IOb-Cache

User Guide, 0.1 , Build af0c05f

May 18, 2022

Contents

List of Tables

List of Figures

1 Introduction

The IObundle CACHE is an open-source pipelined-memory cache. It is a performance-wise and highly configurable IP core. The cache core is isolated from the processor and memory interfaces in order to make it easy to adopt new processors or memory controllers while keeping the core functionality intact. It implements a simple front-end native interface. It also implements an AXI4 interface with configurable data width which allows maximum use of the available memory bandwidth. The IObundle CACHE can be implemented as a Direct-Mapped cache or K-Way Set-Associative cache. It supports both fixed write-through not-allocate policy and write-back policy.

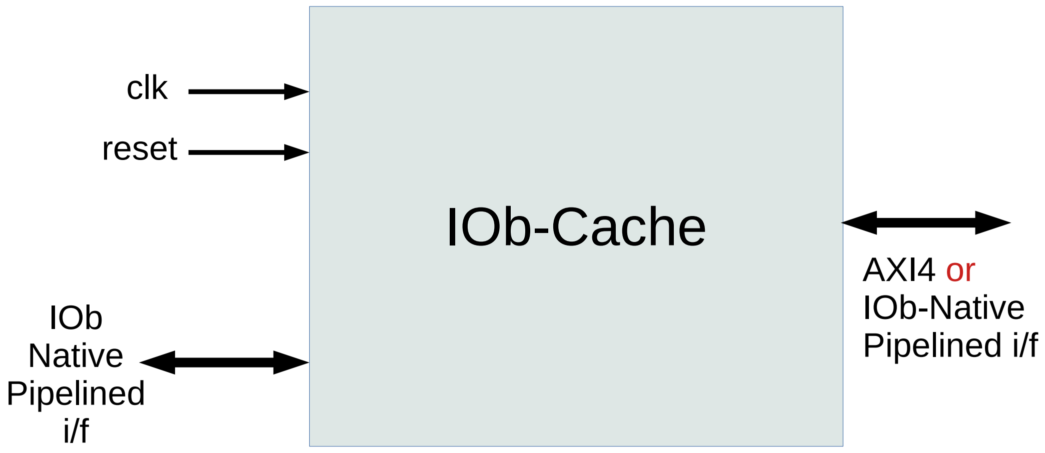

Figure 1: IP Core Symbol.

1.1 Features

- Pipelined-memory (1 request/clock-cycle)
- AXI4 interface with configurable data width
- Simple front-end native interface
- Direct-Mapped or K-Way Set-Associative
- Fixed write-through not-allocate policy
- Write-back policy

1.2 Benefits

- Compact and easy to integrate hardware and software implementation
- Can fit many instances in low cost FPGAs and ASICs
- Low power consumption

1.3 Deliverables

- ASIC or FPGA synthesized netlist or Verilog source code, and respective synthesis and implementation scripts

- ASIC or FPGA verification environment by simulation and emulation
- Bare-metal software driver and example user software
- User documentation for easy system integration
- Example integration in IOb-SoC (optional)

2 Description

2.1 Block Diagram

Figure 2 presents a high-level block diagram of the core, followed by a brief description of each block.

