EECS 222C
System-on-Chip Software Synthesis
Spring 2013

# Assignment 7

**Posted:** May 24, 2013
**Due:** May 31, 2013 at 12pm (noon)

**Topic:** Re-evaluate the ARM7TDMI Processor as a candidate for an implementation of the MP3 Decoder model with dedicated hardware accelleration

## 1. Setup:

While this assignment builds upon the previous assignments, we will switch to the "latest" version of SCE now, as discussed in Lecture 8. As a consequence, we cannot directly reuse the data in your **hw3** directory. Instead, we have to repeat some of the previous steps, now in a new directory **hw7**, as follows:

```
mkdir hw7
cd hw7
```

We will start again by unpacking the provided SpecC model into your **hw7** directory. We also will use two new testbench files for better timing reporting, as discussed in Lecture 8. Setup up your source files as follows:

```
gtar xvzf ~eecs222/EECS222C_S13/mad_SpecC.tar.gz
cp ~eecs222/EECS222C_S13/mp3monitor.sc .
cp ~eecs222/EECS222C_S13/mp3stimulus.sc .
```

As before, we create a symbolic link to the MP3 input streams and setup a directory with the reference PCM streams, as follows:

```
ln -s ../hw1/mad_C/testStream
mkdir reference
cp ../hw1/mad_C/spot1.pcm reference/
cp ../hw1/mad_C/spot1_3K.pcm reference/
cp ../hw1/mad_C/classic1.pcm reference/
vi Makefile
```

In the last step, edit the Makefile so that the copied test streams are used as data files for the 'make test' targets. Specifically, set:

```
TESTSTREAMS = spot1_3K classic1 spot1
```

From now on, we will use the "latest" System-on-Chip Environment (SCE) version from 2012 (and later). Run the setup script as follows:

```
source /opt/sce/bin/setup.csh
```

To avoid incompatibility problems with the previous SCE versions, delete the **.sce** directory in your home directory before starting the new SCE:

```
rm –rf ~/.sce
```

To check that the specification model is setup correctly, do the following:

```
make clean
make
testbench testStream/spot1_3K.mp3 spot1_3K.pcm 8
```

Note that we add the expected number of frames (8) for the monitor as the 3rd argument to the simulator call (as discussed in Lecture 8).

If the above works fine, you can continue from here with the graphical SCE IDE.

```
sce &
```

Again, we create a suitable SCE project file with updated (as per Lecture 8) settings, as follows:

- **Project->New**
- **Project->Settings**
  - Set include path to "**.**" (current directory)
  - Set libraries to "**-xl huffman.o**"
  - Set defines to "**NDEBUG**" (no assertions, no debugging in DUT)
  - Set both verbosity and warning level to 2
  - In the Simulator tab, set the simulation command as follows (in a single line!):
    **./%e testStream/spot1_3K.mp3 spot1_3K.pcm 8 && diff reference/spot1_3K.pcm spot1_3K.pcm**
- **Project->SaveAs "mp3.sce"**

Next, you can load, compile and simulate your MP3 Audio Decoder model in SCE, as follows:

- **File->Import "testbench.sc"**
- **Project->AddDesign**
- Right-click on **testbench.sir** in the project window, and **Rename** the model to **Spec**
- **Validation->Compile**
- **Validation->Simulate**

- **Validation->Profile**
- **Project->Save**

## 2. Task: Refine the MP3 Decoder in SCE down to an ISS Model with dedicated hardware acceleration

We will continue the design flow now and step-wise refine the model down to a cycle-accurate Instruction Set Simulation (ISS) model. In contrast to the software-only architecture of Assignment 6, we will now integrate dedicated hardware units for acceleration of the decoding process.

In the following instructions, several steps are for you to make your own decisions. The goals is to design a cycle-accurate system platform model that meets the timing constraint with minimal resources. Note that for this assignment, it is OK if you just reach down to a simulatable ISS model. You may adjust certain parameters in another system design cycle later. For the final report, however, meeting the timing constraint is necessary, and the closer you get to a cost effective architecture, the better.

