Practical assignment ECA2:
Filter architecture design using Clash

Introduction

Below the set of homework exercises for ECA-2 regarding filters, to be handed in by groups of two students. For some assignments you need a given input signal and/or a large low-pass window with filter coefficients, the files `inputSignal.txt` and `filterCoef.txt` contain this. We assume you successfully completed the tutorial (tutorial.pdf).

deadline for handing in your solutions: 25 January 2026, 23:59, on canvas. **Deliver as group**, submission after the deadline means -5pts subtracted for each day.

Remarks on delivery

- We have provided 4 files

  FiltersXX.hs Main file for `clashi`, extend this file with your code, change the `XX` into your group number in two digits.

  FilterCoefAndInput.hs Contains filter **coefficients** in vector form and **input signal** in list form, you can import this file in your `Filters.hs` (*import FilterCoefAndInput*).

  inputSignal.txt Text file of the input signal.

  filterCoef.txt Text file of the filter coefficients.

- Add your names and student numbers at the top of the `Filters.hs` file.
- Add your names and student numbers to the front page of your report.
- In some assignments you are asked to draw a diagram, you may use any drawing tool you like. We used the free tool yEd (https://www.yworks.com/)
- For all assignments you are asked to deliver Clash code. Please include your Clash code for that specific assignment in a text box in your report.
- There is no need to deliver Verilog/VHDL code.
- Use vector functions, reference documentation of Vectors:
  `hackage.haskell.org/package/clash-prelude/docs/Clash-Sized-Vector.html`
- In some assignments you are asked to describe/show the combinational path, here you need to draw this path in a figure.

- If you are asked to **comment on clock cycles**, we are looking for comments describing the **amount of clock cycles for latency and throughput**.
- You can score 15 points in this assignment.
- Deliver your report (pdf) and the code file (`FilterXX.hs`), where `XX` is your group number in 2 digits.
- The grading also depends on the style and readability of the code.
- Keep your answers short and concise.
- For synthesis it is prefered that you use the EDAcation plugin of vscode.

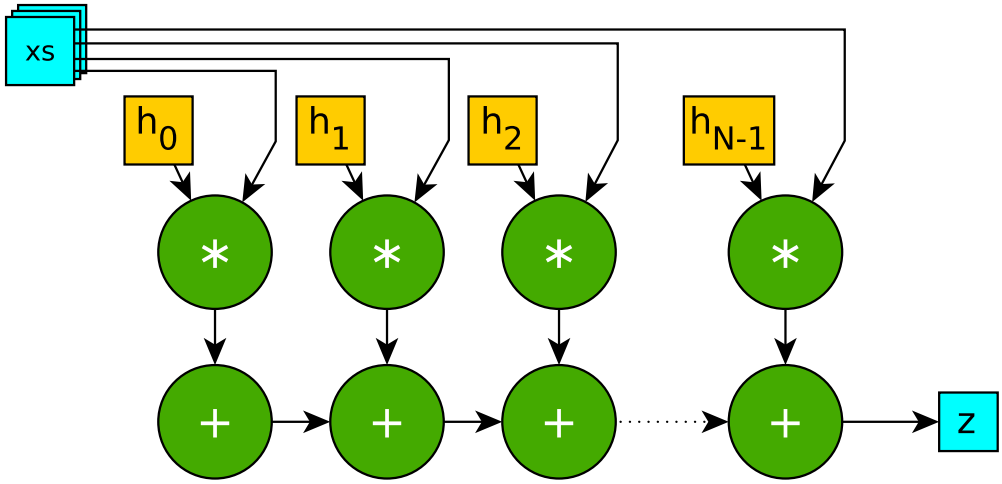

Figure 1: A layout of a FIR filter

1 Parallel FIR filter with N=6

Figure 1 depicts a structure of a FIR filter where all the inputs arrive simultaneously. The goal of the following assignment is to implement the same structured FIR filter in Clash with $N = 6$. Use `Signed 8` as number type for the values. Note that `xs` is an input vector with length `N`, and the `hs` variables are part of the state of the filter. `z` is an output (we ignore saturation). You can use any arbitrary number as filter coefficients (as long as the synthesis tooling does not optimize the multipliers away).

