

# 介绍如何移植 bento4 到 Andes 平台上

随着当今市场对于音频电子产品功能的需求越来越高，8 位和 16 位的 MCU 逐渐向 32 位的 MCU 转型已经成为市场趋势。晶心科技(Andes)作为亚洲首家原创性 32 位微处理器 IP 与系统芯片开发平台的设计公司，面向 32 位 MCU 市场推出了 Andes Core N9 系列低功耗高性价比的 32 位处理器软硬核 IP。基于该系列处理器，晶心科技针对不同音频应用提供了多种解决方案。包括各种音频格式的编码和解码，如 AAC，MP3，MP4，G729 等移植到 Andes 平台上。再加上 nds32 架构的优势和音频的扩展指令集，以及算法上的改进，所以编码解码器有了进一步的优化，使其代码空间变小，运行性能变高。本文以 MP4 的音频解码器为例，介绍如何 bento4 移植到 Andes 平台上。

## 1.MP4 简介:

MP4 为 MPEG-4 Part14，按照正规的 ISO/IEC 14496-14:2003 标准，其作为 MPEG-4 的一部分是一个多媒体存储格式专用的标准。一般常常用来存放数字音频和数字视频流文件(音频可以是：AAC，MP3，MP2 等,视频可以是 H.264，MPEG-4 AVC 等)，也可以存放其他特殊的数据像字幕和静止图像。更很多现在的存储格式一样 MPEG-4 Part14 允许在因特网上传输.这些文件的后缀名是.mp4。

MPEG-4 Part14 是基于苹果公司的 Quicktime 存储格式来实现。MPEG-4 Part14 是基本上和 QuickTime MOV 格式相同，但是它有专门支持 Initial Object Descriptors(IOD)格式和其他的 MPEG 特征。

下面我就详细的介绍下 MP4 怎样在我们 Andes 提供的 AndeSight 集成开发工具上解码。

## 2.环境与软件:

### 2.1 系统环境:

Linux Fedora8

### 2.2 开发环境:

AndeSight1.4

AndeSight 是晶心科技提供的一种基于 nds32 架构开发嵌入式工程的图形化的集成开发环境。主要由 AndeSight IDE, AndESLive 和 nds32 工具集 3 个部分组成。

AndeSight IDE 为工程师提供了各种友好的界面，包括编辑，编译，运行，调试或者评测等操作。

AndESLive 提供了基于 nds32 架构的仿真器和一种图形化的虚拟 SoC 构建模型，它与 AndeSight IDE 相结合为用户提供了一个虚拟的硬件平台。这个虚拟评估平台提供 Andes 自行定义 ISA 的多组系列 32 位 CPU IP 以及各种外围设备 IP，并且支持用户自定义 IP 模型。

AndESLive 配合 AndeSight IDE 不仅使得 SoC 设计者能在计划初期就开始软件设计、纠错、最优化等工作，并对系统架构及功能进行检验，而且使硬件工程师和软件工程师具有一样的能力去制作和修改他们各自的系统模型，可以有效的控制 NRE(NonRecurring Engineering)成本，让软件工程师在拿到硬件原型之前，即可以进行软件的开发和优化。

### 2.3 交叉编译器:

nds32le-linux-gcc

nds32 工具集中对应不同的 Andes Core 型号，不同的系统函数库以及大小端形式等条件，提供了各种对应的交叉编译器。这里我们选用 nds32le-linux-gcc。

### 2.4 软件包:

首先要从 <http://sourceforge.net/projects/bento4>

和 <http://sourceforge.net/projects/melo> 下载 Bento4-0.9.4.002 和 Melo-1.0.0.tar.gz 这两个包，把他们存放到指定目录下。

