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

Assignment 3

Posted: April 26, 2013
Due: May 3, 2013 at 12pm (noon)

Topic: Analyze the SpecC model of the MP3 Audio Decoder

1. Setup:

Using your Linux account, login to one of the hosts:

```
gamma.eecs.uci.edu
omicron.eecs.uci.edu
```

Create a new directory named **hw3** (in parallel to your **hw1** and **hw2** directories) and work inside this directory. Name your directory and all files for submission exactly as shown in these instructions. Otherwise you will not be able to submit your deliverables!

We will use again the SpecC tools as in the previous assignment. Run the setup script, as follows:

```
source /opt/sce-20100908/bin/setup.csh
```

As discussed in the lectures, we will not create the MP3 audio decoder specification model ourselves in this class (that's part of EECS 222A). Instead, a suitable SpecC model ready for system design will be provided to you in this assignment. Unpack the provided SpecC model into your hw3 directory, as follows:

```
cd hw3
gtar xvzf ~eecs222/EECS222C_S13/mad_SpecC.tar.gz
ls
```

For the SpecC model, we will reuse the same test streams as in the original MP3 source code of Assignment 1. The decoded PCM streams from Assignment 1 will serve as "golden" reference streams so that we can validate the output of the SpecC decoder model by comparing the PCM files.

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
```

2. Task A: Validate the SpecC model of the MP3 Decoder

To compile and execute the SpecC model, do the following:

```
make clean
make
testbench testStream/spot1.mp3 spot1.pcm
```

Note that the provided Makefile automatically compiles all components of the MP3 design including the needed test bench. The latter then serves as the name of the executable which you can run to simulate the MP3 design.

To validate that the SpecC model has decoded the test stream correctly, we can simply compare it against the golden reference PCM stream:

```
diff spot1.pcm reference/spot1.pcm
```

If this ‘diff’ doesn’t report any differences, the decoded stream matches the reference.

As a convenience, the provided Makefile also supports a test target that performs this validation automatically for all test streams:

```
make test
make test1
```

The latter alternative performs the validation only for the first stream.

3. Task B: Analyze the specification model of the MP3 Decoder

To study and analyze the provided SpecC model, we can utilize some of the SpecC command line tools that are part of the SpecC compiler package.

First, create a suitable SIR file of the entire MP3 design, as follows:

```
make testbench.sir
```

Next, we can quickly obtain and view some statistics about the model:

```
sir_stats testbench.sir
sir_stats –a testbench.sir
```

A hierarchy tree of the model can be easily obtained as well. For example:

```
sir_tree -blt testbench.sir
sir_tree -blt testbench.sir Mad_Decoder
```

To avoid the handling of the multiple source code files of the model, we can also generate code of a “clean” single-file SpecC model that describes the entire design model:

```
scc testbench -sir2sc -vv -sn -sl –psi
    -o testbench_gen.sc
```

The same can be done using the Makefile as well, as follows:

```
make testbench_gen.sc
```

After the code generation, you can inspect the generated source code and compare it against the original source files:

```
vi testbench_gen.sc
```

Of course, this single-file model can be compiled and executed as well:

```
scc testbench_gen -vv -xl huffman.o
testbench_gen testStream/spot1.mp3 spot1.pcm
diff spot1.pcm reference/spot1.pcm
```

Note that **`huffman.o`** needs to be supplied as an extra object file to the linker. (This file contains compiled code that the SpecC compiler is not supporting.)

4. Task C: Determine potential parallelism for MPSoC implementation

Study, examine, and analyze the SpecC model for the potential of parallel implementation in the desired MPSoC.

Answer the following questions:

4. Is there any parallelism specified in the model?
   If so, where?
   - Find all concurrent behaviors (behaviors that execute in parallel)
   - For each parallel behavior, note
     - Name of the concurrent parent behavior
     - Names of the parallel executing child behaviors
5. Which of the concurrent behaviors identified above
   are candidates for parallel implementation in a MPSoC?
   - In one sentence (per concurrent behavior), explain why or why not the behavior can be implemented with parallel instances in the desired MPSoC for an MP3 player.

Create a PDF document with your answers and name your file
`MP3_Parallelism.pdf`
Use exactly this filename, otherwise you can't submit it.

5. Submission:

For this assignment, submit the following deliverable:

`MP3_Parallelism.pdf`

Place this file into your **`hw3`** directory. Then change into the parent directory of the **`hw3`** directory and enter **`turnin`**.

As in Assignment 1, the **`turnin`** command will locate the deliverable this assignment asks for and allow you to submit it *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)