CS-472: Design Technologies for Integrated Systems

Homework Assignment 2 Due date: 27/10/2022, 23:59

Problem Description

FIR filters are one of the most common digital filters used. You have to write the code that implements a 10 stage FIR filter in VHDL subject to timing and area constraints. For more information on the FIR filter you can have a look at: `http://en.wikipedia.org/wiki/Finite_impulse_response`

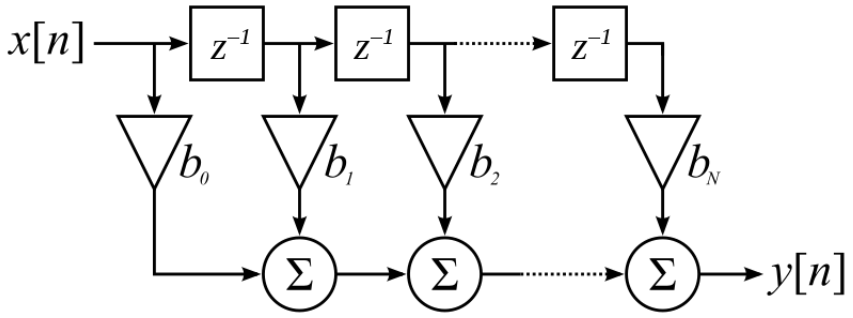

Figure 1: FIR filter

Inputs: 10 – 32bit numbers
Output : 1 – 32bit number
Function: given in Figure 1 (only 10 stages, the delay register is given)

$$y[n] = b_0x[n] + b_1x[n-1] + \cdots + b_Nx[n-N] = \sum_{i=0}^{N} b_i \cdot x[n-i] \quad (1)$$

Constants:
b0=b9=19
b1=b8=390
b2=b7=864
b3=b6=3072
b4=b5=9
Carry and overflows for additions and multiplications are ignored.

Support

Support files are provided in the zip file on Moodle, containing:

1. Basic implementation of the filter (*fir_base*)

2. Synthesis (*Design Compiler*) scripts.

3. Skeleton for your optimized implementation (VHDL components, testbench etc.)

You can login on *selsrv1* with your given username. Check out "Server to be used for programming assignments" on Moodle for your username (and upload the statement if you are required to do so!). Your default password is "**22.eda?user**"[1] (without the quotes). Please change your password on your first login. You can do so by typing `yppasswd` on the terminal.

Task

Goal: Optimize the FIR filter to meet the given constraints:

- Technology: 22nm CMOS.
- Timing: clock cycle 1.75 ns
- Throughput: 1 result per cycle (the delay register samples every clock cycle)
- Total area: maximum 1800 μm^2
- Synthesis tool: Synopsys DC
- Libraries: same as in the tutorial below. Launch dc_shell in the provided folder to get the right libraries. For adder and multipliers, use the provided components libraries.

You are NOT allowed to:

1. Flatten the hierarchy!
2. Change the `edadk.conf`.
3. Change the DC script (e.g., by using `compile_ultra`).
4. Use your own adder and multiplier implementation.
5. Change any files except for the architecture of the file `fir_sol.vhd` (do not touch the entity).

Complete the following tutorial to get to know how to use *Modelsim* and *Synopsys Design Compiler*. Also try to read the code in *SRC/fir_base/fir_base.vhd* to understand the behavioral requirement that you need to complete. Then, implement your optimized version in *SRC/fir_sol/fir_sol.vhd*. Redo the simulation steps (selecting *fir_sol* instead of *fir_base*, of course) to check the correctness of your implementation, and use *Design Compiler* to synthesize your design. Make sure the slack is met and the clock cycle (`report_timing`) and total area (`report_area`) meet the constraints given above.

[1] You should have changed it already

We have noticed in some rare case that even if the design is completely optimized the slack is not met (by a very small amount). If you are in this situation, confident that you did everything possible to optimize the design, try to change the VHDL code by swapping the position of the critical signal (reported by `report_timing`) with another one using the commutative property. For instance rewriting $a+b$ with a critical as $b+a$.