### 3 Bento4 的移植

#### 3.1 编译工程

操作的步骤：

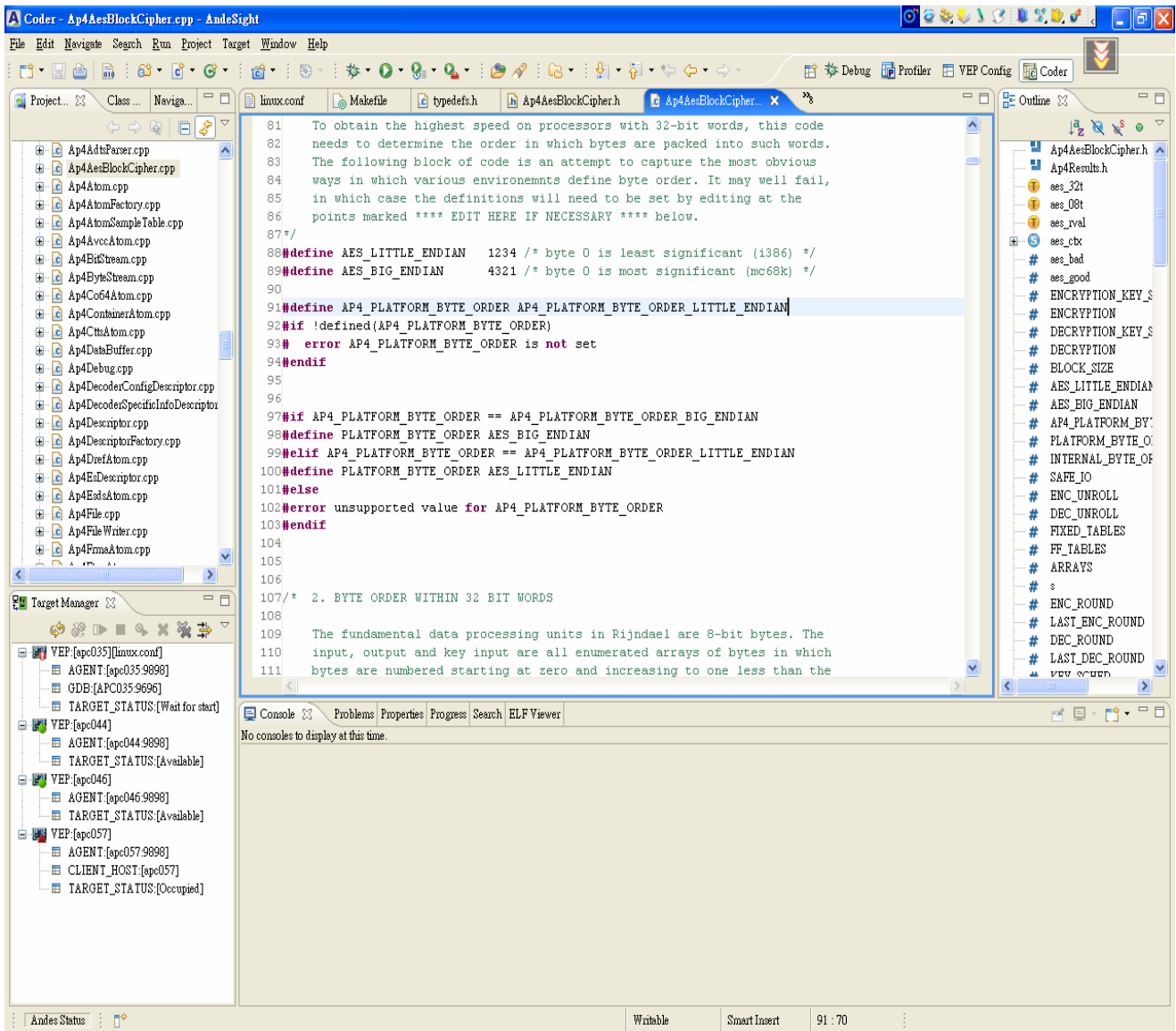
第一步：先将下载下来的两个包解压到一个指定的目录。

第二步：运行 AndeSight1.4，创建一个 Andes C++工程，命名为 Bento4\_EL。

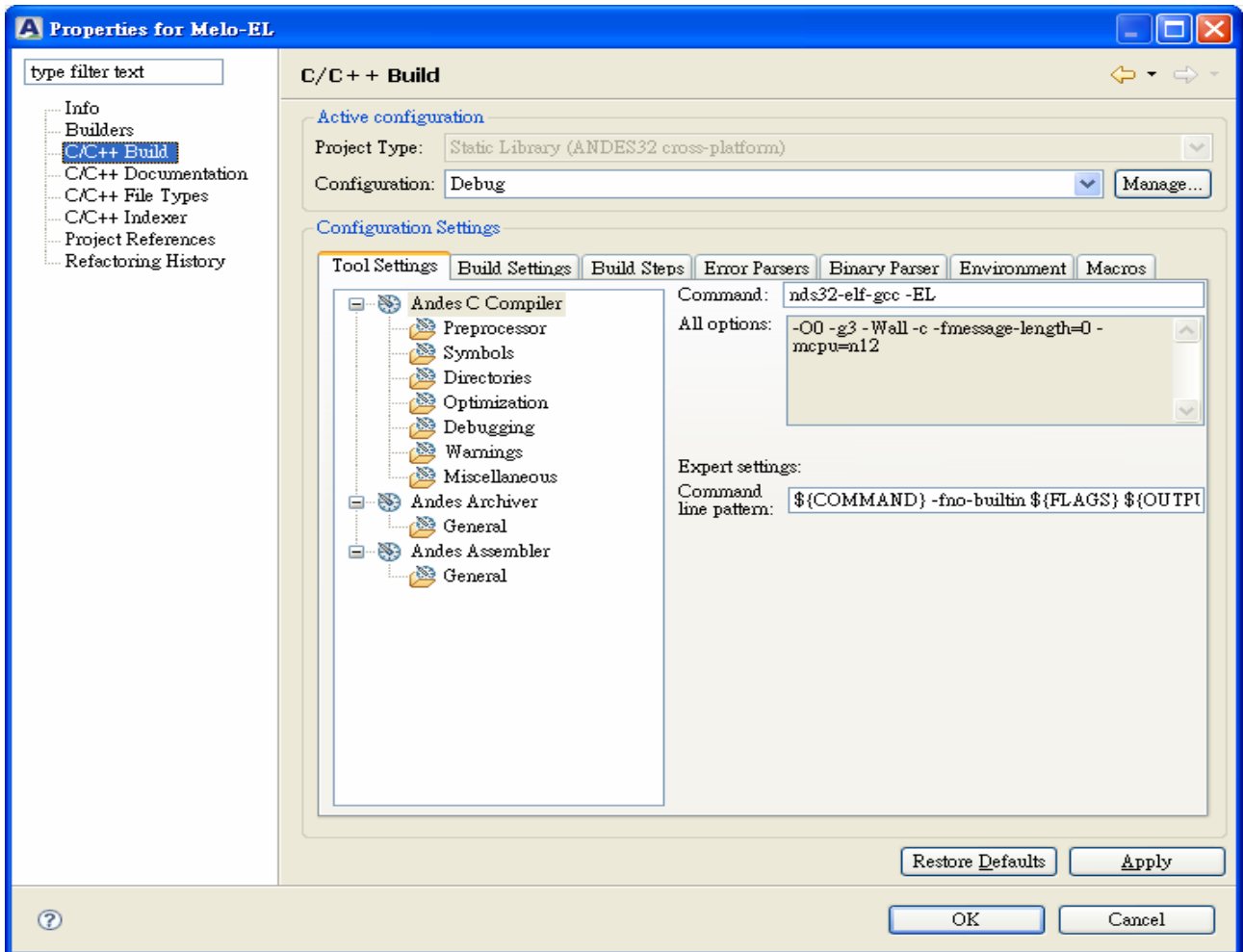
此外在新建的时候可以选择一款 Toolchain，不过在新建之后在工程的 Properties 中也可以设置 Toolchain。

第三步：在 Bento4\_EL 工程中导入所有源代码，找到刚才解压出来的文档目录 Bento4-0.9.4.002,到该目录下的 Source/C++/, 将其中的 Core, Codecs, Metadata, Crypto 和 System/ 下的 StdC 导入 Bento4\_EL 工程。

第四步：找到 Ap4AesBlockCipher.cpp 文件，在该文件的第 90 行加入一句：`#define AP4_PLATFORM_BYTE_ORDER AP4_PLATFORM_BYTE_ORDER_LITTLE_ENDIAN`。



第五步：右击 Bento4\_EL 工程选择 Properties 选项，在弹出来的界面中点击 C/C++ Build 选项，选择其中的 Toolchain 菜单，选择一款然后编译 Bento4-EL 工程，并且指定编译选项-static 和-EL, 如果默认就是-static 和-EL 编译,所以不需要改动。



第六步：新建一个名为 Melo\_EL 的 C 工程的静态链接库，把库的名字命名为 melo.a，如下面两个图所示，然后找到 Melo-1.0.0 目录，导入 Melo-1.0.0/Source 下所有的文件，然后用 little endian 编译。

### New Project

Select a type of project

Select the platform and configurations you wish to deploy on

Project Type:

Configurations:

- Debug
- Release

Show All Project Types

Show All Configurations

### Properties for Melo\_2

C/C++ Build

Active configuration

Project Type:

Configuration:

Configuration Settings

Tool Settings | Toolchain | **Build Settings** | Build Steps | Error Parsers | Binary Parser | Environment | Macros

Build output

Artifact name:  Artifact extension:

