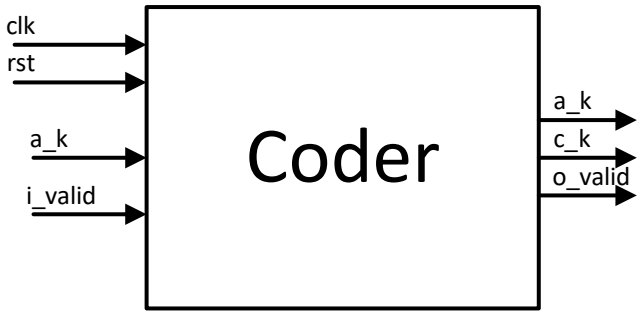
Convolutional Codes Generator

A convolutional code is a type of error-correcting code that generates parity symbols via the sliding application of a boolean polynomial function to a data stream. The sliding application represents the *convolution* of the encoder over the data, which gives rise to the name.

It is required to design a digital circuit for implementing a convolutional codes generator with the following requirements:

1. Code rate $R_c = {}^{1}/_{2}$
2. Constraint length $K = 11$
3. Encoding function: $c_k = c_{k-8} + c_{k-10} + a_k + a_{k-3} + a_{k-4}$

where, a_k is the input data stream, c_k is the generated code, R_c is the ratio between the input data rate and the encoded output rate (in this case, for each input a_k an output (a_k, c_k) is generated), and K indicates the memory of the encoding function (it depends on the previous $K-1$ inputs). The interface of the circuit to be designed is as follows:



Input data is evaluated only when the *i_valid* signal is asserted, otherwise the circuit state is retained. New data on output data must remain valid (*o_valid* = 1) only for one clock cycle, if no data is available (because there are no valid inputs), the output must be invalid (*o_valid* = 0).

You are requested to deal with the various possible error situations, documenting the choices made. In particular, it is necessary to take into consideration:

- Valid input signal changes

The final project report must contain:

- Introduction (circuit description, possible applications, possible architectures, etc.)
- Description of the architecture designed (block diagram, inputs/outputs, etc.)
- VHDL code (with detailed comments) to be attached to the report.
- Test strategy (Test-plan) and related Testbench for verification; a detailed, though not exhaustive, verification is required, including error situations and borderline cases of functioning
- Interpretation of the results obtained in the automatic synthesis/implementation on a Xilinx FPGA platform in terms of maximum clock frequency (critical path), elements used (slice, LUT, etc.) and estimated power consumption. Comment on any warning messages.
- Conclusions