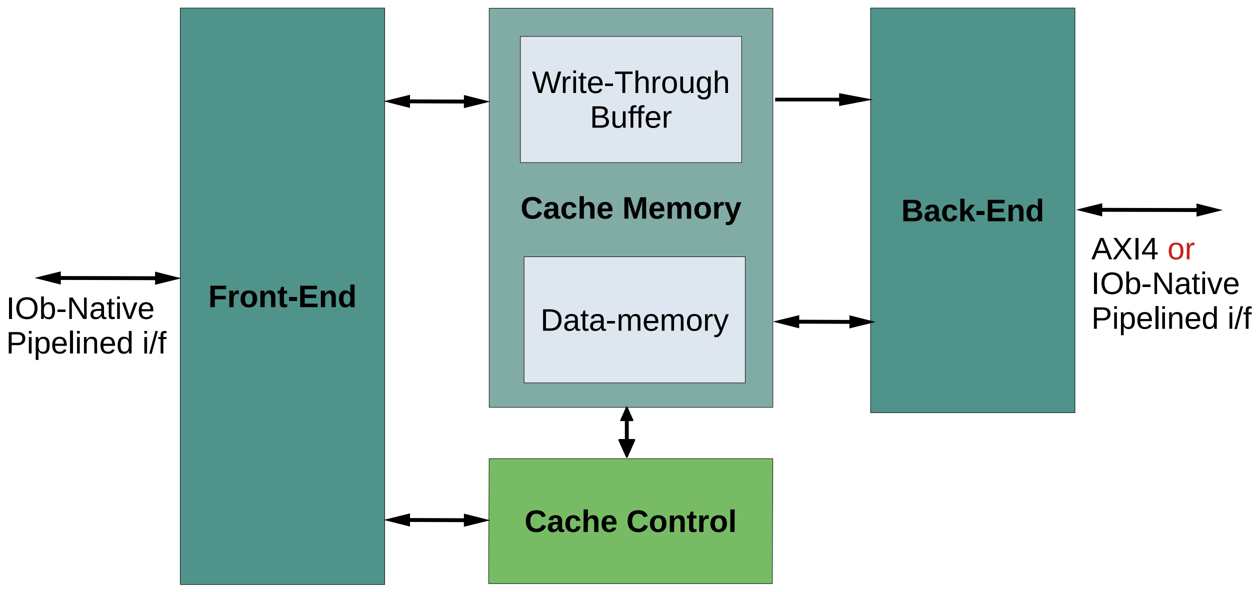

Figure 2: High-Level Block Diagram.

FRONT-END Front-end interface.

CACHE MEMORY This block implements the cache memory.

BACK-END INTERFACE This block interfaces with the system level or next-level cache.

CACHE CONTROL Cache control block.

FRONT-END Front-end interface.

CACHE MEMORY This block implements the cache memory.

BACK-END INTERFACE This block interfaces with the system level or next-level cache.

CACHE CONTROL Cache control block.

2.2 Configuration

2.2.1 Macros

The IP core has no user-definable macros other than the default configuration parameter values, described in Section 2.2.2 if available.

2.2.2 Parameters

The configuration parameters of the core are presented in Table 1. Configuration parameters can vary from instance to instance.

Parameter	Min	Typ	Max	Description
FE_ADDR_W	NS	32	64	The front-end address width defines the memory space accessible via the cache.
FE_DATA_W	32	32	64	Front-end data width
BE_ADDR_W		FE_ADDR_W	NS	Back-end address width. This width can be set higher than FE_ADDR_W to match the width of the back-end interface but the address space is still dictated by FE_ADDR_W.
BE_DATA_W	32	32	256	Back-end data width
NLINES_W	NS	7	NS	Line offset width: 2**NLINES_W is the number of cache lines
WORD_OFFSET_W	0	3	NS	Word offset width: 2**OFFSET_W is the number of words per line
WTBUF_DEPTH_W	NS	5	NS	Write-through buffer depth (log2)
REP_POLICY	0	PLRU_MRU	3	Line replacement policy. Set to 0 for Least Recently Used (LRU); set to 1 for Pseudo LRU based on Most Recently Used (PLRU_MRU); set to 2 for Tree-base Pseudo LRU (PLRU_TREE)
WRITE_POL	0	WRITE_THROUGH	1	Write policy: set to 0 for write-through or set to 1 for write-back
CTRL_CACHE	0	0	1	Instantiates a cache controller (1) or not (0). If the controller is present, all cache lines can be invalidated and the write through buffer empty status can be read
CTRL_CNT	0	0	1	If CTRL_CACHE=1 and CTRL_CNT=1 , the cache will include software accessible hit/miss counters

Table 1: Synthesis Parameters.

2.3 Interface Signals

Name	Direction	Width	Description
clk	INPUT	1	System clock
reset	INPUT	1	System reset, asynchronous and active high

Table 2: General Interface Signals

Name	Direction	Width	Description
req	INPUT	1	Read or write request from CPU or other user core. If `ack` becomes high in the next cyle the request has been served; otherwise `req` should remain high until `ack` returns to high. When `ack` becomes high in reponse to a previous request, `req` may be lowered in the same cycle ack becomes high if there are no more requests to make. The next request can be made while `ack` is high in reponse to the previous request
addr	INPUT	CTRL_CACHE+FE_ADDR_W-FE_NBYTES_W	Address from CPU or other user core, excluding the byte selection LSBs.
wdata	INPUT	FE_DATA_W	Write data fom host.
wstrb	INPUT	FE_NBYTES	Byte write strobe.
rdata	OUTPUT	FE_DATA_W	Read data to host.
ack	OUTPUT	1	Acknowledges that the last request has been served; the next request can be issued when this signal is high or when this signla is low but has already pulsed high in response to the last request.

