

## Key Design Features

- Synthesizable, technology independent VHDL Core
- Fully synchronous design
- Configurable depth and data width
- Register or RAM-based storage<sup>1</sup>
- Full/Empty flags and FIFO fullness counter
- Simple valid/ready pipeline interface protocol
- Output register option for improved timing
- 1 cycle latency<sup>2</sup>

## Applications

- General purpose buffering and rate adaptation
- Interfacing between other pipeline elements
- Registering the data-path in a pipeline to improve timing

## Pin-out Description

Pin name	I/O	Description	Active state
clk	in	Synchronous clock	rising edge
reset	in	Asynchronous reset	low
fifo_full	out	FIFO full flag	high
fifo_empty	out	FIFO empty flag	high
fifo_fullness	out	Counter indicating number of entries in the FIFO that are currently used	counter
datain [dw-1: 0]	in	FIFO input data	data
datain_val	in	Indicates valid data at the FIFO input	high
datain_rdy	out	Indicates that the FIFO is ready to accept data	high
dataout [dw-1:0]	out	FIFO output data	data
dataout_val	out	Indicates that the FIFO contains valid data	high
dataout_rdy	in	Indicates that the output is ready to receive data	high

## Block Diagram

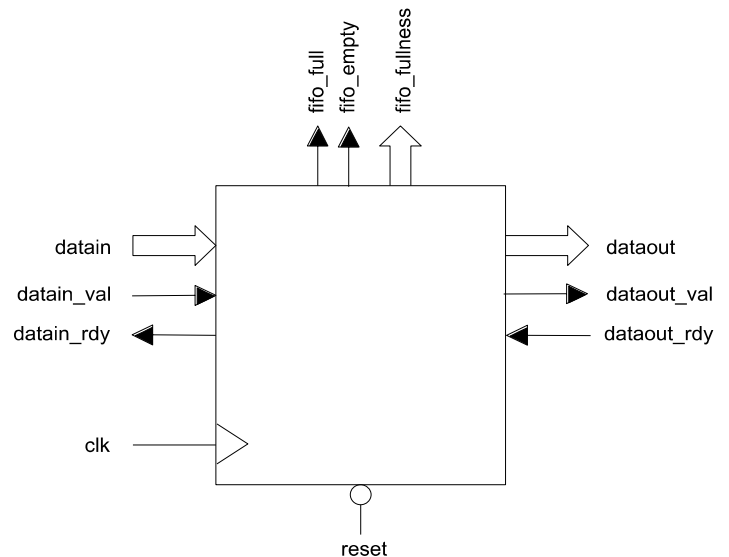


Figure 1: Synchronous FIFO

## Generic Parameters

Generic name	Description	Type	Valid range
dw	FIFO data width	integer	≥ 1
depth	FIFO depth	integer	≥ 2
log2d	Log2 FIFO depth	integer	log2 (depth)
regout	Enable output data register - improves output timing when enabled	boolean	TRUE/FALSE

## General Description

FIFO\_SYNC is a general purpose synchronous FIFO with configurable depth and data width. The FIFO uses register-based storage by default. If RAM-based storage is preferred, then the synthesis tool will normally infer a RAM if the output register option is selected.

Data flows in and out of the FIFO in accordance with the valid/ready protocol. Data is written to the FIFO on a rising clock-edge when *datain\_val* is high and *datain\_rdy* is high. Data is read from the FIFO on a rising clock-edge when *dataout\_val* is high and *dataout\_rdy* is high.

In addition to these handshake signals, the flags *fifo\_full* and *fifo\_empty* are also provided for a more traditional approach to FIFO interfacing. The signals *datain\_val* and *dataout\_rdy* can respectively be considered as the traditional write and read strobes into the FIFO. The counter value *fifo\_fullness* indicates how many entries are currently used in the FIFO. This may be useful in the generation of user defined flags for 'almost full' and 'almost empty' conditions.

<sup>1</sup> Type of inferred storage depends on synthesis tool configuration

<sup>2</sup> If output register is enabled then latency is 2 cycles

It is important to note that when the FIFO is configured with the output register set to true, then the signals `fifo_empty` and `fifo_fullness` are delayed by 1 clock cycle due to the 1 cycle latency of the output register. For this reason it is recommended that only the valid/ready handshake signals are used for flow-control when the output register is enabled.

Normally, if the FIFO is empty, then there is a 1 clock cycle delay between a write at the input and the read data being available at the output. If the output register is enabled, then the delay is 2 clock cycles.

## Functional Timing Diagrams

The following timing diagrams relate to a 16-deep FIFO design with the output register option set to false.

