

Absolute encoder package

Fabien Le Mentec
lementec@esrf.fr

version: SVN revision 550:553M

Contents

1	Description	3
1.1	Overview	3
1.2	Features and limitations	4
1.3	Tested hardware	4
1.4	Performances	5
2	Interfaces	6
2.1	absenc_pkg.master	6
2.2	absenc_pkg.slave	6
3	Examples	8
3.1	master	8
3.2	slave	8

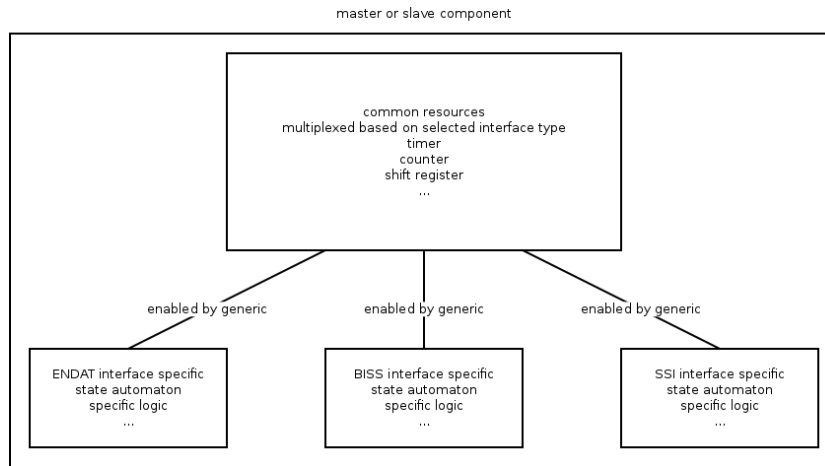
1 Description

1.1 Overview

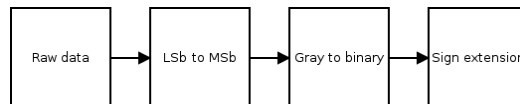
The `absenc_pkg` implements components for absolute encoder masters and slaves.

The term *master* refers to the component driving the clock, or at least initiating the data transfer. It is often referred to as the *controller*. The term *slave* refers to the actual encoder device.

This package is optimized for applications that can be dynamically configured to use one amongst different types of interfaces at a particular time. As much as possible, resources that can be shared across interfaces are factorized (counters, comparators, shift registers ...) and accessed through a multiplexer. However, and in order to avoid penalizing simpler applications, static configuration allows to exclude resources associated with an unused interface.



The master implements the following data conversion pipeline:



1.2 Features and limitations

- Applicable to all
 - master and slave modes,
 - configurable master clock frequency,
 - configurable data length (up to 48 bits),
 - static timeout.
- ENDAT
 - send position mode version 2.1 only,
 - no CRC check.
- BISS
 - point to point configuration only,
 - no interleaved bit support (ie. CDS, CDM).
- SSI
 - optional gray data coding (master only),
 - optional parity bit (not checked).
- HSSL
 - reader only,
 - stand alone, no master or slave interface.

1.3 Tested hardware

The package has been tested with the following masters:

- ICEPAP ESRF motor controller (SSI),
- PEPUP ESRF encoder processor (SSI, BISS, ENDAT),
- MUSST ESRF sequencing platform (SSI),
- Aerotech Soloist CP controller (BISS, ENDAT).

The package has been tested with the following slaves:

- PEPU ESRF encoder processor (SSI, BISS, ENDAT),
- Kubler sendix 5863 SIL BISS encoders,
- IVO industries GA241 SSI encoders,
- Heidenhain ROQ 425 512 ENDAT encoders,
- Heidenhain ROQ 437 2048 ENDAT encoders.