Build command

Use default command

Build Macros usage

Expand Build Environment Macros

Internal Builder

NOTE: This is experimental functionality

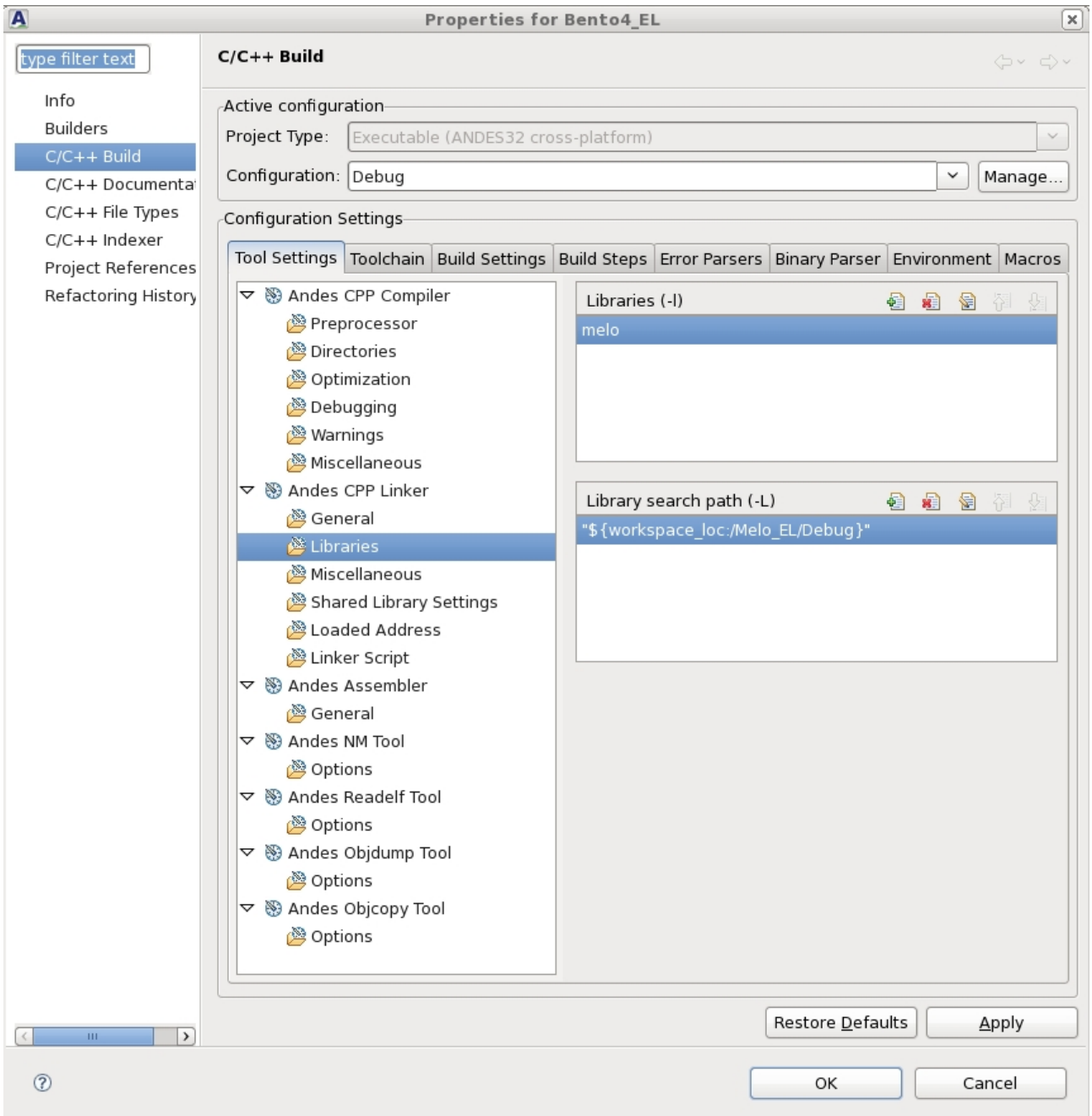
Enable Internal Builder

Ignore build errors

第七步：在 Bento4\_EL 工程中导入 Melo-1.0.0/Apps/MeloDecode 中的 MeloDecoder.cpp 文件，然后把第 128 行的 "MLO\_SampleBuffer\* pcm\_buffer = NULL;" 搬到第 121 行，如下图所示，然后保存。

```
112 MLO_DecoderConfig decoder_config;
113 MLO_Result result = MLO_DecoderConfig_Parse(encoded_config->GetData(),
114                                             encoded_config->GetDataSize(),
115                                             &decoder_config);
116 if (MLO_FAILED(result)) {
117     fprintf(stderr, "ERROR: decoder config is unsupported or unsupported (%d)\n", result);
118     return;
119 }
120
121 MLO_SampleBuffer* pcm_buffer = NULL;
122 // create the decoder
123 MLO_Decoder* decoder = NULL;
124 result = MLO_Decoder_Create(&decoder_config, &decoder);
125 if (MLO_FAILED(result)) {
126     fprintf(stderr, "ERROR: failed to created MLO_Decoder (%d)\n", result);
127     goto end;
128 }
129
130 result = MLO_SampleBuffer_Create(0, &pcm_buffer);
131 if (MLO_FAILED(result)) goto end;
132
133 while (AP4_SUCCEEDED(track->ReadSample(index, sample, data))) {
134     result = MLO_Decoder_DecodeFrame(decoder, data.GetData(), data.GetDataSize(), pcm_buffer);
135     printf("MLO_Decoder_DecodeFrame return %d\n", result);
136     output->Write(MLO_SampleBuffer_GetSamples(pcm_buffer), MLO_SampleBuffer_GetSize(pcm_buffer));
137     index++;
138 }
139
140 end:
141 if (pcm_buffer) MLO_SampleBuffer_Destroy(pcm_buffer);
142 if (decoder) MLO_Decoder_Destroy(decoder);
143 }
```