Rules for Submission

Please compress the entire SRC folder and submit the zip file only. If you fail to fulfill the above constraints and still hope to get partial points, write a short report to describe your circuit design (e.g., data-flow diagram, what optimization techniques you have used, etc).

Tutorial

First, you need a computer with Linux system or with SSH support. To use applications with graphic interfaces, you also need to have an X11 server installed on your computer (e.g., XQuartz for MacOS, or Xming for Windows). Log in to server: *selsrv1.epfl.ch* with your edauser*X* account, where *X* is the index of your account (see table on Moodle). The command to establish a connection to the server is:

```
ssh -Y edauserX@selsrv1.epfl.ch
```

Steps in the terminal of selsrv1/2:

1. Open firefox web browser by command: `firefox &` and download the provided files from Moodle.
2. Extract files by command: `unzip <zip_file_name>`.
3. Read the README file inside the extracted file directory.
4. Go to folder "*MSIM*" and launch *Modelsim* by command: `vsim &`.
5. Steps for *Modelsim*:
   (i) Create a new project by going to menu bar: *File* $\rightarrow$ *New* $\rightarrow$ *Project*.
   (ii) Give the project name and click *OK*.
   (iii) Add existing files to the project (Do not copy them to the project! Choose to link them to the project):

   ```
   SRC/components/array_t.vhd
   SRC/components/adder.vhd
   SRC/components/mult.vhd
   SRC/components/reg.vhd
   SRC/fir_base/fir_base.vhd
   SRC/fir_base/fir_base_wrapper.vhd
   SRC/fir_base/tb_fir_base.vhd
   ```

(iv) You can always right click the project window and select *Add to Project* → *Existing files* to include additional VHDL files.

(v) Right click in the project window, and select *Compile* → *Compile All* button to compile all the included VHDL files (Compiler will report detected errors in the transcript window. It could be the case that more than one compilation is needed to get the correct order). Alternatively, you can select *Compile* → *Compile Order* to open the Compile Order dialog box, and then click on *Auto Generate* followed by *OK*.

(vi) Right click in the project window and select *Add to Project* → *Simulation Configuration* to create a simulation profile.

(vii) In the pop-up window, give a name to the simulation profile.

(viii) Select the `tb_fir_base` in the library *work* as the unit to be simulated.

(ix) Click *Save* to finish simulation creation.

(x) Double click the simulation profile and simulation will be automatically initialized.

(xi) In the object window, select all the signals and *right click*: Select *add wave* and in the toolbar of the pop-up window, you specify the length of a simulation period. We recommend you to use 200 ns.

(xii) Click button *run* once in the toolbar. You should be able to see the results.

(xiii) Select all the signals, *right click* and select *Radix* → *Decimal*. This changes the display of signal values from binary numbers to decimal numbers, which is easier for you to identify.

(xiv) Close *Modelsim*.

6. Go to folder "*SNPS_DC*" and launch *Design Compiler* by command: `dc_shell -f ../SCRIPTS/fir_base.tcl .` Area and timing results will be reported at the end of synthesis.

7. (Optional) To see more detailed results from DC:

   (i) Go to directory "*SCRIPTS*" and open `fir_base.tcl` with a command-line text editor such as *nano*, *vim* or *emacs*.

   (ii) Comment out the last line: `#exit` to prevent *Design Compiler* to quit after executing all the commands in the script.

   (iii) Redo the `dc_shell` command in step 6.

   (iv) After synthesis is finished, type: `start_gui` to launch the graphic user interface.

   (v) You can use commands `report_area` and `report_timing` to see area and timing results.

   (vi) Right click in the window "Logical Hierarchy" on the design name: `fir_base_wrapper`.

   (vii) Select *Schematic View*. Design compiler will display the schematic of synthesized design.

   (viii) In the menu bar, *Timing* → *Path Slack*. Then click *OK*.

   (ix) `report_area -hier` to see multipliers and adders area.