1.4 Performances

TODO:

2 Interfaces

2.1 absenc_pkg.master

Interface from file ../../src/absenc_pkg.vhd at line 390

```
component master
generic
(
  -- local clock frequency
  CLK_FREQ: integer;

  -- enable a specific implementation, or all by default
  -- disabling unneeded implementations optimizes resources
  ENABLE_ENDAT: boolean := TRUE;
  ENABLE_BISS: boolean := TRUE;
  ENABLE_SSI: boolean := TRUE
);
port
(
  -- local clock and reset
  clk: in std_logic;
  rst: in std_logic;

  -- ma_clk is the clock signal generated by the master
  -- to the slave. its frequency is CLK_FREQ / ma_fdiv
  ma_fdiv: in unsigned;
  ma_clk: out std_logic;

  -- master out, slave in
  mosi: out std_logic;
  miso: in std_logic;

  -- gate to drive output (1 to drive it). used in PEPUs
  -- first versions, set to open if not used.
  gate: out std_logic;

  -- data received from slave. typically the position
  -- important: use the smallest possible width, as
  -- this is used to derive other resource sizes
  data: out std_logic_vector;

  -- data length. typically the encoder resolution
  -- important: use the smallest possible width, as
  -- this is used to derive other resource sizes
  len: in unsigned;

  -- the selected encoder type
  enc_type: in integer;

  -- ssi specific flags
  -- set to work.absenc_pkg.SSI_DEFAULT_FLAGS if unused
  -- ssi_flags<0>: ssi spy mode (ie. master without clock)
  -- ssi_flags<1>: 0 or 1 for binary or gray data coding
  -- ssi_flags<2>: '.S' terminating pattern
  -- ssi_flags<3>: 'ES' terminating pattern
  -- ssi_flags<4>: 'OS' terminating pattern
  ssi_flags: in std_logic_vector;

  -- ssi frame delay. divides CLK_FREQ
  -- set to work.absenc_pkg.SSI_DEFAULT_DELAY_FDIV if unused
  ssi_delay_fdiv: in unsigned
);
end component;
```

2.2 absenc_pkg.slave

Interface from file ../../src/absenc_pkg.vhd at line 183

```

component slave
generic
(
  -- local clock frequency
  CLK_FREQ: integer;

  -- enable a specific implementation, or all by default
  -- disabling unneeded implementations optimizes resources
  ENABLE_ENDAT: boolean := TRUE;
  ENABLE_BISS: boolean := TRUE;
  ENABLE_SSI: boolean := TRUE
);
port
(
  -- local clock and reset
  clk: in std_logic;
  rst: in std_logic;

  -- clock from master
  ma_clk: in std_logic;

  -- master in, slave out
  miso: out std_logic;
  mosi: in std_logic;

  -- gate to drive output (1 to drive it). used in PEPUs
  -- first versions, set to open if not used.
  gate: out std_logic;

  -- data sent to master. typically the position
  -- important: use the smallest possible width, as
  -- this is used to derive other resource sizes
  data: in std_logic_vector;

  -- data length. typically the encoder resolution
  -- important: use the smallest possible width, as
  -- this is used to derive other resource sizes
  len: in unsigned;

  -- the selected encoder type
  enc_type: in integer;

  -- ssi specific flags
  -- set to work.absenc_pkg.SSI_DEFAULT_FLAGS if unused
  -- ssi_flags<0>: unused
  -- ssi_flags<1>: 0 or 1 for binary or gray data coding
  -- ssi_flags<2>: '.S' terminating pattern
  -- ssi_flags<3>: 'ES' terminating pattern
  -- ssi_flags<4>: 'OS' terminating pattern
  ssi_flags: in std_logic_vector
);
end component;

```

3 Examples

3.1 master

Example from file `./../sim/common/main.vhd` at line 228

```
master: work.absenc_pkg.master
generic map
(
  CLK_FREQ => CLK_FREQ
)
port map
(
  clk => clk,
  rst => rst,
  ma_fdiv => ma_fdiv,
  ma_clk => ma_clk,
  mosi => mosi,
  miso => miso,
  gate => open,
  data => master_data,
  len => len,
  enc_type => enc_type,
  ssi_flags => work.absenc_pkg.SSI_DEFAULT_FLAGS,
  ssi_delay_fdiv => work.absenc_pkg.SSI_DEFAULT_DELAY_FDIV
);
```

3.2 slave

Example from file `./../sim/common/main.vhd` at line 208

```
slave: work.absenc_pkg.slave
generic map
(
  CLK_FREQ => CLK_FREQ
)
port map
(
  clk => clk,
  rst => rst,
  ma_clk => ma_delayed_clk,
  miso => miso,
  mosi => mosi,
  gate => open,
  data => slave_data,
  len => len,
  enc_type => enc_type,
  ssi_flags => work.absenc_pkg.SSI_DEFAULT_FLAGS
);
```