

Key Design Features

- Synthesizable, technology independent VHDL Core
- Fully synchronous design
- Configurable data width
- Simple valid/ready pipeline interface protocol
- Self flushing
- 1 cycle latency

Applications

- Interfacing between other pipeline elements
- Registering the datapath in a pipeline to improve timing
- Registering a datapath in and off chip
- Simple buffering

Pin-out Description

Pin name	I/O	Description	Active state
clk	in	Synchronous clock	rising edge
reset	in	Asynchronous reset	low
datain [dw-1:0]	in	Pipeline register input data	data
datain_val	in	Indicates valid data at the pipeline register input	high
datain_rdy	out	Indicates that the pipeline register is ready to accept data	high
dataout [dw-1:0]	out	Pipeline register output data	data
dataout_val	out	Indicates that the pipeline register contains valid data	high
dataout_rdy	in	Indicates that the output is ready to receive data	high

Generic Parameters

Generic name	Description	Type	Valid range
dw	data width	integer	≥ 1

Block Diagram

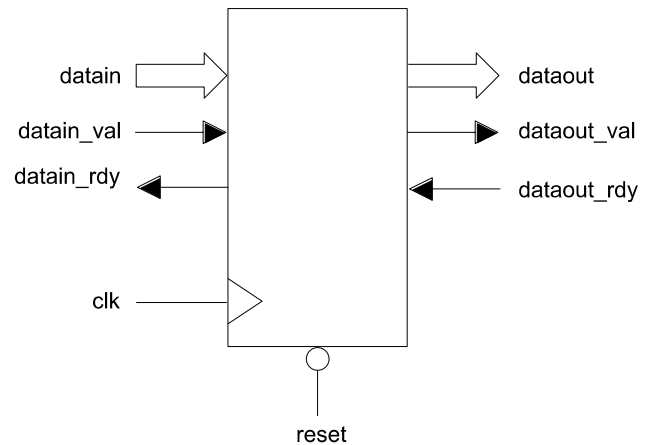


Figure 1: Pipeline register

General Description

PIPELINE_REG is a self-flushing register element used as a fundamental building block in pipelined architectures. It's principal use is to register the datapath between series elements in a pipeline and also to interface between blocks using the valid/ready pipeline protocol.

Data flows in and out of the pipeline register in accordance with the valid/ready protocol. Data is written to the register on a rising clock-edge when *datain_val* is high and *datain_rdy* is high. Data is read out of the register on a rising clock-edge when *dataout_val* is high and *dataout_rdy* is high. Pipeline_reg has a 1 clock cycle latency.

Pipeline_reg is self-flushing in that once data is written to the register then the register is full and the internal state-machine will maintain *dataout_val* asserted. The signal *dataout_val* will be held high until the downstream side of the pipeline asserts *dataout_rdy* to accept the data. Likewise, if the register is empty, then the state-machine will assert *datain_rdy* high to actively 'suck' data down the pipeline.

Functional Timing

Figure 2 shows a simple data transfer when the pipeline register is initially empty and the downstream side is stalling. The downstream side then asserts *dataout_rdy* high to complete the data transfer. Figure 3 demonstrates a series of sequential data transfers without stalling.

Note that data is only transferred at the pipeline register 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](#).

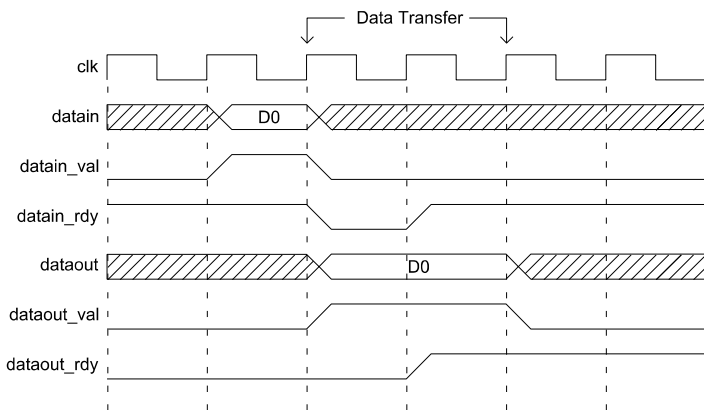


Figure 2: Pipeline stalling

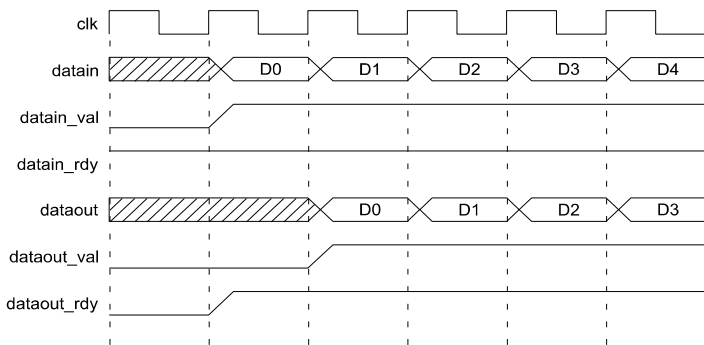


Figure 3: Sequential data transfers

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	Top-level block
pipeline_reg_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. pipeline_reg_bench.vhd

The VHDL testbench instantiates the pipeline register 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 pipeline register.

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

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

Synthesis

The source file 'pipeline_reg.vhd' is the only file required for synthesis. There are no sub-modules in the design.

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.

Trial synthesis results are shown with the generic parameters set to: dw = 32. Resource usage is specified after Place and Route.

VIRTEX 5

Resource type	Quantity used
Slice register	33
Slice LUT	34
Block RAM	0
DSP48	0
Clock frequency (worst case)	485 MHz
Clock frequency (best case)	634 MHz

STRATIX III

Resource type	Quantity used
Register	33
ALUT	2
Block Memory bit	0
DSP block 18	0
Clock frequency (worse case)	368 MHz
Clock frequency (best case)	500 MHz

Revision History

Revision	Change description	Date
1.0	Initial revision	19/04/2008