Assignment 1 (1 pts):

Implement this FIR in Clash, generate Verilo/VHDL, and synthesise. Provide the following in your report:

- Clash code
- Resource overview (Flow Summary in Quartus or statsview in EDAcation)
- RTL schematic
- Description/Figure of longest combinational path. (Drawing in figure is prefered)

2 A larger parallel FIR filter

To filter out a high frequency signal we use a low-pass filter, we make use of the following filter coefficients: (this is not a de-facto on how to construct filter coefficients):

$$l(n) = \begin{cases} \frac{\sin\left(2\pi f_t\left(n-\frac{M}{2}\right)\right)}{\pi\left(n-\frac{M}{2}\right)}, & \text{for } \left(n-\frac{M}{2}\right) \neq 0, \\ 2f_t, & \text{for } \left(n-\frac{M}{2}\right) = 0, \end{cases} \tag{1}$$

with the following constants:

$$\begin{aligned} \text{samplingFrequency} &= 2000, \\ \text{cutOffFrequency} &= 50, \\ \text{filterOrder} &= 100, \\ f_t &= \frac{\text{cutOffFrequency}}{\text{samplingFrequency}}, \\ M &= \text{filterOrder} - 1 \end{aligned} \tag{2}$$

The list with filter coefficients are in the file `filterCoef.txt`. Note that you can use the .hs file `FilterCoefAndInput.hs` for both the `inputSignal` and `filterCoef`. The goal of this assignment is to implement a FIR filter with the same structure as in Assignment 1, but with a different filter window. Use `SFixed 5 13` as numerical type for all the values.

Assignment 2 (0.5 pts):

Implement this FIR in Clash, generate Verilog/VHDL, and perform RTL Synthesis. Provide the following in your report:

- Clash code
- Resource overview (Flow Summary in Quartus or statsview in EDAcation)
- Parts of RTL schematic that show the extra logic surrounding the multipliers/adders
- Comment on the extra logic, why is it introduced?
- Describe the longest combinational path

(Note: The Fitter (Place & Route) will fail because there are not enough IO ports available on the device, only RTL Synthesis is required).

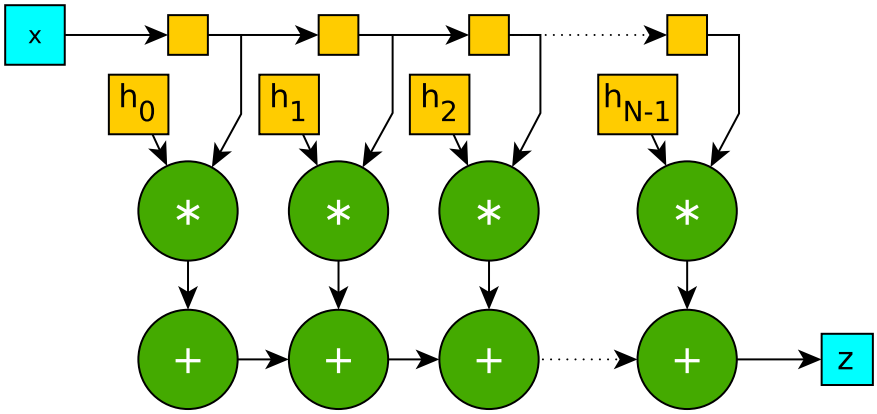

Figure 2: A transformed FIR filter

3 FIR filter with N=6

Figure 2 shows a FIR filter with registers on the input, which means that not all inputs arrive simultaneously. For the following assignment use the `Signed 8` number type for the values.

Assignment 3 (1.5 pts):

Implement this FIR in Clash with $N = 6$, generate Verilog/VHDL, synthesise. Use the predefined `hs` as coefficients. Indicate inside the RTL schematic the higher-order functions that you used. The `simulate` or `simulateN` function can simulate designs in Clash. Simulate your `mealy` machine, not your `topEntity`. Implement the simMfir2_6 that uses the simulateN function to simulate the your mealy machine with the input: inp = [1..20]. Provide the following in your report:

- Clash code
- Resource overview (Flow Summary in Quartus or statsview in EDAcation)
- RTL schematic with indication of hofs
- The result of simMfir2_6
- What are the latency and throughput in number of clock cycles?

