ingo.korb [☺] tu-dortmund.de

Exercises to
Cyber-physical System Fundamentals
Summer term 2016

Assignment 3

(10 Points)

To be solved in the weeks starting on Monday, June 6, 2016

Hints:

- For this session we use GHDL and GtkWave under Linux, using the virtual machine **CPSF-VHDL**. User name and password are the same as you have used for logging into Windows before. Unpack the template files from the file server to a convenient folder in your home directory. The templates contain a file `README.txt` with additional informatin.
  Type “make” to compile your source files, type “make view” to run the waveform viewer. Use the source code template “a3.1_template” for an initial test. It is readily compilable.
- Lecture slides 2.06/40ff provide additional material in VHDL, which is also placed in the assignment folder.

3.1 Semantics of VHDL-simulations (4 Points)

VHDL features two different timing models called *inertial* (default) and *transport*. The former model supports the simulation of inherent lag in circuitries. Consider the following source code. Understand its semantics by simulation. What happens when lines 25 and 26 are changed from `...<= ...` to `...<= transport ...`? What is the meaning of the two timing models in this example and how does it manifest itself?

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use work.all;
4
5  entity vhdl_entity is
6  end vhdl_entity;
7
8  architecture rtl of vhdl_entity is
9    signal u, v, w : std_logic;
10 begin
11
12   p1: process
13   begin
14     u <= '0',
15          '1' after 10 ns,
16          '0' after 15 ns,
17          '1' after 20 ns,
18          '0' after 25 ns,
19          '1' after 30 ns;
20     wait;
21   end process p1;
22
23   p2: process (u)
24   begin
25     v <= u after 4 ns;
26     w <= u after 6 ns;
27   end process p2;
28
29 end rtl;
```

3.2 VHDL Syntax (6 Points)

In the lecture, a full-adder written in VHDL has been sketched . Extend this initial example to form an N-bit ripple carry adder (RCA). Such an adder passes the carry-bit bit from one “stage” to the next, hence the term *ripple carry*.

Implement a 4-bit RCA like shown, based on the source code template “a3.2_template”. The template already provides basic structural elements, a driver component for the simulation, some example code for reference and comments (see also the slides for reference). Compile and test your solution. Please note that in the template both the full adder (`full.vhdl`) and the 4-bit ripple carry adder (`rca.vhdl`) must be implemented by you.

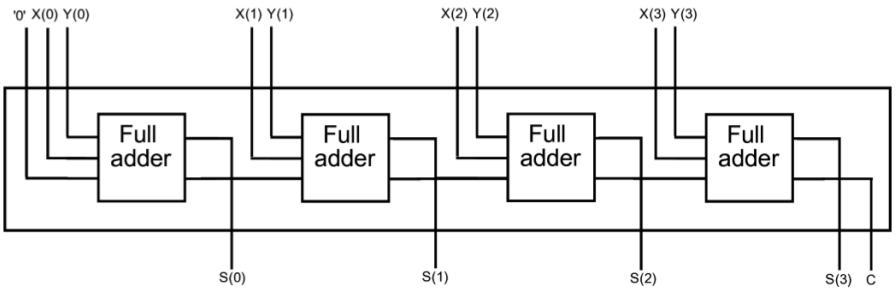

Figure 1: Ripple Carry Adder

General notes:

Dates and additional information can be found on the lecture website. The assignments will be typcically be published **Tuesdays** on a weekly basis and have to be solved in the lab session of the following week. To pass the labs, a minimum of 50% of the total points must be achieved in the first half and the second half, respectively.