

fifo_pipe.v

AUTHORS

JAY CONVERTINO

DATES

2021/06/29

INFORMATION

Brief

Pipe fifo signals to help with timing issues, if they arise.

License MIT

Copyright 2021 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

fifo_pipe

```
module fifo_pipe #(
  parameter
  RD_SYNC_DEPTH
  =
  0,
  parameter
  WR_SYNC_DEPTH
  =
  0,
  parameter
  DC_SYNC_DEPTH
  =
  0,
  parameter
```

```

BYTE_WIDTH
=
1,
parameter
DATA_ZERO
=
0,
parameter
COUNT_WIDTH
=
1
)

input
rd_clk,
input
rd_rstn,
input
rd_en,
input
rd_valid,
input
[(BYTE_WIDTH*8)-1:0]
rd_data,
input
rd_empty,
output
r_rd_en,
output
r_rd_valid,
output
[(BYTE_WIDTH*8)-1:0]
r_rd_data,
output
r_rd_empty,
input
wr_clk,
input
wr_rstn,
input
wr_en,
input
wr_ack,
input
[(BYTE_WIDTH*8)-1:0]
wr_data,
input
wr_full,
output
r_wr_en,
output
r_wr_ack,
output
[(BYTE_WIDTH*8)-1:0]
r_wr_data,
output
r_wr_full,
input
data_count_clk,
input
data_count_rstn,
input
[COUNT_WIDTH:0]
data_count,
output
[COUNT_WIDTH:0]

```

(

```
r_data_count  
)
```

Pipe fifo signals to help with timing issues, if they arise.

Parameters

BYTE_WIDTH parameter	How many bytes wide the data in/out will be.
COUNT_WIDTH parameter	Data count output width in bits. Should be the same power of two as fifo depth(256 for fifo depth... this should be 8).
RD_SYNC_DEPTH parameter	Add in pipelining to read path. Defaults to 0.
WR_SYNC_DEPTH parameter	Add in pipelining to write path. Defaults to 0.
DC_SYNC_DEPTH parameter	Add in pipelining to data count path. Defaults to 0.
DATA_ZERO parameter	Zero out data output when enabled.

Ports

rd_clk input	Clock for read data
rd_rstn input	Negative edge reset for read.
rd_en input	Active high enable input of read interface.
rd_valid input	Active high output input that the data is valid.
rd_data input [(BYTE_WIDTH* 8)- 1:0]	Output data input
rd_empty input [(BYTE_WIDTH* 8)- 1:0]	Registered Active high output when read is empty.
r_rd_en output [(BYTE_WIDTH* 8)- 1:0]	Registered Active high enable of read interface.
r_rd_valid output [(BYTE_WIDTH* 8)- 1:0]	Registered Active high output that the data is valid.
r_rd_data output [(BYTE_WIDTH* 8)- 1:0]	Registered Output data
r_rd_empty output [(BYTE_WIDTH* 8)- 1:0]	Active high output when read is empty.
wr_clk input [(BYTE_WIDTH* 8)- 1:0]	Clock for write data
wr_rstn input [(BYTE_WIDTH* 8)- 1:0]	Negative edge reset for write
wr_en input [(BYTE_WIDTH* 8)- 1:0]	Active high enable of write interface, feed into register.
wr_ack input [(BYTE_WIDTH* 8)- 1:0]	Active high when enabled, that data write has been done, feed into register.
wr_data input [(BYTE_WIDTH* 8)- 1:0]	Input data, feed into register.
wr_full input [(BYTE_WIDTH* 8)- 1:0]	Active high output that the FIFO is full, feed into register.
r_wr_en	Register Active high enable of write interface.

```
output [(BYTE_WIDTH* 8)- 1:0]
```

r_wr_ack

```
output [(BYTE_WIDTH* 8)- 1:0]
```

Register Active high when enabled, that data write has been done.

r_wr_data

```
output [(BYTE_WIDTH* 8)- 1:0]
```

Register Input data

r_wr_full

```
output [(BYTE_WIDTH* 8)- 1:0]
```

Register Active high output that the FIFO is full.

data_count_clk

```
input [(BYTE_WIDTH* 8)- 1:0]
```

Clock for data count

data_count_rstn

```
input [(BYTE_WIDTH* 8)- 1:0]
```

Negative edge reset for data count.

data_count

```
input [COUNT_WIDTH:0]
```

Output that indicates the amount of data in the FIFO.