Figure 2 shows a FIFO write operation when the FIFO state changes from almost full to full. Figure 3 demonstrates a FIFO read operation when the FIFO state changes from almost empty to empty. Note that data is only transferred at the FIFO interfaces on a rising clock edge when valid and ready are both active high.

For more examples of the valid/ready protocol and it's implementation see ZIPcores application note: [app\\_note\\_zc001.pdf](#).

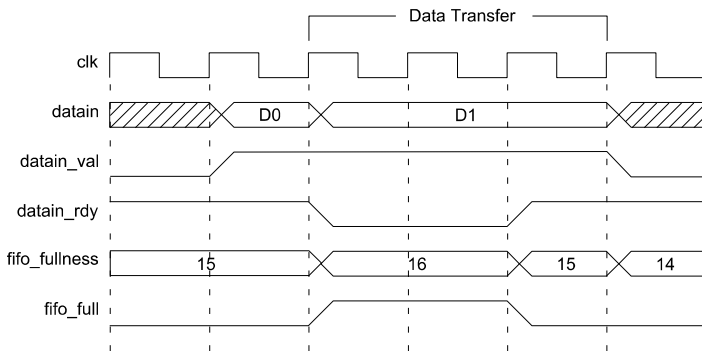


Figure 2: FIFO write operation

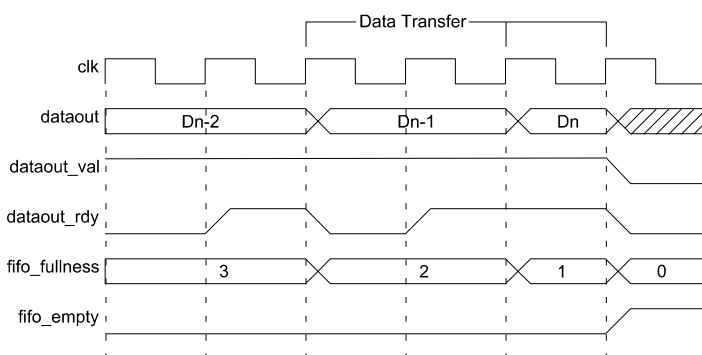


Figure 3: FIFO read operation

## Source File Description

All source files are provided as text files coded in VHDL. The following table gives a brief description of each file.

Source file	Description
pipeline_reg.vhd	FIFO output register for the case when the generic <code>regout</code> is set to true
fifo_sync.vhd	Top-level block
fifo_sync_bench.vhd	Top-level test bench

## Functional Testing

An example VHDL testbench is provided for use in a suitable VHDL simulator. The compilation order of the source code is as follows:

1. pipeline\_reg.vhd
2. fifo\_sync.vhd
3. fifo\_sync\_bench.vhd

The VHDL testbench instantiates the FIFO component and the user may modify the generic parameters as required.

The simulation must be run for at least 3 ms during which time a randomized input data sequence will be written to the FIFO.

In addition to random input data, the test bench also generates randomized valid/ready handshake signals at the input and output of the FIFO in order to verify that the the flow-control is working correctly.

The simulation generates two text files: `fifo_sync_in.txt` and `fifo_sync_out.txt`. These files respectively contain the input and output data captured at the FIFO interfaces during the test. If the files match then the test has been successful.

## Synthesis

The files required for synthesis and the design hierarchy is shown below:

- fifo\_sync.vhd
- pipeline\_reg.vhd

The VHDL core is designed to be technology independent. However, as a benchmark, synthesis results have been provided for the Xilinx Virtex 5 and the Altera Stratix III series of FPGA devices. The lowest and highest speed grade devices have been chosen in both cases for comparison.

Note that as the FIFO is highly generic in nature, the synthesis results will vary significantly with choice of parameters. It is also important to note that the type of inferred storage used by the FIFO is largely dependent on the attributes set up in the synthesis tool. If the output register is enabled, then internally, a FIFO memory read has a 1 clock-cycle latency. This permits the tool to infer a dual-port RAM. If the output register is disabled, then distributed RAM (registers) are generally inferred.

Trial synthesis results are shown with the generic parameters set to: dw = 16, depth = 32, log2d = 5, regout = true.

Resource usage is specified after Place and Route.

#### VIRTEX 5

<b>Resource type</b>	<b>Quantity used</b>
Slice register	41
Slice LUT	51
Block RAM	0
DSP48	0
Clock frequency (worst case)	285 MHz
Clock frequency (best case)	430 MHz

#### STRATIX III

<b>Resource type</b>	<b>Quantity used</b>
Register	545
ALUT	207
Block Memory bit	0
DSP block 18	0
Clock frequency (worse case)	305 MHz
Clock frequency (best case)	392 MHz

#### Revision History

<b>Revision</b>	<b>Change description</b>	<b>Date</b>
1.0	Initial revision	16/04/2008