Introduction to the combined use of Gezel with a VHDL simulator

Tutorial 01 on Dedicated systems

Teacher: Giuseppe Scollo

University of Catania
Department of Mathematics and Computer Science
Graduate Course in Computer Science, 2016-17

Table of Contents

tutorial outline

this tutorial deals with:

- hardware models in Gezel: features, limitations, practical uses
- Gezel software installation
- translation of Gezel models to VHDL models
- Quartus software installation
- compilation, analysis and tuning of VHDL models in Quartus
- editing of testing waveforms in Quartus
- functional simulation in Quartus' ModelSim

hardware models in Gezel

single-clock synchronous digital circuits, composed of an interconnection of:

- combinational logic
- flip-flops

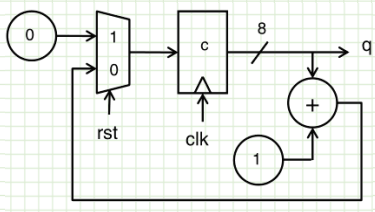
thus also including widely used components such as: registers, adders, multiplexers etc.

abstraction level: clock cycles, RTL models

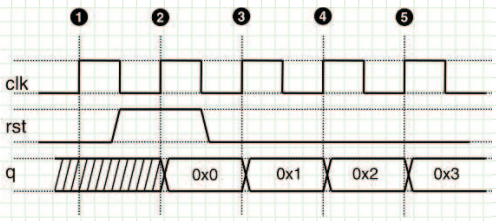
what *cannot* be modeled: asynchronous hardware, HW with latches, multi-phase clocked HW etc.

RTL models are adequate in a great deal of practical cases, viz. to describe hardware implementations of algorithms

example:

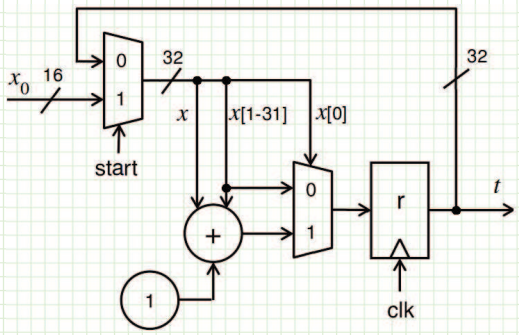



```
dp count(in  rst : ns(1);
         out q   : ns(8)) {
  reg c : ns(8);
  always {
    q = c;
    c = rst ? 0 : c + 1;
  }
}
```



Schaumont, Fig. 1.1 - Models and behaviour of a hardware component

reconsider the example seen in the first lecture:




```
dp collatz ( in start : ns(1) ; in x0 : ns(16) ;
             out t ns(32)) {
  reg r : ns(32) ;
  sig x : ns(32) ;
  always {
      t = r ;
      x = start ? x0 : r ;
      r = x[0] ? x + (x >> 1) + 1 : x >> 1 ;
  }
}
```

what can be done with such a model?

e.g., how can one visualize its behaviour?

industrial development tools need descriptions in standard languages such as VHDL or Verilog ...

the code generator of the Gezel platform yields a translation into *synthesizable* VHDL

lab experience:

1. install Gezel
2. create the source file collatz.fdl containing the Gezel example description
3. run from the command line: fdlvhd collatz.fdl
4. install Quartus 13.1 (Web edition) by Altera, launch it, and then in this system:
5. create a new Quartus project named collatz
6. copy the .vhd files produced by step 3 into the project directory
7. assign the aforementioned files to the project and compile
8. check any error or warning messages
9. set the clock to a frequency that warrants a positive value for the worst-case slack
10. create test waveforms for the collatz circuit, with clock input corresponding to the frequency established in the previous step and value 27 for the trajectory start
11. run the functional simulation
12. repeat the simulation for different trajectory starts

a few tips to perform the lab experience on Ubuntu 16.04:

the following notes present a few workarounds to little troubles which otherwise may affect the execution of the lab experience

- *you may download the ZIP archive of all tips*

1. Gezel installation tips
2. Quartus 13.1 (Web Edition) installation and startup tips on Ubuntu 16.04
3. Quartus project assignment tips
4. clock fine tuning tips using Quartus TimeQuest Analysis
5. tips on using Quartus ModelSim

recommended readings:

Schaumont (2012) Cap. 1, Sez. 1.1.1; App. A.1
Quartus II Introduction Using VHDL Designs - For Quartus II 13.1; Altera University Program
Using TimeQuest Timing Analyzer - For Quartus II 13.1, Sect. 1-2, 4; Altera University Program

for further consultation:

Schaumont (2012) App. A.2

other useful material for the proposed lab experience:

Gezel installation manual
Quartus II 13.1 Web Edition download
Altera University Program Installer