第八步：把所有 Melo/Source 下的头文件导入到工程 Bento4 中，并且在 Andes CPP Linker 的 Libraries 选项栏添加 -lmelo 和 -L "\${workspace\_loc:/Melo\_EL/Debug}" 这两个参数，然后选择一款 Toolchain，再编译(-EL)。

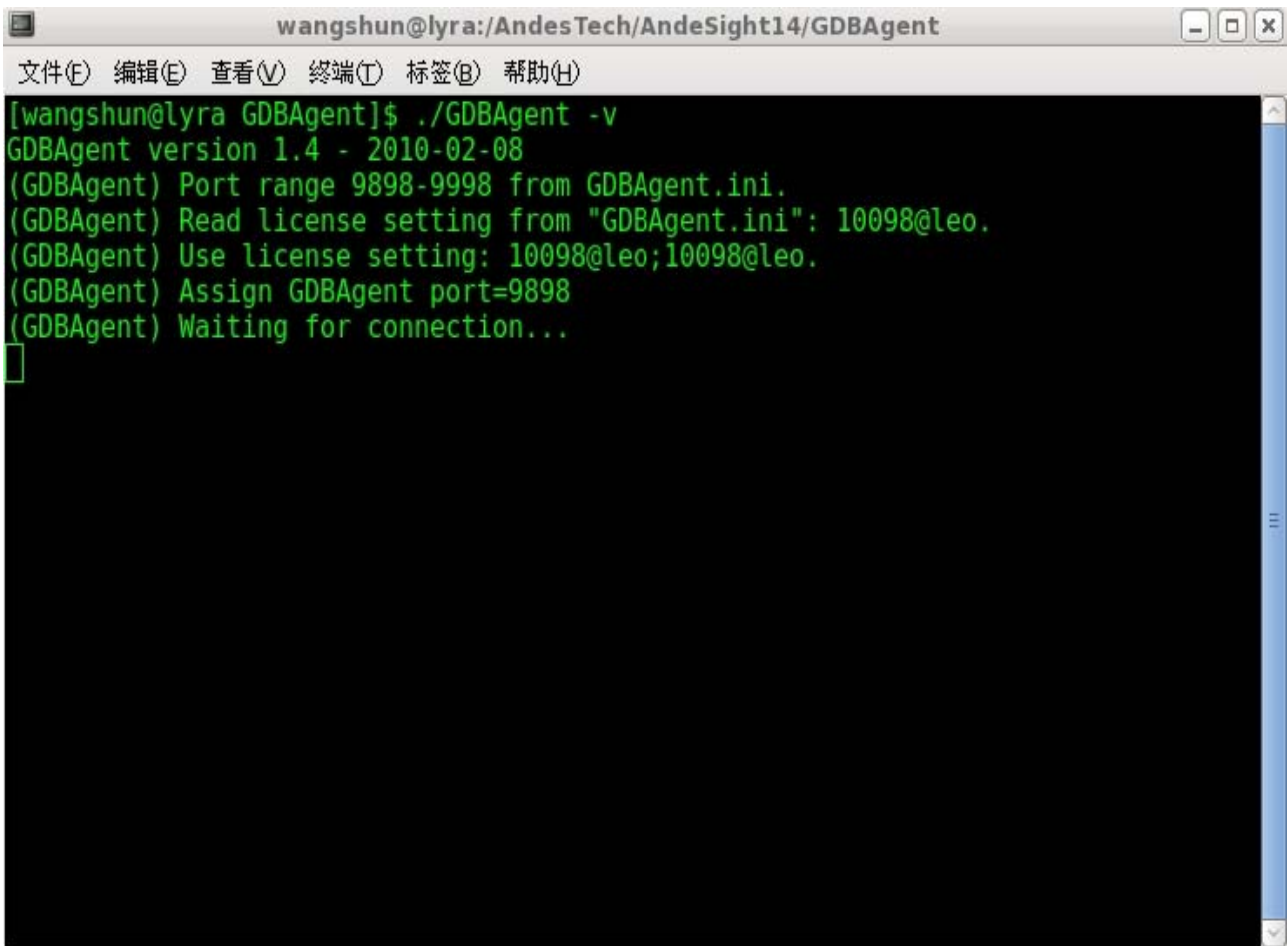


### 3.2 运行工程

以上这些步骤就是所有的编译过程，最终会生成一个 MP4 decoder 的可执行文档。

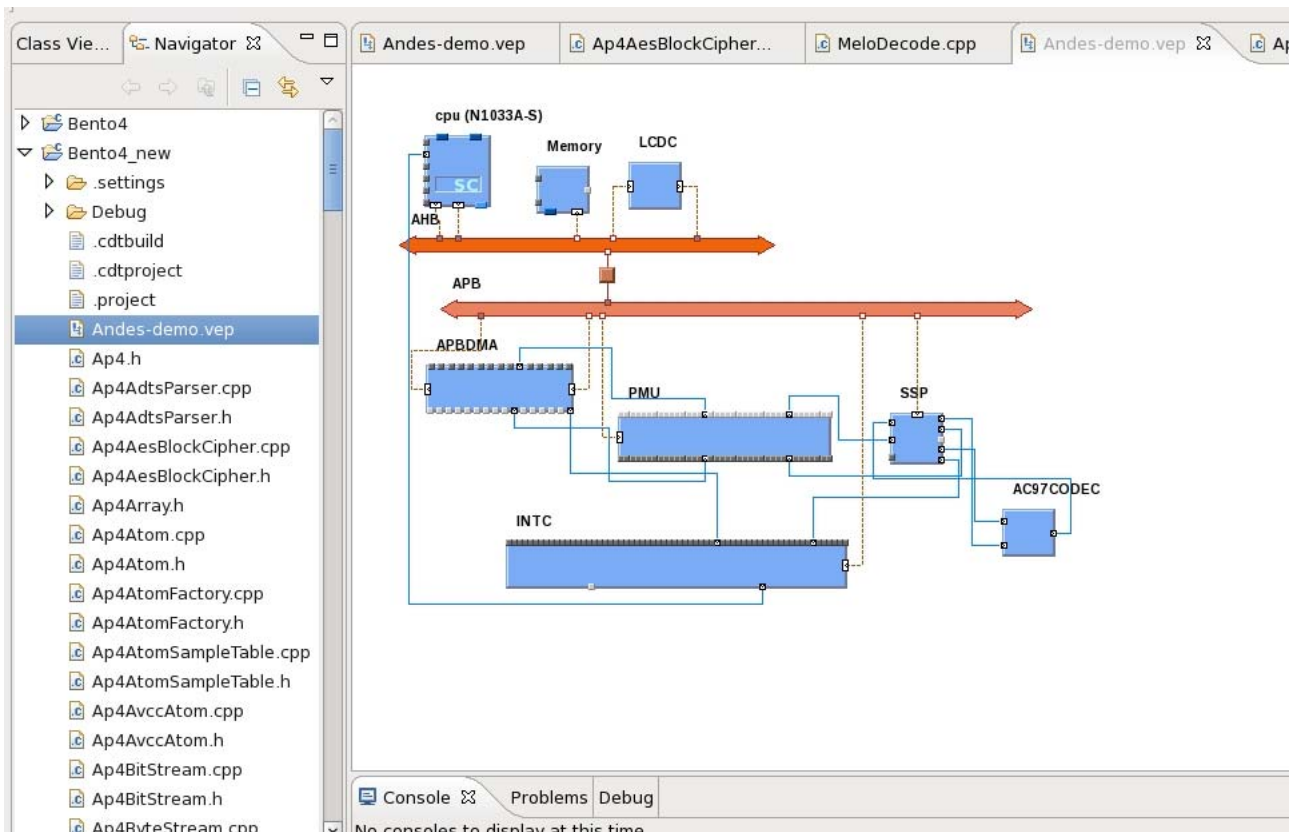
接下来是运行的部分：

第一步：说到运行首先要开启我们的 GDBAgent，在 command line 中进入 AndeSight 安装的目录进入 GDBAgent 这个目录，在 command line 上输入：“./GDBAgent -v”，开启 GDBAgent。

