Beyond classical chip design
Exercise VIII

summer term 2015

Matthias Függer

Ex 24. (4P) A widely used implementation of the 2C-Element is the "van Berkel" implementation. See, C.H. van Berkel: "Beware the isochronic fork", 1991 (Figure 4.5).

- Discuss how the implementation works, and whether it drives strong or weak 0 and 1s in presence of strong inputs. Does it have large static power consumption (e.g., by connecting GND to Vdd like in the pseudo nMOS gates)? What is a timing constraint that has to hold on the arriving inputs for the implementation to work correctly? (2P)

- Compare it with the last exercise's (i) weak inverter-feedback implementation, and (ii) your combinational implementation in terms of static power, and latency. For latency consider consider two scenarios (S1) and (S2): In both scenarios assume that the C-Element is initialized to 0. (S1) Input $A = 1$ and $B = 0$ and for a long time when a $0 - 1$ transition occurs at B. (S2) Input $A = 0$ and $B = 1$ and for a long time when a $0 - 1$ transition occurs at A. Are delays for the output to make a $0 - 1$ transition equal in both cases for both (i) and (ii)? Why or why not? (2P)

Ex 25. (1P) In the lecture we needed a completion detection circuit (CD) for the dynamic asynchronous pipeline PS0. Give a gate-level implementation of it (you may use C-Elements and standard gates) in case of $n \geq 1$ bits that are dual-rail encoded (see lecture 5, p.17).

Ex 26.* (2P) Consider Sutherland's micropipeline with the capture-pass latches and C-Elements as presented in the lecture. This is called a "linear" pipeline as data is linearly feed through the pipeline. Pipelines, however, need not be linear, but can have forks and joins (explained below).

Imagine that you would like to fork data at the end of a linear pipeline A and process it further, along two different pipelines B and C. Afterwards, the data of both pipelines B and C is joined again and fed into the beginning of A. Fork and Join are defined as follows:

- Fork: Let x be the data at the end of the linear pipeline A. The fork duplicates x and feeds it into pipelines B and C.

- Join: Let x and y be the data at the end of pipelines B and C. The join merges both by applying a function f and feeding the result $z = f(x, y)$ into pipeline A.

Forks and Joins have to handle req and ack properly. How would you implement such a fork (1P) and join (1P) at gate level (you may use C-Elements and standard gates)? Draw the join and fork circuits and also a small complete circuit (pipelines A, B, C can be only 1 or 2 stages deep).