

Absolute encoder package

Fabien Le Mentec
lementec@esrf.fr

version: SVN revision exported

Contents

1	Licence	3
2	Description	4
2.1	Overview	4
2.2	Supported features	4
2.3	Performances	4
2.4	Architecture	4
3	Interfaces	5
3.1	abs_enc_pkg.master	5
3.2	abs_enc_pkg.slave	5
4	Examples	7
4.1	master	7
4.2	slave	7

1 Licence

Note from file `../src/abs_enc_pkg.vhd` at line 12

```
-- this package and the associated files are distributed
-- under IDCL, the I Dont Care Licence. You are allowed
-- to do whatever you want with it, at your own risk and
-- profit. The author cannot be held responsible for any
-- damage, including rocket crash (while he would be very
-- interested to know if this code is used in space).
```

2 Description

2.1 Overview

The `abs_enc_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 sometimes called the controller. The term *slave* refers to the actual encoder device.

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

2.2 Supported features

TODO:

The following encoder types are available:

- ENDAT (version 2.1, send position mode),
- BISS,
- SSI.

2.3 Performances

TODO:

2.4 Architecture

TODO:

3 Interfaces

3.1 abs_enc_pkg.master

Interface from file ../../src/abs_enc_pkg.vhd at line 343

```
component master
generic
(
  CLK_FREQ: integer;
  ENABLE_ENDAT: boolean := TRUE;
  ENABLE_BISS: boolean := TRUE;
  ENABLE_SSI: boolean := TRUE
);
port
(
  -- local clock
  clk: in std_logic;
  rst: in std_logic;

  -- generated clock for slave
  -- fdiv divides CLK_FREQ
  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)
  gate: out std_logic;

  -- data from slave, and data length
  data: out std_logic_vector;
  len: in unsigned;

  -- encoder type
  enc_type: in integer
);
end component;
```

3.2 abs_enc_pkg.slave

Interface from file ../../src/abs_enc_pkg.vhd at line 163

```
component slave
generic
(
  CLK_FREQ: integer;
  ENABLE_ENDAT: boolean := TRUE;
  ENABLE_BISS: boolean := TRUE;
  ENABLE_SSI: boolean := TRUE
);
port
(
  -- local clock
  clk: in std_logic;
  rst: in std_logic;

  -- master clock
  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)
  gate: out std_logic;
```

```
-- data and length
data: in std_logic_vector;
len: in unsigned;

-- encoder type
enc_type: in integer
);
end component;
```

Note from file ../../sim/common/main.vhd at line 91

```
-- by default, the master implements all the encoder
-- types. this can be specialized by using the generics
-- to limit resource consumption.
```

AN ERROR OCCURED

4 Examples

4.1 master

Example from file `./../sim/common/main.vhd` at line 100

```
master: work.abs_enc_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
);
```

4.2 slave

Example from file `./../sim/common/main.vhd` at line 71

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