the performance of loop filter function on single SPU, which occupies about 10% encoding time during X264 encoding.

Following table shows the code structure of my work:

File/Directory	Description
x264_cell/*.ch]	
x264_cell/common	PPU program. source of this code is mostly from X264. rev652, only
x264_cell/encoder	<i>ppu_parm.h</i> , <i>spu_thread.c</i> and <i>common/encoder.c</i> are created or modified.
x264_cell/extra	

x264_cell/spu/*	SPU program. The ultimate goal is to make this spu program to encode a strip of frame, but now I only implement and optimized Loop filter related functions on SPU.
x264_cell/spu/dma_lst.h x264_cell/spu/spu_simd.h x264_cell/spu/func_param.h	Header files for SPU program, including my own macros of SIMD, DMA and program dependent structure definition.
x264_cell/spu/encoder.c	Main routine of SPU program, which only issues loop filter request currently.
x264_cell/spu/common/mc.c x264_cell/spu/common/frame.c	These files contain important functions of loop filter, such as SIMD optimized horizontal filter and vertical filter ( <i>mc_hh_vec</i> , <i>mc_hv_vec</i> , <i>mv_hc_vec</i> , <i>plan_expande_border</i> , etc), they are also optimized based on CBE with double buffer DMA, SIMD and huge register techniques.

X264\_cell source code can be compiled as following:

```
cd x264_cell
make
```

It will generate executable named “x264\_cell\_simd”, which is optimized using SIMD during loop filtering. X264\_cell without SIMD optimization can be made by:

```
make clean
make NOSIMD=1
```

This will generate executable named “x264\_cell”.

### 3 Basic Result

Directly compiling C code of X264 on SPU is not feasible. The first porting version with double buffer DMA support but without SIMD can encoded QCIF at 54.8 fps, using default parameters of X264 (Baseline), while after utilizing SIMD on only loop filter (~10% encoding time), the performance is increased to 62.9 fps with the same parameters.

### 4 Developing Environment

**Operating System: Ubuntu 7.04 i686 2.6.20-16-386**

**Hardware Platform: Sempron 2800+, 1G DDR2**

**Software: CBE SDK2.1**