Integrated Systems Architectures

Lab 2: digital arithmetic
General description

The aim of this lab is to deal with digital arithmetic issues.

1 Digital arithmetic and logic synthesizers

1.1 Introduction and background

As you could appreciate during the first lab, modern logic synthesizers (such as Synopsys Design Compiler) handle the behavioural description of adders and multipliers by directly inferring the component in the netlist. For Design Compiler this is possible due to the availability of Design Ware (DW), which is basically a collection of ready-to-use blocks. In particular, the `report_resources` command shows the arithmetic resources employed in your design and the corresponding architecture. As detailed in the documentation, Design Ware contains among the others a parametric multiplier (*DW02_mult*), which can be implemented as carry-save (csa) or parallel-prefix (pparch). You can force Design Compiler to use a specific architecture for each element (adder, multiplier, ...) in your design by using the `set_implementation` command. After specifying the clock constraints, before issuing the `compile` command you can specify the implementation of each cell.

1.1.1 Dealing with hierarchy

If your design has some hierarchical organization you have to discover the hierarchy to customize the `set_implementation` command. To ease this task you can first flatten the hierarchy and then force Design Compiler to use the DW component you want.

example Flatten the hierarchy and specify the architecture (e.g. csa) for all the multipliers:

```
ungroup -all -flatten
set_implementation DW02_mult/csa [find cell *mult*]
```

Then, you can issue the `compile` command and check the result with `report_resources`.

1.2 Pipelining

Instead of writing by hand a pipelined multiplier or adder you can exploit Design Compiler capability of optimizing register position (retiming). A simple example is the design of a pipelined multiplier. You can describe the multiplier with the behavioural operator "*" and place a chain of registers at the output. Then, you can synthesize the design with the `compile` command and then force Design Compiler to re-compile the design performing retiming by using the `optimize_registers` command.

1.3 Optimization

Several optimization tools are automatically enabled by using the *ultra* mode in Design Compiler. The *ultra* mode is automatically enabled by issueing the `compile_ultra` command instead of `compile`.