Table 3: Front-End Interface Signals

Name	Direction	Width	Description
invalidate_in	INPUT	1	Invalidates all cache lines if high.
invalidate_out	OUTPUT	1	This output is asserted high whenever the cache is invalidated.
wtb_empty_in	INPUT	1	This input may be driven the next-level cache, when its write-through buffer is empty. It should be tied to high if there no next-level cache.
wtb_empty_out	OUTPUT	1	This output is high if the cache's write-through buffer is empty and the `wtb_empty_in` signal is high.

Table 4: Invalidate and Write-Through Buffer Empty Chain Interface Signals

Name	Direction	Width	Description
mem_req	OUTPUT	1	//Read or write request to next-level cache or memory. If `mem_ack` becomes high in the next cyle the request has been served; otherwise `mem_req` should remain high until `mem_ack` returns to high. When `ack` becomes high in reponse to a previous request, `mem_req` may be lowered in the same cycle ack becomes high if there are no more requests to make. The next request can be made while `mem_ack` is high in reponse to the previous request.
mem_addr	OUTPUT	BE_ADDR_W	Address to next-level cache or memory
mem_wdata	OUTPUT	BE_DATA_W	Write data to next-level cache or memory
mem_wstrb	OUTPUT	BE_NBYTES	Write strobe to next-level cache or memory
mem_rdata	INPUT	BE_DATA_W	Read data to host.
mem_ack	INPUT	1	//Acknowledges that the last request has been served; the next request can be issued when this signal is high or when this signal is low but has already pulsed high in reponse to the last request.

Table 5: Back-End Interface Signals

3 Usage

Figure 4 illustrates how to instantiate the IP core and, if applicable, the required external blocks.

bla bla bla...

3.1 Simulation

The provided testbench uses the core instance described in Section 3. A high-level block diagram of the testbench is shown in Figure 4. The testbench is organized in a modular fashion, with each test described in a separate file. The test suite consists of all the test case files to make adding, modifying, or removing tests easy.

In this preliminary version, simulation is not yet fully functional. The provided testbench merely allows compilation for simulation, and drives the clock and reset signals. Behavioural memory models to allow pre-synthesis simulation are already included. In the case of ROMs, their programming data is also included in the form of .hex files.

3.2 Synthesis

A simple `.tcl` script is provided for the Cadence Genus synthesis tool. The script reads the technology files, compiles and elaborates the design, and proceeds to synthesise it. The timing constraints are contained within the constraints file provided, or provided in a separate file.

After synthesis, reports on silicon area usage, power consumption, and timing closure are generated. A post-synthesis Verilog file is created, to be used in post-synthesis simulation.

