Key Design Features

- Synthesizable, technology independent VHDL IP Core
- Dual-clock architecture
- Configurable data width and depth
- Gray-encoded read/write pointers
- Full / Empty flags
- Uses a simple valid-ready streaming protocol
- Compatible with other streaming protocols such as: AMBA® AXI4-stream, Altera® Avalon-ST and Xilinx® local-link
- Output register option for improved timing
- Very high-speed operation achieving 400 MHz+ on basic FPGA devices

Applications

- Re-synchronizing between clock domains
- Registering the data-path on and off chip
- General purpose buffering
- Adapting to different data rates - e.g. digital video applications with varying pixel clock frequencies or frame rates

Pin-out Description

Pin name	I/O	Description	Active state
clk_a	in	Input clock domain a	rising edge
clk_b	in	Input clock domain b	rising edge
reset	in	Asynchronous reset	low
fifo_full	out	FIFO full flag (clock domain a)	high
fifo_empty	out	FIFO empty flag (clock domain b)	high
a_data [dw-1:0]	in	FIFO input data	data
a_val	in	Indicates valid data at the FIFO input	high
a_rdy	out	Indicates that the FIFO is ready to accept data	high
b_data [dw-1:0]	out	FIFO output data	data
b_val	out	Indicates that the FIFO contains valid data	high
b_rdy	in	Indicates that the output is ready to receive data	high

Block Diagram

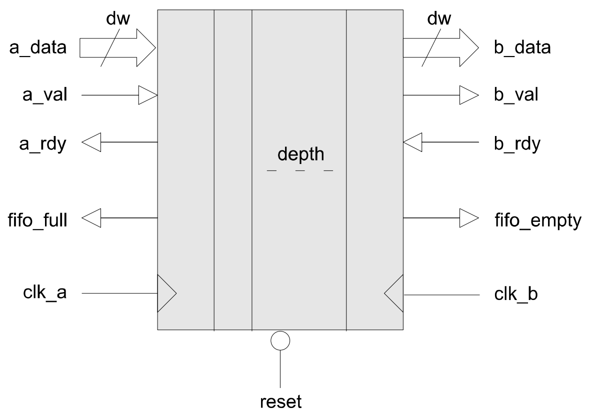

Figure 1: Asynchronous FIFO

Generic Parameters

Generic name	Description	Type	Valid range
dw	FIFO data width	integer	≥ 1
depth	FIFO depth	integer	8 or 16
regout	Enable output data register - improves output timing	boolean	TRUE/FALSE

General Description

FIFO_ASYNC (Figure 1) is an Asynchronous FIFO with a configurable depth of 8 or 16 entries and a generic data width. The FIFO uses register-based storage by default.

The FIFO has two independent clocks 'a' and 'b' in order to allow the re-synchronization of the datapath across different clock domains. Internally, the FIFO uses gray-encoded read and write pointers. This ensures that the pointers are re-synchronized as single-bit transitions only.

Data flows in and out of the FIFO in accordance with the valid-ready pipeline protocol[1]. Data is written to the FIFO on the rising clock-edge of *clk_a* when *a_val* is high and *a_rdy* is high. Data is read from the FIFO on the rising clock-edge of *clk_b* when *b_val* is high and *b_rdy* is high.

In addition to these handshake signals, the flags *fifo_full* and *fifo_empty* are also provided for compatibility with standard FIFO interfaces. The signal *fifo_full* is synchronized to clock domain 'a' and *fifo_empty* to clock domain 'b'. Both signals immediately reflect the internal state of the FIFO. The signals *a_val* and *b_rdy* can be considered as the traditional write and read strobes into the FIFO.

It is important to note that when the FIFO is configured with the output register set to true, then the signal *fifo_empty* is delayed by 1 clock cycle due to the 1 cycle latency of the output register.

1 For more examples of the valid-ready protocol and it's implementation see Zipcores application note: app_note_zc001.pdf.

Functional Timing Diagrams

The following timing diagrams relate to a 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.

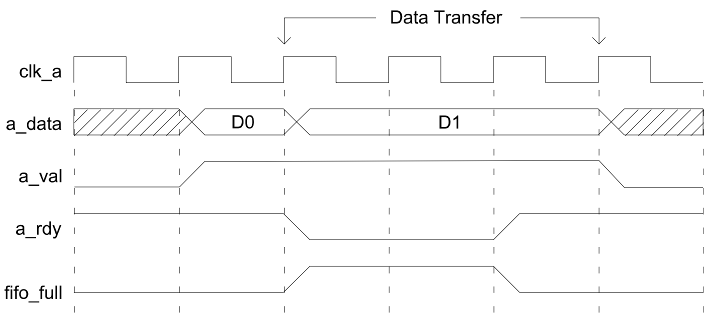

Figure 2: FIFO write operation

Figure 3, below, demonstrates a FIFO read operation when the FIFO state changes from almost empty to empty.

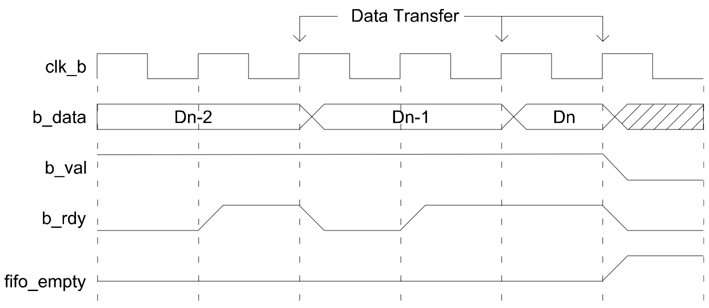

Figure 3: FIFO read operation

Note that data is only transferred at the FIFO interfaces on a rising clock edge when valid and ready are both active high.

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	Output register used when the parameter *regout* is set to *true*
fifo_async8.vhd	8-deep FIFO component
fifo_async16.vhd	16-deep FIFO component
fifo_async.vhd	Top-level block
fifo_async_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_async.vhd
3. fifo_async8.vhd
4. fifo_async16.vhd
5. fifo_async_bench.vhd

The VHDL testbench instantiates the FIFO component and the user may modify the generic parameters as required. In addition, the user may specify the frequency of *clk_a* and *clk_b* in order to observe the data flow under different asynchronous clocking conditions.

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 called: *fifo_async_in.txt* and *fifo_async_out.txt*. These files respectively contain the input and output data captured at the FIFO interfaces during the test. Matching input and output files indicates that the test has run successfully.

Synthesis

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

- fifo_async.vhd
  - fifo_async8.vhd
  - fifo_async16.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 6 and Spartan-6 FPGA devices. Synthesis results for other FPGAs and technologies can be provided on request.

Setting the parameter *regout* to *true* and using an 8-deep FIFO will result in the fastest and most efficient implementation. In this configuration, clock speeds of up to 500 MHz or better are attainable on some FPGA devices.

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

Resource usage is specified after Place and Route.

VIRTEX 6

Resource type	***Quantity used***
Slice register	71
Slice LUT	91
Block RAM	0
DSP48	0
Occupied slices	28
Clock frequency (approx)	420 MHz

SPARTAN 6

Resource type	***Quantity used***
Slice register	71
Slice LUT	89
Block RAM	0
DSP48	0
Occupied slices	30
Clock frequency (approx)	333 MHz

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision	21/04/2008
1.1	Updated synthesis results	21/04/2010
1.2	Added 8-deep and 16-deep options. Optimized for very high-speed operation	08/09/2011
1.3	Includes more efficient pipeline register component	07/02/2014