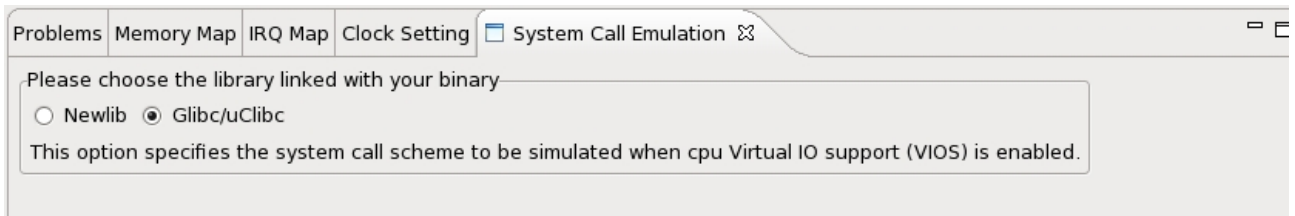
A terminal window titled 'wangshun@lyra:/AndesTech/AndeSight14/GDBAgent'. The window contains the following text:

```
[wangshun@lyra GDBAgent]$ ./GDBAgent -v
GDBAgent version 1.4 - 2010-02-08
(GDBAgent) Port range 9898-9998 from GDBAgent.ini.
(GDBAgent) Read license setting from "GDBAgent.ini": 10098@leo.
(GDBAgent) Use license setting: 10098@leo;10098@leo.
(GDBAgent) Assign GDBAgent port=9898
(GDBAgent) Waiting for connection...
█
```