Detailed instructions on the model refinement steps are listed below. The same instructions are also available for your convenience on the server in file: **/home/eecs222/EECS222C_S13/Assignment7.txt**

```
Step 1: Architecture Refinement
  -> start from the profiled "Spec" model
  -> set top-level: MP3Decoder mp3decoder
  -> allocate 1 ARM_7TDMI CPU
        -> set clock period as desired (e.g. 20000ps = 50MHz)
        -> default port parameters (i.e. 20000ps bus period)
        -> name the PE "ARM7"
        -> if you like to adjust the profiling weight tables
           (as discussed in Lecture 8), click "Tables" and
           change the factor on the top left of the
           Computation tab
  -> allocate 2 custom hardware I/O units of type HW_Virtual
        -> keep clock frequency at 100MHz
        -> name the PEs "MP3_IN" and "PCM_OUT"
  -> allocate up to 6 custom hardware units of type HW_Standard
        -> keep clock frequency at 100MHz
        -> name the PEs "HW1", "HW2", etc.
  -> Synthesis->ShowChannels
  -> map the behaviors to the target architecture
        -> map MP3Decoder to ARM7
        -> map Mad_Stimulus to MP3_IN
        -> map channel stream_in to MP3_IN
        -> map Mad_Monitor to PCM_OUT
        -> map channel pcm_out to PCM_OUT
        -> map suitable critical blocks to the allocated HW units
           (e.g. Imdct to HW1 and Imdct_0 to HW2)
```

```
            Hint: in order to map the multiple instances
            of a behavior (e.g. Imdct) to different HW units,
            you need to select the behavior and then
            RightClick->Isolate (the two IMDCT instances
            then become Imdct and Imdct_0)
      -> Validation->Evaluate
      -> Validation->ShowEstimates
      => Look up the estimated execution time for the MP3 decoder
         on your platform, calculate the decode time per frame,
         and adjust and repeat the above allocation and mapping
         steps as needed!
  -> Synthesis->ArchitectureRefinement
         -> default options
  -> Project RightClick->Rename "Arch" (or similar)
  -> Validation->Compile
  -> Validation->Simulate
     => Note the decode time per frame in your time table!
        (it ignores sequentialized parallelism)

Step 2: Scheduling Refinement
  -> Synthesis->ScheduleBehaviors
     -> select Dynamic Scheduling "None" for ARM7
       (not needed for MP3 decoder)
        -> select ARM_7TDMI_20000_0_ARM7, at top-level
RightClick->SerializeTree
     -> leave I/O units alone (MP3_IN, PCM_OUT)
     -> if needed, perform static scheduling for your HW units
  -> Synthesis->SchedulingRefinement
         -> default options
  -> Project RightClick->Rename "Sched" (or similar)
  -> Validation->Compile
  -> Validation->Simulate
     => Note the decode time per frame in your time table!
        (it ignores communication)

Step 3: Network Refinement
  -> Synthesis->AllocateNetwork
    -> for Busses, rename pre-allocated AMBA bus to "MainBus"
    -> if desired, add extra busses as needed
       (e.g. Simple.HardwareBus)
    -> for CEs, none are needed (nothing to do)
    -> for Connectivity, connect
        - ARM7,PortA to MainBus as master0
        - MP3_IN,Port0 to MainBus as slave1
        - PCM_OUT,Port0 to MainBus as slave2
        - your allocated HW units as slave3, slave4, etc.
    -> if needed (for extra busses), "Show Channels" and map
       them to the busses
  -> Synthesis->NetworkRefinement
         -> default options
  -> Project RightClick->Rename "Net" (or similar)
  -> Validation->Compile
```

```
  -> Validation->Simulate
     => Note the decode time per frame in your time table!
        (it assumes infinite network speed)

Step 4: Communication Link Refinement
  -> Synthesis->AssignLinkParameters
    -> simply RightClick->Autofill all addresses
    -> use polling or interrupt-based communication as desired
       (LEON3 ISS does not support interrupts, ARM7 should be
fine)
  -> Synthesis->CommunicationRefinement
        -> choose Transaction-level model, default options
  -> Project RightClick->Rename "TLM" (or similar)
  -> Validation->Compile
  -> Validation->Simulate (be patient!)
     => Note the decode time per frame in your time table!
        (it includes TLM communication delay now)

  -> go back and select "Net" model again
  -> Synthesis->CommunicationRefinement
        -> choose Pin-accurate model, default options
  -> Project RightClick->Rename "PAM" (or similar)
  -> Validation->Compile
  -> Validation->Simulate
     => Note the decode time per frame in your time table!
        (it includes PAM communication delay now)

Step 5: Software synthesis (C code generation)
  -> continue from the "TLM" model generated in Step 4
  -> Synthesis->CCodeGeneration
    -> choose "C code reintegration"
    -> leave other options as per default
  -> Project RightClick->Rename "TLM_C" (or similar)
  -> Inspect generated C code
    -> in shell where SCE was started:
       % vi ARM7/ARM7.h
       % vi ARM7/ARM7.c
       % vi ARM7/Makefile
    -> in behavior tree, select leaf
       mp3decoder->ARM7->hw_TLM->hal->ARM7
       -> View->Source
  -> Validation->Compile
  -> Validation->Simulate
     => Note the decode time per frame in your time table!
        (it includes TLM communication as before)

Step 6: ISS integration
  -> continue from the "PAM" model generated in Step 4
  -> Synthesis->CCodeGeneration
    -> choose "Instruction-set simulation"
    -> leave other options as per default
  -> Project RightClick->Rename "ISS" (or similar)
```

```
-> Adjust generated Makefile
  -> in shell where SCE was started:
     % cd ARM7
     % cp ../huffman.h .
     % cp ../huffman.c .
     % vi Makefile
       -> replace "USR_SRC := ARM7.c"
          with    "USR_SRC := ARM7.c huffman.c"
       -> replace "USR_LFLAGS := huffman.o"
          with    "USR_LFLAGS :="
-> Cross-compile the generated code for the ARM7 processor
  -> in shell where SCE was started:
     % make
-> Validation->Compile
-> Validation->Simulate
   => Note the decode time per frame in your time table!
      (note that the frame delays may vary at ISS level)
```

