TSTE87 Laboratory Work – Lab 3

Oscar Gustafsson, Kenny Johansson, and Erik Bertilsson

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

Goals

Find an efficient schedule for the interpolation filter.

Preparations

1. Read Chapter 6 and Sections 7.1–7.6 + 7.10 in Wanhammar, DSP Integrated Circuits.
2. Read "Additional commands in the DSP toolbox" below.

Additional commands in the DSP toolbox

To obtain an inital schedule, based on ASAP-scheduling, the command `getinitialschedule` can be used as

```
schedule = getinitialschedule(sfg, timinginfo, scheduletime)
```

where `sfg` is the signal flow graph, `timinginfo` is obtained from `getdefaulttiminginfo` and `scheduletime` is the required schedule time. If no schedule time is given it will be set to the time of the critical path.

The latency and execution times of the different operations are determined by a structure containing the information. A default setting of 1 for both latency and execution time for all arithmetic operations (i.e. not inputs, constants etc.) is obtained from `getdefaulttiminginfo` as

```
timinginfo = getdefaulttiminginfo
```

The timinginfo can be changed based on the operand names as

```
timinginfo.constmult.latency = 2
timinginfo.twoport.executiontime = 4
```

Once the schedule is obtained, it is possible to view it using

```
printschedule(schedule)
```

and

```
plotschedule(schedule)
```

To obtain the possible change in start time for all operations you can use

```
printstarttimes(schedule)
```

To obtain the possible change in start time for one operation use `getstarttimes` as

```
[starttime, upperbound, lowerbound] =
getstarttimes(schedule, operandname, number)
```

If an operation is to be rescheduled it can be done by using `changestarttime` as

```
schedule = changestarttime(schedule, operandname, number, timedifference)
```

where `timedifference` is the amount of time to move the operation (positive or negative).

The scheduling time can be changed as

```
schedule = setscheduletime(schedule, scheduletime)
```

Note that no operations can start at or after the new schedule time. Also, the execution time and the recursive loops may inhibit decreasing the schedule time.

The resolution of the time scale can be changed by

```
schedule = updatetimescale(schedule, factor)
```

where `factor` is the number of times to increase the resolution (integer > 1).

Tasks

1. To test the scheduling functionality a simple SFG is available in the file

   `/courses/TSTE87/labs/lab3/simplesfg.m`

   - Compute an initial schedule and print it. Assume latency = execution time = 2 time units.
   - Print the possible changes in start times.
   - Reschedule so that at most one multiplication and one addition are executed in parallel, with a schedule time of at most six time units. Note that it is possible to move outputs.

2. In this task we will use the pipelined single rate realization of the interpolator filter from Task 3 of Laboratory work 2.

   - Obtain an initial schedule with a schedule time of six time units and print it. Assume a latency of 2 time units and an execution time of 1 time unit.
   - How many two-port processing elements are required for this schedule?

     ....

   - Reschedule the operations so that there are at most four operations starting in each time slot. Note that it may help to consider the connection between different two-ports (see Lab 2).
   - How many two-port processing elements are required for this schedule?

     ....

   - Increase the resolution of the time scale by a factor of four and reschedule the operations so that at most one operation starts in each time slot.
   - How many two-port processing elements are required for this schedule?

     ....

   - Save the final schedule! (Either using save or keep the file used to generate it)