TSTE87 Laboratory Work – Lab 4

Oscar Gustafsson, Kenny Johansson, and Erik Bertilsson

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

Goals

Perform resource allocation and assignment.

Preparations

1. Read Sections 7.8–7.10 in Wanhammar, DSP Integrated Circuits.
2. Read "Additional commands in the DSP toolbox" below.
3. Make a valid PE assignment for the schedule derived in Lab 3. No more than 4 PEs maybe used.

Additional commands in the DSP toolbox

The latency and execution time for operands already in a schedule can be changed using the command `updatepetiming` as

```
 schedule = updatepetiming(schedule, [latency executiontime],
                           operandname)
```

to update all operands of a certain type or

```
schedule = updatepetiming(schedule, [latency executiontime],
                          operandname, operandnumber)
```

to update a certain operand. Note that no checking that the resulting schedule is valid is done. The memory variables for a schedule can be extracted using the following command

```
mv = extractmemoryvariables(schedule)
```

The memory variables can then be printed or plotted using

```
printmemoryvariables(mv)
```

```
plotmemoryvariables(mv)
```

A schedule may require that variables are read or written concurrently in the schedule which may not be supported by the memories used. Hence, the memory variables must be partitioned into several different memories. This can be done by the command `memorypartitioning` as

```
[mvlist, iv] = memorypartitioning(mv, readports, writeports,
concurrentports)
```

where `readports`, `writeports`, and `concurrentports` are the number of ports of different type for the memory. Usually, all these variables are 1. The result, `mvlist= {mv1, mv2, ...}`, is a list of several memory variable sets, where each set corresponds to one memory. Set i can be obtained by `mvi = mvlist{i}` (note the curly brackets). `iv` contains all direct PE to PE transfers (memory variables with time 0). To each set of memory variables, we can apply the left edge algorithm as

```
cellassignment = leftedgealgorithm(mvi)
```

The resulting cell assignment can be printed and plotted with

```
printcellassignment(cellassignment)
plotcellassignment(cellassignment)
```

An initial PE assignment can be obtained with

```
peassignment = getpeassignment(schedule)
```

The `peassignment` variable is a list, where each set contains an array of the form `[operandid number1 number2 number3 ...]`. Hence, it is easy to change the PE assignment if required. The PE assignment can be obtained in a more readable form with

```
printpeassignment(peassignment)
```

To analyse the interconnection between a memory and the PEs, the following command can be used

```
printinterconnection(mvi, peassignment)
```

In order to reduce the number of switches required in the interconnect, the port assignment of the twoport adaptors can be modified. It is possible to flip a twoport and still obtain the same functionality, if the adaptor coefficient is changed. This can be done with the command

```
schedule = fliptwoport(schedule, number)
```

Tasks

1. To test the resource allocation, a small schedule is available in the file

   `/courses/TSTE87/labs/lab4/simpleschedule.m`

   This is a simple filter, where adders have a latency and execution time of one time unit and multipliers have a latency and executiontime of two time units.

   - Extract and print/plot the memory variables.
   - Partition the memory variables using memories with one read port and one write port, assuming that it is possible to read and write concurrently. How many memories are required? How can the number of memories be reduced?

     ....

   - Apply the left edge algorithm to each memory. How many memory cells are required for each memory?

     ....

   - Determine the number of adders and multipliers required.

     ....

   - Assume that pipelining is introduced in the multipliers so that the execution time is halved. Change the execution time of the multipliers to one time unit.
   - Determine the number of adders and multipliers required.

     ....

   - Analyse the interconnect between the memories and processing elements. Draw by hand an interconnect schematic for memories and PEs.

2. In this task we will use the schedule of the single rate realization of the interpolator filter from Task 2 of Laboratory work 3. Make sure that you save all results!

   - To avoid concurrent reads and writes, increase the resolution of the schedule by a factor of two and decrease the latency of each twoport by one time unit. This should result in a latency of 15 time units and an execution time of 8 time units.
   - Extract the memory variables and partition the memories using memories with one read and one write port with at most one concurrent memory access. How many memories are required?

     ....

- Apply the left edge algorithm and determine the number of cells required.

  . . . .

- Perform processing element assignment and determine the number of PEs.

  . . . .

- Analyse the interconnect and try to minimize the number of PE ports connected to each memory. Comments?

  . . . .