Hardware description languages: Gezel, VHDL, Verilog, SystemC

Tutorial 03 on Dedicated systems

Teacher: Giuseppe Scollo

University of Catania
Department of Mathematics and Computer Science
Graduate Course in Computer Science, 2017-18

Table of Contents

tutorial outline

this tutorial deals with:

- Gezel basic concepts
- getting started with synchronous hardware description in VHDL
- a glimpse at Verilog and SystemC
- lab experience

Gezel: basic concepts

essentially, two sorts of variables: signals and registers

- one or more bit-wide, of type **ns** or **tc** ecc.
- registers are initialized to zero, signals have no memory
- datapath inputs and outputs are always signals
- a register output gets the register input value at the clock active edge, therefore the (discrete time) semantics of register assignment is different from the (instantaneous) semantics of signal assignment

assigment order is irrelevant

> hardware is inherently parallel

assignment right-hand-side expressions are constructed by operators that have a direct interpretation as hardware components, see Schaumont Table 5.1 and Sect. 5.1.3

datapath encapsulation and structural hierarchy, with module reuse and cloning, see Schaumont Sect. 5.2

getting started with VHDL

abstraction level: discrete events; semantics: queue of future events

specification units:

- *entity*: interface specification (ports: names, types and directions of I/O signals)
  may have parameters, by *generic* declarations
- *architecture*: specification of the entity internals

more than one architecture may be associated to a given entity, following different specification styles:

- *functional*: in terms of sequential *processes*, activated by change of given signals (sensitivity list)
- *behavioral*: in terms of *concurrent* assignments (signal relations)
- *structural*: in terms of *components* and port connections between *instances* thereof

encapsulation and hierarchy:

- components: entity-like encapsulation, structural hierarchy
- processes: encapsulate functionality, yet process nesting is not allowed

see Zwolinski ch. 3 for a primer on syntax of VHDL constructs and simple examples

a glimpse at Verilog and SystemC

abstraction level: discrete events (same as VHDL)

Verilog, a few features:

- similar to VHDL for description of schematics as connected entities
- more features for transistor-level description
- less flexible for top-down system specification
  SystemVerilog designed to this purpose, but less supported by automated synthesis tools
- see Zwolinski, App. B for a primer on Verilog syntax and simple examples

SystemC, a few features:

- C++ class library with functions required for hardware description
- concurrent processes controlled by sensitivity list (cf. VHDL)
- layered language architecture, with channels for communication, for models of computation, and methology-specific channels
- see Marwedel sect. 2.7 for a more informative introduction to SystemC

lab experience

the VHDL translation produced by the Gezel code generator sometimes yields an unexpected outcome...

- create the source file delay_collatz.fdl containing the second version of the Gezel example shown in the second lecture (the file is available in the lab reserved area, in the 3x+1 folder)
- run from the command line: fdlvhd delay_collatz.fdl
- launch Quartus and in this system create a new project named delay_collatz
- copy the .vhd files produced by step 2 into the project directory and open file delay_collatz.vhd in the Quartus editor
- assign the aforementioned files to the project, compile and visualize the resulting error message, then, by double-click on the error message in the compilation report, find out the VHDL source line causing the error
- analyse the delay_collatz.vhd source and find a suitable modification that would fix the error
- set the clock to a frequency that warrants a positive value for the worst-case slack (see clock fine tuning tips from the first lab lab experience)
- compare usage of resources (n. of LEs and registers) and worst-case slack with those obtained from compilation and timing analysis completed in the second lab experience
- create test waveforms for the collatz circuit, with clock input corresponding to the frequency established in the previous step, initial value 27 for the first trajectory and, as a follow-up in the same waveform, another initial value for a second trajectory after 75 clock cycles
- run the functional simulation and check the correspondence of the outcome to expectations
- find out whether and how could one modify the Gezel source delay_collatz.fdl in such a way that its translation to VHDL would not break the compilation

references

recommended readings:

Schaumont, Ch. 5, Sect. 5.1-5.2

Zwolinski, Ch. 3, Sect. 3.1-3-7

other sources for consultation:

Brandolese, Fornaciari, App. B

Zwolinski, App. B

Marwedel, Ch. 2, Sect. 2.7

Vahid, Digital Design, Ch. 9, Hardware Description Languages (PDF slides)

Smith, VHDL & Verilog Compared & Contrasted