**Integrated Systems Architectures**

# Lab 2: digital arithmetic
# Assignement

Read the Lab 2 description and address the following points.

## 1 Digital arithmetic and logic synthesys

- Download from *opencores.org* or download from the portale the "Floating Point Adder and Multiplier" (fpuvhdl) project.
- Unpack the project.
- Take the pipeline multiplier, (and the common) files. The multiplier architecture is very close to the one shown on p.84 of the slides "Part 2-B Arithmetic circuits", but with pipeline.
- Verify the model by creating a testbench to perform some floating point multiplications(1).
- Modify the VHDL to add registers to the inputs and verify it still behaves correctly.
- Build a table summarizing the results asked in the following points:
  1. Force Design Compiler to flatten the hierarchy and synthesize. Find the maximum frequency and the area.
  2. Force Design Compiler to flatten the hierarchy and to implement the Significands multiplier in Stage2 as a CSA multiplier. Find the maximum frequency and the area.
  3. Force Design Compiler to flatten the hierarchy and to implement the Significands multiplier in Stage2 as a PPARCH mulitplier. Find the maximum frequency and the area.

**Note:** the maximum frequency corresponds to the minimum period with slack equal to zero.

(1) you can use the data in file "fp_samples.hex" and compute the square values, which results are stored in file "fp_prod.hex".

### 1.1 Fine-grain Pipelining and optimization

- Build a table summarizing the results asked in the following points:

  1. Add one register at the output of the Significands multiplier and add all the required registers such that the timing of the whole Stage2 is correct. Then, try to issue the `optimize_registers` command after `compile`. Find the maximum frequency and the area.
  2. Try to issue the `compile_ultra` command instead of `optimize_registers+compile`.

- Design an MBE multiplier for unsigned data. Partial products must be generated with no adders/subtracters[(2)]. The adder plane must rely on a Dadda-tree[(3)]. Sign extension bits in the Dadda-tree must be simplified in order to reduce the number or half adders and full adders[(4)]. Then, use the MBE multiplier, instead of the behavioural operator "*" in the Significands multiplier in Stage2 of the Floating point multiplier. Verify via simulation that the model is still working correctly. Then, find the maximum frequency and the area.

- Build a table summarizing all the results you obtained in this lab and comparing the different solutions.

(2) see the file "modified-booth.pdf".

(3) see the file "wallace_dadda.pdf".

(4) see the file "sign_extension_ booth_multiplier_Stanford.pdf"