Due: Friday, Nov. 10, 2006.

In this assignment, you will be expanding the hardware functions of your audio decoder. Specifically, the following additional two functions are to be implemented in Verilog: the delta decoder and the Huffman decoder.

Starting from your design for assignment #3, the delta decoder should be added to your existing scaling module (i.e., the scaling functions and delta decode functions are both incorporated into a single slave device on the APB, assigned to port 12). At the software level, the following routines should be supported by this module:

```c
void set_scale_factor(uint8 scale_factor);
/* set the scale factor */

int16 scale(int8 sample);
/* scale input sample, returning scaled sample */

void scale_vector(int8 samples[],
        int16 scaled_samples[], uint16 sample_count);

uint8 read_scale_factor(void);
/* return the current scale factor */

void set_delta_base(int16 delta_base);
/* set the base value for the delta decoder */

int16 read_delta_base(void);
/* return the current delta base */

int16 delta_decode(int16 sample);
/* perform delta decoding on input sample, updating
    delta base and returning delta decoded sample */

void delta_decode_vector(int16 samples[],
        int16 decoded_samples[], uint16 sample_count);
```

The first four routines above have the same function as in assignment #3. The `set_delta_base()` routine will write the value of `delta_base` to your slave device. The `read_delta_base()` routine will return the current delta base, which is altered for each sample that is decoded. The `delta_decode()` routine performs the actual decoding operation, writing `sample` to the slave device and reading the decoded sample from the slave device. The routine `delta_decode_vector()` performs the delta decoding function on a vector of size `sample_count`. 
The Huffman decoder should be an independent device on the APB (at port 13). Your team is responsible for designing its own low-level software interface (analogous to the above set of functions for scaling and delta decoding) as well as its own hardware/software interface for the Huffman decoder function. Note a new requirement for the variable length decoding function is that the number of output samples is not fixed with the number of input bits (or words). Your hardware/software interface as well as low-level software interface must deal with this fact.

Also, the file format has been altered to explicitly indicate termination. The last frame (which will not contain any samples) will indicate that it is the last frame by storing the value 0 in the `encoded_frame_size` field of the header.

On Tuesday, Oct. 24, each group must give a 10 min. PowerPoint presentation that describes the following design decisions made by your group:

1. What is the hardware/software interface for your scaling delta decoder module (i.e., what operations will take place for reads and writes from addresses within the scope of port 12)?
2. What is the low-level software interface for your variable length decoder?
3. What is the hardware/software interface for your variable length decoder?
4. What is the high-level architectural design for your variable length decoder (i.e., how do you plan to perform the variable length decoding operation in hardware)?

One of the two group members should give this presentation, but both members should be involved in deciding what will be presented. Note that the group member that does not present on Oct. 24 will make a presentation later in the semester as part of the next assignment.

Once built, test your two subsystems using test software you design to exercise all of the interfaces. Follow this with the complete design driven by the test data from `input03.h`, `input04.h`, and any additional files provided by us prior to the due date.

For both hardware devices, you are required to design a Verilog testbench that mimics the processor side of the APB. It should exercise your hardware with the same functional requirements as the software test, although you may decrease the number of tests performed if you wish (i.e., smaller number of samples). Be sure and exercise your hardware modules: 1) using the testbench, and 2) on the actual hardware.

Time the operation of both the software and hardware versions with text output disabled.

Turn in commented code (C and Verilog), timing results (including a description of how you performed the timing), output data, and simulation screenshots. Include enough description so we can tell what is going on.