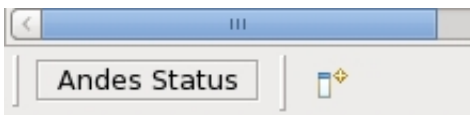
第二步：导入和上面的 Toolchain 相匹配的 vep 文件，如 Andes-demo.vep(一定要可以匹配的)，然后编辑这个文件，到 VEP config 界面设置 cpu 选项,设置 Data endianness 为 little endian，然后到 vep config 模式下修改 System Call Emulation 的 link library 为 Glibc。



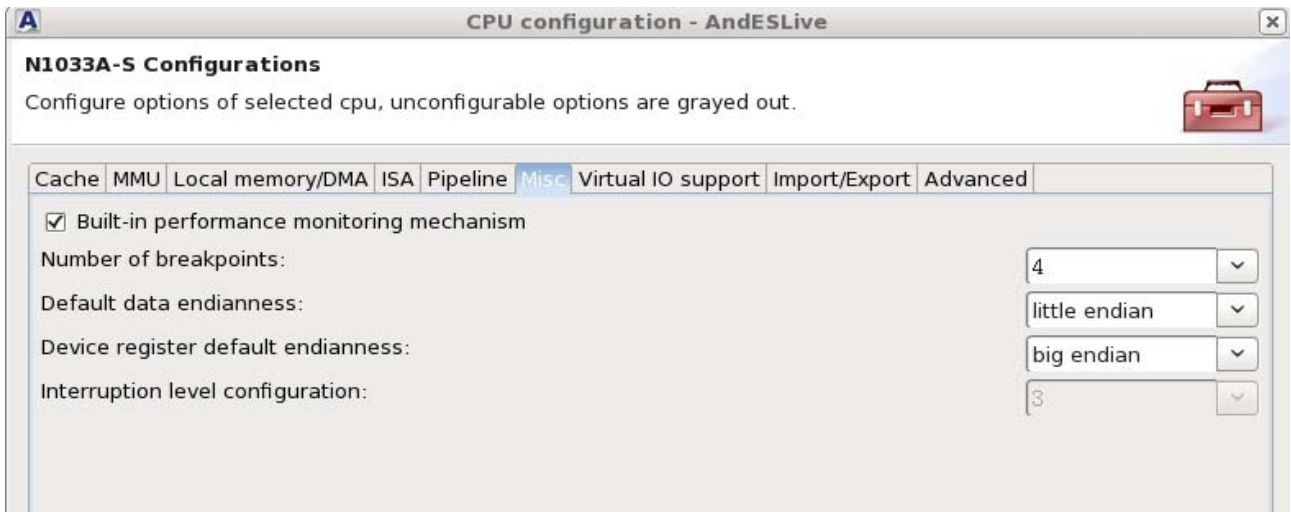
下图是将 System Call Emulation 的 link library 改成 Glibc/uClibc



下图是将 Data endianness 为 little endian，此操作是在 CPU 属性配置窗口修改，可以点击 vep 图中 CPU 的图标，然后选择左下方的图标：

