HDL Coding Style and Conventions

This document describes the conventions used by the **cloudFPGA Development Kit (cFDK)** to name HLD signals, HDL processes and HDL variables.

Preliminary

The following naming rules aim at improving the readability of the HDL code and at facilitating the understanding of the various components instantiated by the *cFDK*, while restricting the documentation to a minimum.

For example, one of the general guidelines is to always explicitly indicate the source and/or the destination of a signal into the name of the signal itself (cf.Figure-1).
As you will read, some of these rules may preclude the generic re-use of a process or a variable because their names are too specific or too much spelled out (.e.g a signal name may carry both the name of its source and its destination).
This drawback is acknowledged and is accepted since the high-level components of the *cFDK* are mostly unique and are barely replicated. Instead, these high-level components come with large number of I/O ports and interfaces which are easier to understand and to interconnect if they use explicit and/or descriptive names.
Notice however that these rules are intended to vanish and to be replaced by more traditional HDL naming conventions as the design becomes more generic during the top-down specification. This transition typically occurs when instantiating specific IP cores or generic template blocks such as FIFOs or RAMs.

General Naming Conventions

The following general rules apply to the **identifiers** used to name objects when coding in VHDL or Verilog. Object items which can be named are: constants, variables, types, entities, architectures, modules, instances, configuration, signals, ports, processes, procedures, functions, libraries and packages.

The cFDK applies the following naming conventions to name its identifiers:

- All identifiers use mixed casing (i.e. ***CamelCase*** style) except **instances** such as components, entities and IP blocks which use only upper-case letters .
- If applicable, an identifier starts with a lower-cased **prefix** that indicates the type of the named object (e.g. '***s***' for '*signal*' or '***pi***" for '*Port-In*').
- If applicable, an identifier must be followed by a **suffix** indicating special properties of the named object (e.g. '***_n***' for '*active-low*' or "*En*" for Enable).

Instance Names

A **cFDK instance** is a block such as a VHDL entity, a VHDL component or a Verilog module.

- An instance name is an ***abbreviation*** formed with initial or significant letters of the instantiated component or module (*e.g.* * '***TOE***' for '*TcpOffloadEngine*' or * '***ETH***' for '*Ethernet*').
- An instance name is ***3-5 characters*** long (*e.g.* * '***SHL***' or '***SHELL***').
- An instance name is always written in **UPPER-CASE**.

Port Names

A **cFDK port** is a primary communication channel between an **instance** and its environment. As such, a port always declared as a single- or multiple-bits '**signal**' operating in input, bidirectional or output mode.

- A port name uses is a combination of strings consisting of a mandatory '**pi|pio|po**' prefix, an optional **instance** name, an optional **interface**, an optional list of **sub-interface(s)** and a **suffix**:
  pi|*pio*|*po*[*INST*]_[*Itf*]_[*SubItf*]_**PortName**_[*Suffix*]
  with
  - *pi*|*pio*|*po* = a prefix indicating the input, bidirectional or output direction of the port.
    - E.g. : pi**Reset**, pio**Data**, po**Led**.
  - [*INST*] = a string indicating the name of the instance that sources or sinks the port. This instance name is always in UPPER-CASE and is followed by an underscore. The idea is to minimize the amount of guesswork required from the user when attaching signals to ports.

        - E.g. : piSHELL_**Reset**, pioMMIO_**Data**, poTOP_**Led**.
    - [*Itf*] and [*SubItf*] = a string indicating the name of the interface and/or sub-interface(s) that the primary port element belongs to. Such an interface name is always in *CamelCase* followed or separated by underscore(s). The aim of the interface and sub-interface(s) names is to group a set of ports under a common interface and/or sub-interface name, and is to be thought here as a replacement for the VHDL record construct which is not supported by many synthesis tools.
        - E.g. : piSHELL_Mem_**Reset**, pioMMIO_Emif_XMem_**Data**, poTOP_Status_**Led**.
    - [Suffix_] = a string indicating one or multiple properties of the port.
        - E.g. : piSHELL_Mem_**Reset**_n, poTOP_Status_**Led**_a.

Streamed Port Names

The cFDK makes heavy use of data streaming interfaces and provides a dedicated naming convention for those streams. A streamed port name is a short name for a set of ports that are grouped under the same streaming interface or a streaming sub-interface.

- A streamed port follows the same naming conventions as the port names except for the mandatory prefix with becomes '**si|so**' :
  si|so[*INST*]_[*Itf*]_[*SubItf*]_**PortName**_[*Suffix*]
  with
    - *si|so* = a prefix indicating the input, bidirectional or output direction of the stream. Here is an examples of 5 ports being part of the same stream:
        - siSHL_Tcp_**Data**_tdata
        - siSHL_Tcp_**Data**_tkeep
        - siSHL_Tcp_**Data**_tlast
        - siSHL_Tcp_**Data**_tvalid
        - siSHL_Tcp_**Data**_tready,

Signal Names

Signals are the primary objects describing a hardware system and are equivalent to "wires". They interconnect concurrent statements within an instance as well as communication channels among instances.

