TSTE87 Laboratory Work – Lab 5

Oscar Gustafsson, Kenny Johansson, and Erik Bertilsson

- Student name: .....
- Student personal id: .....
- Passed: .....

Goals

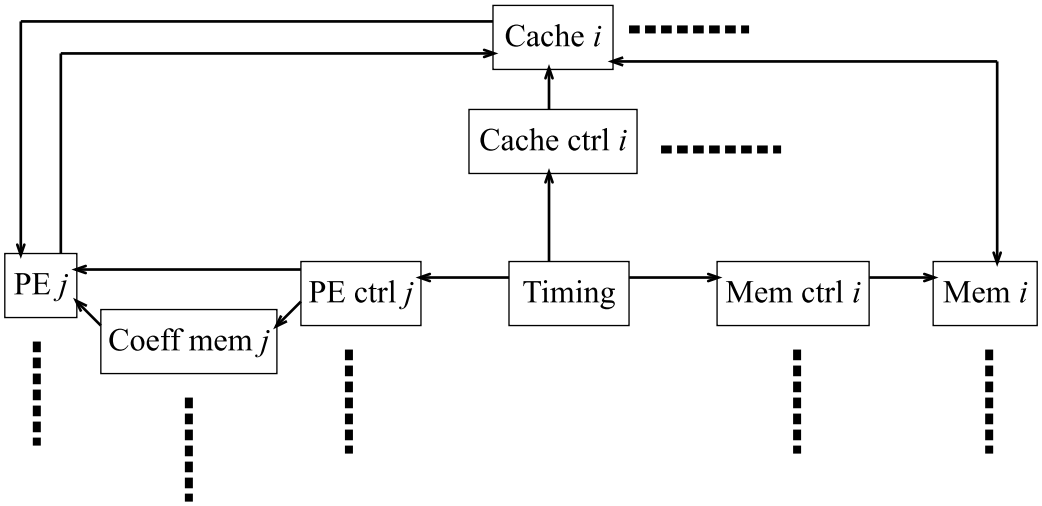
Generate the architecture and an HDL description. Investigate the effect of resource limitation using automatic scheduling.

Preparations

1. Read Chapters 8, 9, and 11 in Wanhammar, DSP Integrated Circuits.
2. Read "Additional commands in the DSP toolbox" below.

Additional commands in the DSP toolbox

At the moment it is not possible to automatically generate a fully functional implementation, including processing elements. However, control and memory control units can be generated. The general architecture is as depicted below. Hence, there is a central timing unit which basically is a counter that keeps track of where in the scheduling period the execution is currently. Each memory has a memory controller which generates addresses and read/write signals based on the current time. Similarly, each memory has a corresponding cache memory. Each processing element has a controller, which generates start signals to the PE and optionally generates addresses to the coefficient memories.



A number of commands are available to generate VHDL code for the controllers.

To generate the timing information, the following command can be used

```
generatetimingcontroller(scheduletime)
```

Note that the schedule time can be obtained from a schedule as

```
scheduletime = schedule(1,1)
```

The default component name is `timingcontroller` and the default file name is `timingcontroller.vhdl`. A different component and file name can be selected by a second argument as

```
generatetimingcontroller(scheduletime, componentname)
```

The standard behavior is to generate a binary counter. However, note that there are other counters that may be better from e.g. speed and power point of view. The output is a vector, `state`, which contains the current state. Inputs to the timing controller is a clock and a reset signal, `clk` and `reset`, respectively. For the memory controller, the command `generatememorycontroller` is available which is used as

```
generatememorycontroller(cellassignment, number)
```

where `cellassignment` is a cell assignment and `number` is an identifying number. The component will be named `memorycontroller#` where `#` is replaced by the identifier.

Similarly, a PE controller can be generated as

```
generatepecontroller(peassignment, schedule, number, bits)
```

where `peassignment` is one PE assignment, i.e., not the list of all PE assignments (use curly brackets to obtain one element from a list), and `bits` denotes the total number of bits used to represent the coefficients. If required, `generatepecontroller` will also generate one or more coefficient memories (for multipliers, adaptors etc). The component will be named `pecontroller#`.