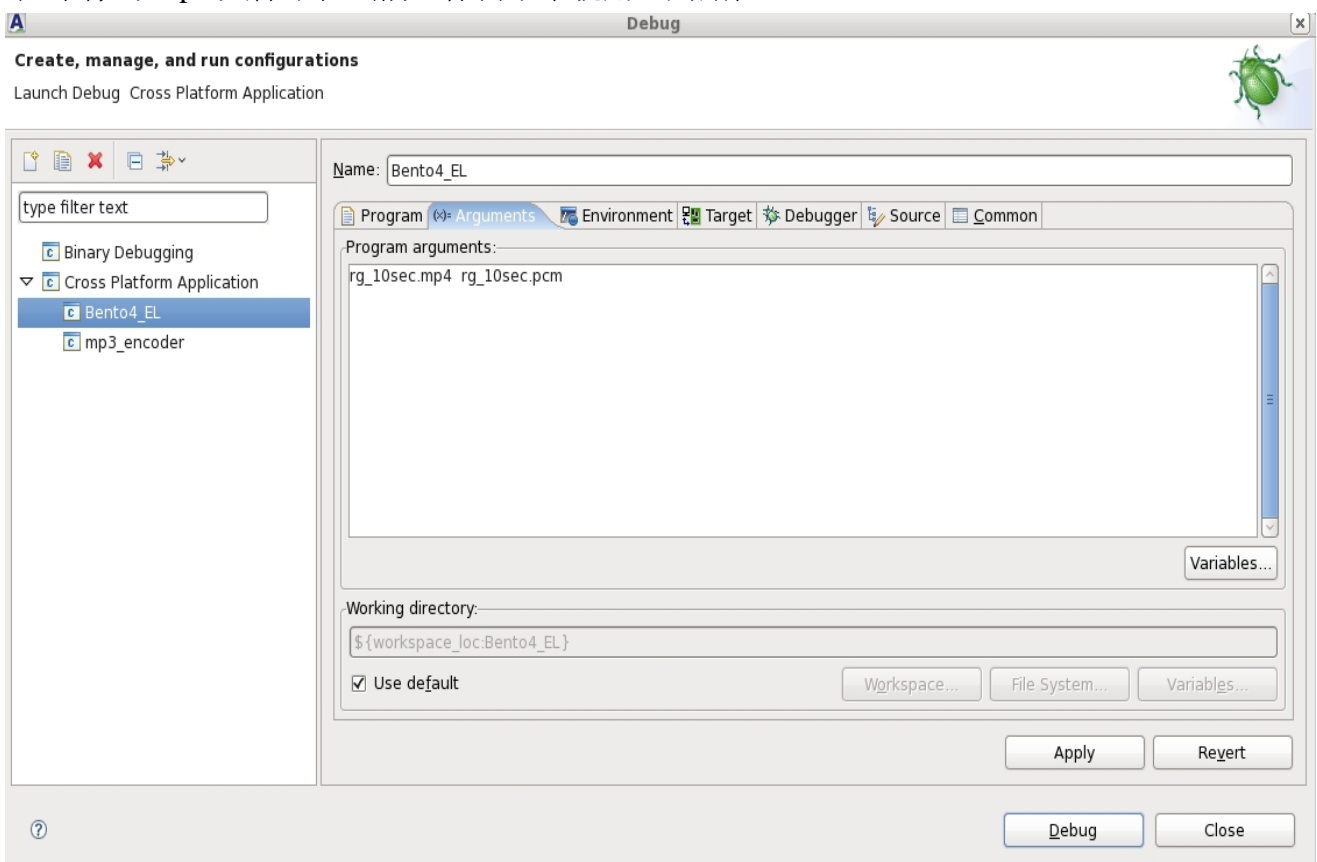


1. 点击那个有加号的那个图标，选择 Properties 选项，接着就如下图的 CPU configuration 界面，你可以配置 CPU 的参数。



第三步：找一个 mp4 的文件放到 AndeSight 的工作目录下，或者下面直接用绝对路径。

第四步：右击 Bento4-EL 工程选择 Debug As 选项，再选择下菜单中的 Debug 选项，在 Debug 对话框中，首先选择左边的 Cross Platform Application 菜单，然后选择 Bento4\_EL 选项，然后在 Arguments 选项内填写输入文件和输出文件，如：“XXX.mp4 XXX.pcm”，如果你的 mp4 文件不在当前工作目录下就用绝对路径。





### 3.4 播放 pcm 文档

在 linux 系统安装可以运行 pcm 歌曲的播放器，这里我们使用的是 mplayer，然后使用 mplayer 播放 pcm 歌曲。

```
[wangshun@lyra 04_28]$ mplayer -demuxer rawaudio -rawaudio rate=44100:channe
ls=2 /home/wangshun/Bento4_and_Melo/Melo-1.0.0/rg_6sec_new.pcm
MPlayer 1.0rc2-4.3.2 (C) 2000-2007 MPlayer Team
CPU: Intel(R) Core(TM)2 Duo CPU      E4500  @ 2.20GHz (Family: 6, Model: 15, S
tepping: 13)
CPUflags:  MMX: 1 MMX2: 1 3DNow: 0 3DNow2: 0 SSE: 1 SSE2: 1
Compiled for x86 CPU with extensions: MMX MMX2 SSE SSE2

Playing /home/wangshun/Bento4_and_Melo/Melo-1.0.0/rg_6sec_new.pcm.
rawaudio file format detected.
=====
Opening audio decoder: [pcm] Uncompressed PCM audio decoder
AUDIO: 44100 Hz, 2 ch, s16le, 1411.2 kbit/100.00% (ratio: 176400->176400)
Selected audio codec: [pcm] afm: pcm (Uncompressed PCM)
=====
[A0 OSS] audio_setup: Can't open audio device /dev/dsp: Device or resource b
sy
A0: [alsa] 44100Hz 2ch s16le (2 bytes per sample)
Video: no video
Starting playback...
A:  5.6 (05.6) of 6.0 (06.0)  0.1%
Exiting... (End of file)
```

### 3. 总结

以上所述方法即可将 Bento4 成功的移植到 andes 平台上。