- A signal name uses is a combination of strings consisting of a mandatory ‘**s**’ prefix, an optional source **instance** name, an optional destination **interface**, an optional list of **sub-interface(s)** and a **suffix**:
  s[*FROM*]_[*TO*]_[*Itf*]_[*SubItf*]_**PortName**_[*Suffix*]
  with
  - *s* = a prefix indicating the ‘*signal*’ nature of this object.
    - E.g. : s**Reset**, s**Data**, s**WrEn**.
  - [*FROM*] = a string indicating the name of the block or instance that sources the signal. This name is always in UPPER-CASE and is followed by an underscore. The idea is to minimize the amount of guesswork required from the user when attaching signals to blocks and instances.
    - E.g. : sSHELL_**Reset**, sROLE_**Data**, sMMIO_**WrEn**.
  - [*TO*] = a string indicating the name of the instance that sinks the signal. This name is always in UPPER-CASE and is followed by an underscore. The idea is to minimize the amount of guesswork required from the user when attaching signals to ports.
    - E.g. : sSHELL_ROLE_**Reset**, sROLE_SHELL_**Data**, sMMIO_NTS0_**WrEn**.
  - [*Itf*] and [*SubItf*] = a string indicating the name of the interface and/or sub-interface(s) that the signal belongs to. Such an interface name is always in *CamelCase* followed or separated by underscore(s). The aim of the interface and sub-interface(s) names is to group a set of ports under a common interface and/or sub-interface name, and is to be thought here as a replacement for the VHDL record construct which is not supported by many synthesis tools.
    - E.g. : sROLE_SHELL_Mmio_**DataValid**, sSHELL_ROLE_Mmio_DiagCtrl_**Start**.
  - [Suffix_] = a string indicating one or multiple properties of the signal.
  - - E.g. : sROLE_SHELL_Mmio_**DataValid**_n,
      sSHELL_ROLE_Mmio_DiagCtrl_**Start**_a.

Streamed Signal Names

The cFDK makes heavy use of data streaming interfaces and provides a dedicated naming convention for those streams. A streamed signal name is a short name for a set of signal that are grouped under the same streaming interface or a streaming sub-interface.

- A streamed signal follows the same naming conventions as the signal names except for the mandatory prefix with becomes ‘**ss**’ :
  ss[*FROM*]_[*TO*]_[*Itf*]_[*SubItf*]_**PortName**_[*Suffix*]
  with

- *ss* = a prefix indicating the '*streamed signal*' nature of this object. Here is an examples of 5 ports being part of the same stream:
    - ssSHL_ROL_Tcp_**Data**_tdata
    - ssSHL_ROL_Tcp_**Data**_tkeep
    - ssSHL_ROL_Tcp_**Data**_tlast
    - ssSHL_ROL_Tcp_**Data**_tvalid
    - ssSHL_ROL_Tcp_**Data**_tready,

Combining Signals, Ports and Instance Names

This section shows an example that combines the above listed convention names.
Figure-1

Figure-1: Combining signals, ports and instances names

Secondary Conventions

Constant Names

A constant name might be prefixed with the letter '***c***' in lower-case.

- E.g.: cAddrWidth, cDataLen

Function Names

A function name might be prefixed with the letter '***f***' in lower-case.

- E.g.: fLog2Ceil

Generics

A generic information might be prefixed with the letter '***g***' in lower-case.

- E.g.: gBusWidth

Process Names

A process name might be prefixed with the letter '***p***' in lower-case.

- E.g.: pMmioWrReg, pCheckCrc

Type ans Sub-type Names

A type definition might be prefixed with the letter '***t***' and a sub-type wit the letters '***st***' in lower-case.

- E.g.: tQByte, stIpTotalLen

Variable Names

A variable name might be prefixed with the letter '***v***' in lower-case.

- E.g.: vCurrent, vNext

Prefixes

Prefix	Structure	Description
pi	Port-In	An input port
pio	Port-InOut	A bidirectional port
po	Port-Out	An output port
s	Signal	A signal (alias wire)
si	Stream-In	An input port that is part of a streaming interface
so	Stream-Out	An output port that is part of a streaming interface
ss	Stream-Signal	A signal that is part of a streaming interface

Secondary Prefixes

Prefix	Structure	Description
c	Constant	A constant name
f	Function	A function name
g	Generics	A generic statement
p	Process	A process name
st	Data Type	A sub-type definition name
t	Data Type	A type definition name
v	Variable	A variable name

Suffixes

Prefix	Structure	Description
Add|Addr	Signal	An address type of signal
Clk|Clock	Signal	A clock type of signal
Dat|Data	Signal	A data type of signal
Comb	Process	A combinational type of process
Reg	Process	A register type of process
Rst|Reset	Signal	A reset type of signal
_a	Signal	An synchronous type of signal
_n	Signal	An active low type of signal
_tdata	Stream	A streamed data type
_tkeep	Stream	A streamed keep type

Prefix	Structure	Description
_tlast	Stream	A streamed last type
_tvalid	Stream	A streamed valid type
_tready	Stream	A streamed ready type
_z	Signal	A three-state type of signal