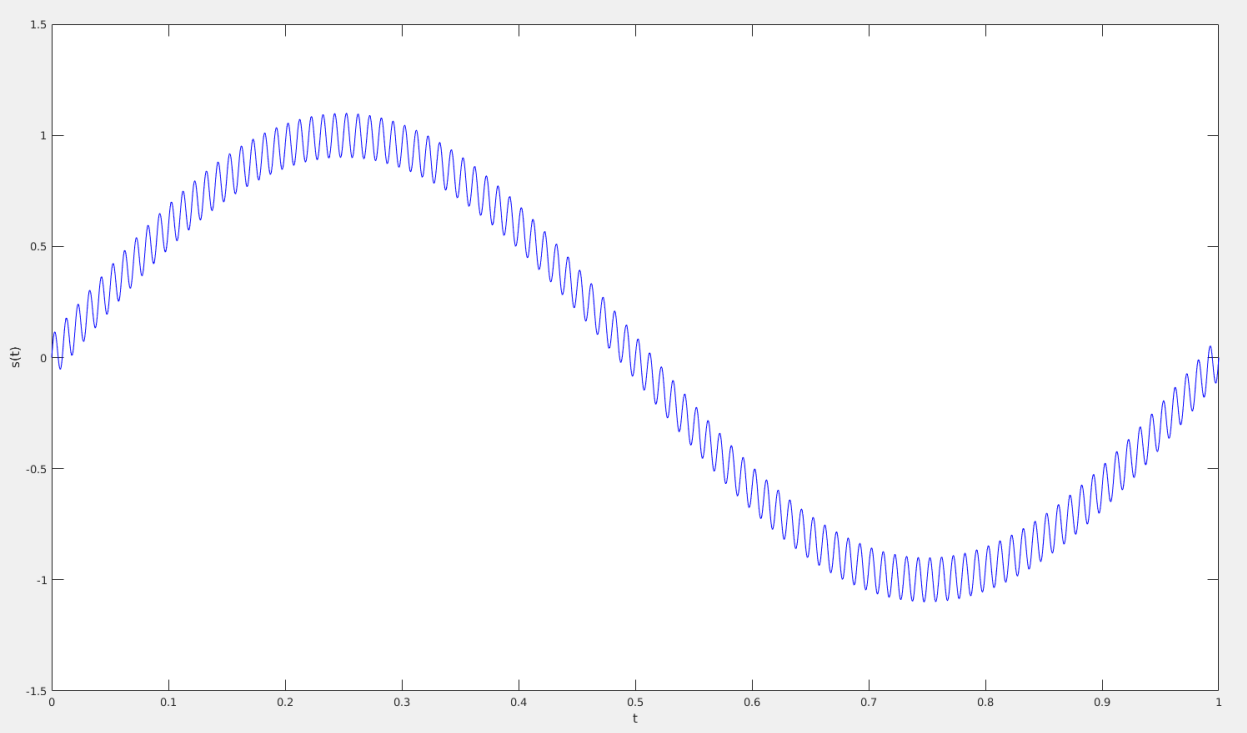

Figure 3: The input signal for Assignment 4 and 6.

4 A larger FIR filter

Figure 3 shows the input signal:

$$s(t) = a_l \sin(\omega_l t + \rho_l) + a_h \sin(\omega_h t + \rho_h) \tag{3}$$

where:

$$\begin{aligned} a_l &= 1, & a_h &= 1, \\ \omega_l &= 2\pi f_l, & \omega_h &= 2\pi f_h, \\ f_l &= 1, & f_h &= 100, \\ \rho_l &= 0, & \rho_h &= 0. \end{aligned} \tag{4}$$

The following assignment is to create a FIR filter in Clash with the same structure as described in Assignment 3. Use `SFixed 5 13` as number type for values. For this assignment use the same filter coefficients as in Assignment 2. As simulation input use the input from Figure 3. The file `inputSignal.txt` contains the input signal. You can import `FilterCoefAndInput.hs` that contains the input signal in Haskell. You may use any plotting program (for example: MATLAB or GNUPlot) to plot the output signal.

Assignment 4 (0.5 pts):

Implement this transformed FIR in Clash, generate Verilog/VHDL, synthesize. Provide the following in your report:

- Clash code
- Resource overview (Flow Summary in Quartus or statsview in EDAcation)
- Show how you used the `simulate` function
- Plot of the simulation result
- Comment on the number of registers used
- Describe the longest combinational path

5 Symmetric FIR filter with N=6

The low-pass filter used in the following assignments is symmetric (see equation (5), where x_n is the input signal, and h_k are the filter coefficients). In these assignments use the symmetry of the filter coefficients to reduce the hardware usage by a factor of 2.

$$h_0x_0 + h_1x_1 + h_2x_2 + h_2x_3 + h_1x_4 + h_0x_5 \tag{5}$$

Assignment 5 (2 pts):

Design an FIR filter with the following hardware constraints:

- 1 input (Signed 8)
- 3 multipliers (DSPs)
- Low-pass window length 6 (Signed 8 values)
- 6x8 registers

Describe the optimization algebraically. Implement this symmetric FIR filter in Clash using the `hs` as coefficients, generate Verilog/VHDL, and synthesize. Implement the `simMfir3_6` that uses the `simulateN` function to simulate the your fir with the input: `inp = [1..20]`. Provide the following in your report:

- Clash code
- Resource overview (Flow Summary in Quartus or statsview in EDAcation)
- Algebraic description of optimization
- Does resource consumption match the hardware constraints? if not, why not?
- What are the latency and throughput in clock cycles?
- The result of `simMfir3_6`
- Compare the longest combinational path with the one from Assignment 1 and comment on the difference

6 Symmetric FIR filter

In this assignment use the structure from Assignment 5, the filter coefficients from Assignment 2, and the input from Assignment 4. You can use `SFixed 5 13` as number type.