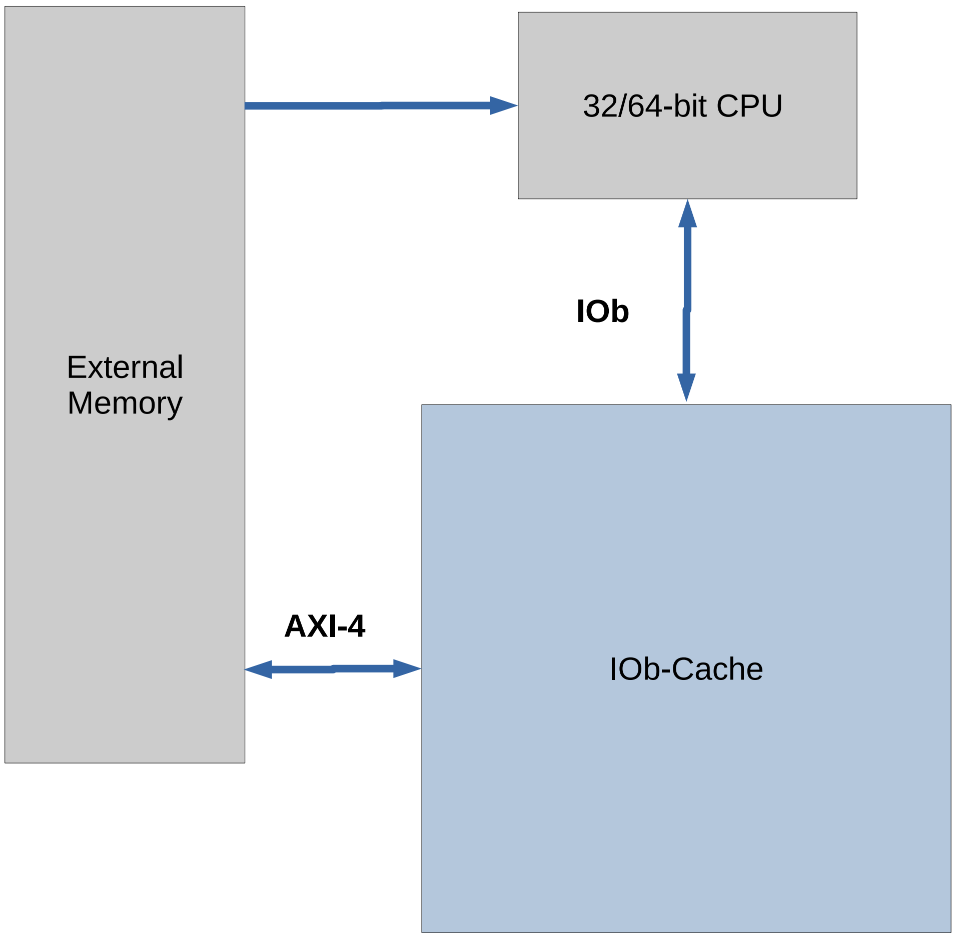

Figure 3: Core Instance and Required Surrounding Blocks

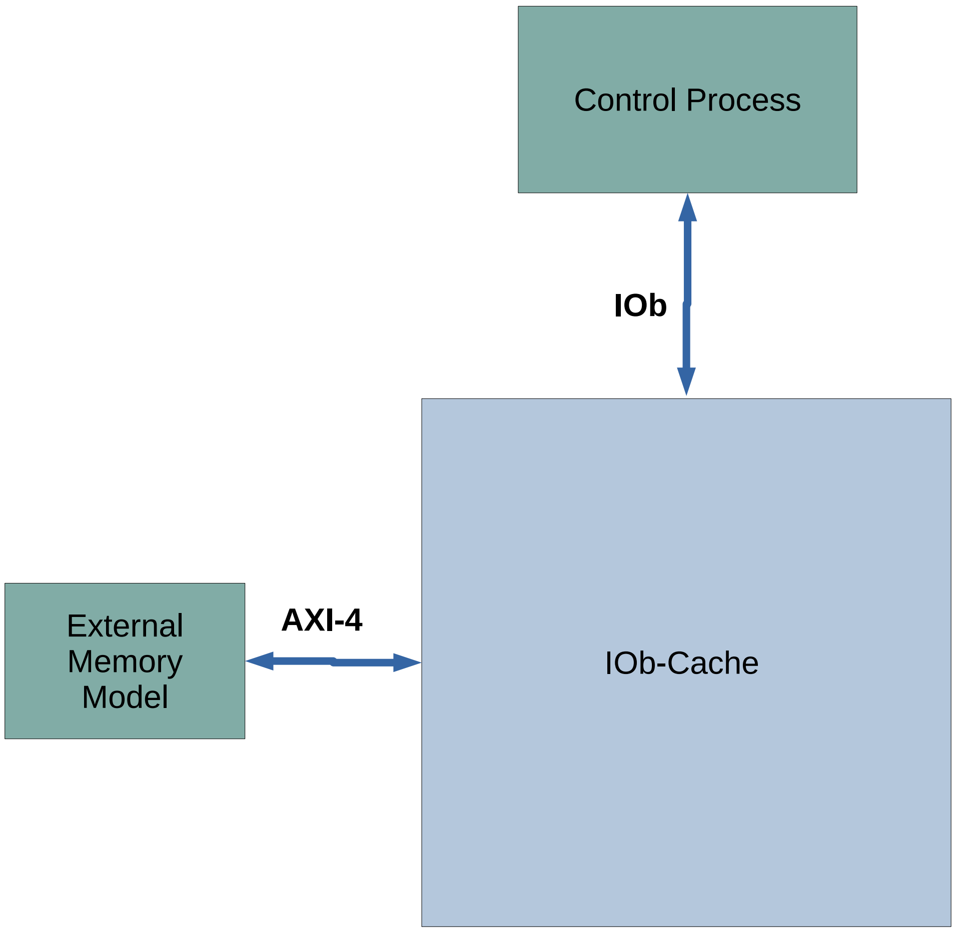

Figure 4: Testbench Block Diagram

In this preliminary version, synthesis of the IP core without the memories is functional. The memories are for now treated as black boxes.

It is important not to include the memory models provided in the simulation directory in synthesis, unless they are to be synthesised as logic. In a next version, the memories will be generated for the target technology and included.