As deliverables for this assignment, submit the final **`ISS.sir`** model and note the estimated/simulated frame delays in a table as follows:

| Refinement Step | Model | Decode time per frame |
|---|---|---|
| Profiling estimation | Spec | |
| Architecture Refinement | Arch | |
| Scheduling Refinement | Sched | |
| Network Refinement | Net | |
| Transaction-Level Refinement | TLM | |
| C Code Generation | TLM_C | |
| Pin-Accurate Refinement | PAM | |
| Instruction Set Simulation | ISS | |

For submission, convert your table into a PDF file. Name the PDF file **`ARM7plusHW_Evaluation.pdf`**, place it into your **`hw7`** directory, and make it readable for the submission script. Make the final **`ISS.sir`** file readable as well.

```
chmod 644 ARM7plusHW_Evaluation.pdf
chmod 644 ISS.sir
```

Use exactly these filenames, otherwise you can't submit.

**3. Submission:**

For this assignment, submit the following deliverables:

```
ARM7plusHW_Evaluation.pdf
ISS.sir
```

The files should be placed in your **`hw7`** directory. Then, in its parent directory, enter **`turnin`**.

As in the previous assignments, the **`turnin`** command will locate the deliverables and allow you to submit them *before the deadline.*

Again, you can submit at any time before the deadline, *but not after!* You can also submit as many times as you want. Newer submissions will overwrite older ones.

*Late submissions will not be accepted!*

--
Rainer Doemer (EH3217, x4-9007, doemer@uci.edu)