Assignment 6 (0.5 pts):

Implement the symmetric FIR in Clash, generate Verilog/VHDL, and synthesise. Provide the following in your report:

- Clash code
- Resource overview (Flow Summary in Quartus or statsview in EDAcation)
- Show how you used the simulate function
- Plot of simulation result
- What are the latency and throughput in clock cycles?

7 Transformed symmetric FIR filter with N=6

For this assignment the filter from Assignment 5 is used. In practice the following transformation steps keep the functionality of the filter the same.

- Reverse the direction of all lines
- Interchange input and output
- Replace junction points by adders and adders by junction points

Assignment 7 (3 pts):

Perform the transformation as described above. Implement the transformed FIR in Clash, generate Verilog/VHDL, and synthesise. Implement the `simMfir3t_6` that uses the `simulateN` function to simulate the your fir with the input: `inp = [1..20]`. Provide the following in your report:

- Clash code
- Diagram of transformation
- Resource overview (Flow Summary in Quartus or statsview in EDAcation)
- RTL schematic
- The result of `simMfir3_6`
- What are the latency and throughput in clock cycles?
- Draw the longest combinational path in your diagram

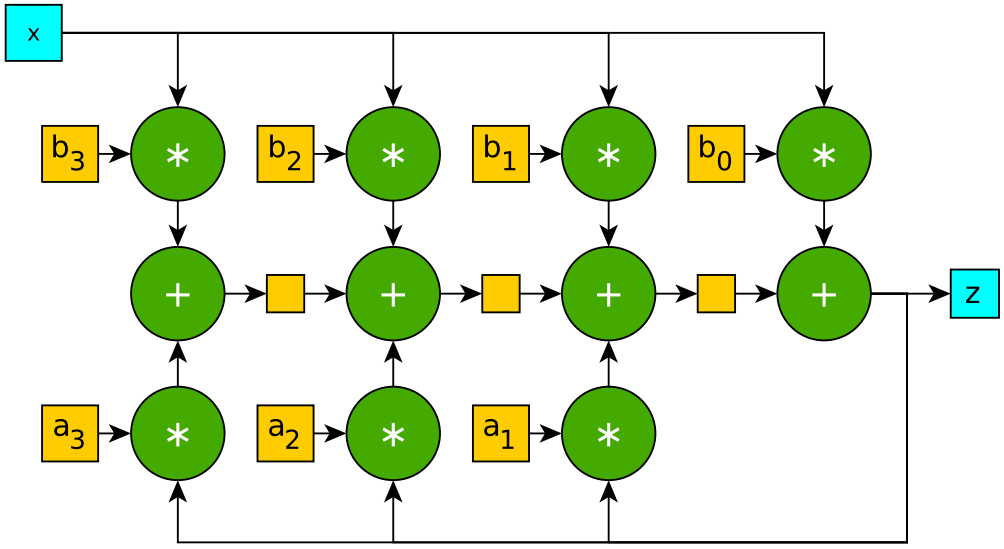

Figure 4: IIR filter structure

8 IIR filter

Figure 4 shows a structure of an FIR filter. In the following assignment implement such a filter with the following coefficients (`SFixed 5 13`):

$$bs = [b_0 = 0.0623348, b_1 = 0.1870044, b_2 = 0.1870044, b_3 = 0.0623348] \quad (6)$$

$$as = [a_1 = 0.9853304, a_2 = -0.5929545, a_3 = 0.1089457] \quad (7)$$

Assignment 8 (3 pts):

Implement the IIR in Clash, generate Verilog/VHDL, and synthesise. Implement the `simMiir1` that uses the `simulateN` function to simulate the your fir with the input: `inp = (1:L.replicate 49 0)`. Provide the following in your report:

- Clash code
- Resource overview (Flow Summary in Quartus or statsview in EDAcation)
- RTL schematic with indication of higher-order functions from your Clash code
- Draw the longest combinational path in the diagram
- What are the latency and throughput in clock cycles?
- The result of `simMiir1`, and give a plot of the result

9 Transformed IIR filter

For this assignment the same transformation technique is used as in 7.

Assignment 9 (3 pts):

Perform the transformation as described in 7. Implement the transformed IIR in Clash, generate Verilog/VHDL, and synthesise. Implement the `simMiir2` that uses the `simulateN` function to simulate the your fir with the input: `inp = (1:L.replicate 49 0)`. Provide the following in your report:

- Clash code
- Resource overview (Flow Summary in Quartus or statsview in EDAcation)
- RTL schematic
- Draw the longest combinational path in the diagram
- What are the latency and throughput in clock cycles?
- The result of `simMiir2`, and give a plot of the result