University of Toronto
ECE532 Digital Hardware
Real Time MP3 Decoder Implentation Using MicroBlaze Soft Processor

Version 1.1 For EDK 6.2i 12/28/2004

Acknowledgement

This document was created to accompany the example MP3 design we received and adopted. Many thanks to the anonymous donor.

We do our best to keep it working, but no promises.

Goals

- Get a better understanding of MicroBlaze systems by using this example design, which has instructions stored in on-chip BRAMs and data stored in the ZBT external memory.
- Get an idea of how audio works on the Multimedia development board.

Requirements

Access to EDK 6.2i, the Xilinx Multimedia development board, and speakers or headphones plugged into the Multimedia board headphones jack.

Preparation

- Copy pc/courses/532/labs/sampledesigns/MicroBlaze_MP3Decoder_Rel_VerECE532.zip to your working directory and unzip it. MAKE SURE THIS PATH HAS NO SPACES!
- Read the README file in the unzipped directory.
- Read the section on user-specified Makefiles in the EST Guide -> Xilinx Platform Studio (XPS) -> Flow Tool Settings and Required Files.
- Make sure your speakers/headphones are plugged into the Multimedia board and have power, if needed. Make sure the speaker volume is not so high that it will make you go deaf or give a heart attack to your TAs.

Note

- Although this design is supposed to work with EDK 6.1i, we have not had success with that version of the tools. This design was found to work with EDK 6.2i Service Pack 1, but started mis-behaving slightly with EDK 6.2i Service Pack 2. See the Step-by-step section for more details.
- Also, the opb_zbt_controller officially has Xilinx IP OBSOLETE status with EDK 6.2i service pack 2. To get around this, the pcore directory for this core was copied and added to the pcores/ directory of the example design. Then the MPD file was modified to describe it as having DEPRECATED status instead. Otherwise, XPS would claim that the project has errors.

Step-by-step

You will use XPS to play your own MP3 song. This can be done via the GUI, by a slightly different procedure than what is in the README file.

- Change directory to the directory you extracted for this example design.

- Change directory to the XPS_DESIGN_2V2000 directory.
- Copy system.make as system_mp3.make. The makefile that is provided has been modified from the XPS version to use a bootloop code to initialze processor memory during bitstream download. Why do you think we need to rename the makefile?
- If you are using EDK 6.2i with Service Pack 2, XPS will complain unless you do the following: Go to your pcores directory. Rename bufg_block_v2_00_a to bufg_v2_00_a. Open the directory and go to the data directory. Rename bufg_block_v2_1_0.mpd and bufg_block_v2_1_0.pao to bufg_v2_1_0.mpd andbufg_v2_1_0.pao. Open the PAO file for editing and change bufg_block_v2_00_a to bufg_v2_00_a. Go back to the main project directory. Open the MHS file and change the lines "BEGIN bufg_block" to "BEGIN bufg". Save the files you edited.
- Open the 2v2000 design in XPS and allow it to rev it up. Read the dialog box that pops up to see what takes place during a rev-up. Click Yes to continue.
- You will get some errors during the parsing of some MPD files. The Output tab will tell you where to find the files. Open them and comment out the OPTION SPECIAL lines. Save the modified MPD files.
- In GUI set the device to 2v2000 ff896 -4. This will cause it to change in the system.xmp file.
- In Options -> Project Options, specify system_mp3.make as the custom makefile.
- Modify system_mp3.make to follow the new 6.2 convention of including system_incl.make. (You can read more about this in the EST Guide.) Comment out the variable definitions in system_mp3.make but keep MY_BLAZE_OUTPUT and MY_BOOTLOOP. Replace all instances of _DO_SCRIPT with _SIM_SCRIPT. This is important because it makes sure that the makefile reflects changes to options and settings made in XPS.
- Compile and download everything using the system_mp3.make makefile. Make sure that you have lots of free disk space, otherwise compilation may fail unnecessarily. This would be sad because generating the netlist and bitstream takes quite a while.
- Note that with EDK 6.2i Service Pack 2, Timing fails. What was the constraint that failed?.
- Open xmd.
- Make sure user input switches are in the positions 1: off, 2: on
- Connect to the MicroBlaze debug module: mbconnect mdm.
- Provide the size of the MP3 file at location 0xfa800000: mwr 0xfa800000 0x1ec00
- Provide the MP3 data to download at location 0xfa800004 (The length of the song should be less than 0xFFFFF): dow -data ../MP3Streams/classic2.mp3 0xfa800004
- Download Microblaze executable (This downloads code to local memory and data to instruction memory): dow my_blaze/code/executable.elf
- Execute the program from location 0x0: con 0x0
- If you are running EDK 6.2i Service Pack 1, your song should now play on the headphone jack of the Multimedia board. If you are running EDK 6.2i Service Pack 2, then the song should still play in spite of the Timing constraint violation.