Automatic Scheduling

The command `evaluation` can be used to obtain a schedule, based on list-scheduling, using a given number of processing elements as

```
schedule = evaluation(sfg, resources, multexecutiontime, onlyschedule}
```

where `sfg` is the signal flow graph and `resources` is the available functional units on the form `[multipliers, adders, subtracters]`. This algorithm assumes that latency and execution time are the same. The execution time for addition and subtraction is one time unit and the execution time for multiplication can be set by the input argument `multexecutiontime`. If only a schedule is to be generated, i.e., no information about the overhead hardware requirements (registers and multiplexers) is desired, the `onlyschedule` parameter should be set to one.

Tasks

1. The first task is to generate VHDL descriptions of the memory and PE controller using information from Laboratory work 4, and simulate the behavior of these.

   - Copy the files `interp.do` and `interpolator.vhdl` from `/courses/TSTE87/labs/lab5` to a working directory.
   - Generate VHDL for the timing controller. Place the file in the working directory.
   - Generate VHDL for the memory controllers, number them 1 and 2. Place the files in the working directory.
   - Generate VHDL for the PE controllers, number them 1, 2, 3, and 4. Note that the number of bits should be one integer bit plus the number of fractional bits. Place the files in the working directory.
   - Edit the beginning of the file `interpolator_tb.vhdl` to set the correct constants for your design (see source code).
   - Compile and simulate the code. This is done with the terminal command `vsim -do interp.do &` note that you must add the module mentor (`module add mentor`).
   - For one memory controller and one PE controller, check that the outputs corresponds to the expected.
   - For one memory variable, explain in detail when the data is read and written and what the values of the address, enable, and read/write signals are at those time instances. Furthermore, identify from which PE the data was obtained and which PE will consume it.

     .....

   - What are the coefficient values used in the two PE operations associated to the memory variable discussed above, according to the simulation?

. . . . . .

- To end up with a fully functional implementation we have to add processing elements, memories and cache memories (with controller). However, this is not included here.

2. In this task we will use the allpass filter of the interpolator filter from laboratory work 2 (`/courses/TSTE87/labs/lab2/interpolatorallpass.m`)

   - The latency and execution time are assumed to be equal in this task. Furthermore, all basic operations (addition, subtraction, and multiplication) are assumed to have an execution time of one time unit.
   - Study the schedule obtained from `getinitialschedule` using an execution time corresponding to the critical path of a twoport adaptor. See previous laboratory works to find the number of basic operations in the critical path.
   - Transform the twoport operations to lower level operations using the command `flattensfg`. Study the schedule and compare with the previous schedule where twoport operations was used. Which low level operations correspond to which twoport operations? What is the required number of processing elements?

     . . . . . .

     . . . . . .

   - Add the path to the `evaluation` command in MATLAB. This can be done by

     ```
     addpath /courses/TSTE87/labs/evaluation/
     ```

   - Study the schedule obtained by `evaluation` using "infinite" resources. Compare to the previous schedule. Comments?

     . . . . . .

   - Study the schedule obtained using resources corresponding to one twoport operation. Comments?

     . . . . . .

   - What is the required overhead hardware, i.e., registers and multiplexers (just count the total number of multiplexer inputs)? Note that information about this is printed in the MATLAB command window.

     . . . . . .

   - Pipeline the allpass filter so that the critical path only includes two twoport operations.

- Compare the schedules obtained by `getinitialschedule` and `evaluation` (using resources corresponding to one twoport operation). Comments?

.....

- What is the required overhead hardware?

.....

- Study a schedule obtained using resources corresponding to two twoport operations. Comments?

.....

- What is the required overhead hardware?

.....

- In which of the three schedules for the pipelined filters is the hardware used most efficiently? What is the utilization ratio for this case?

.....