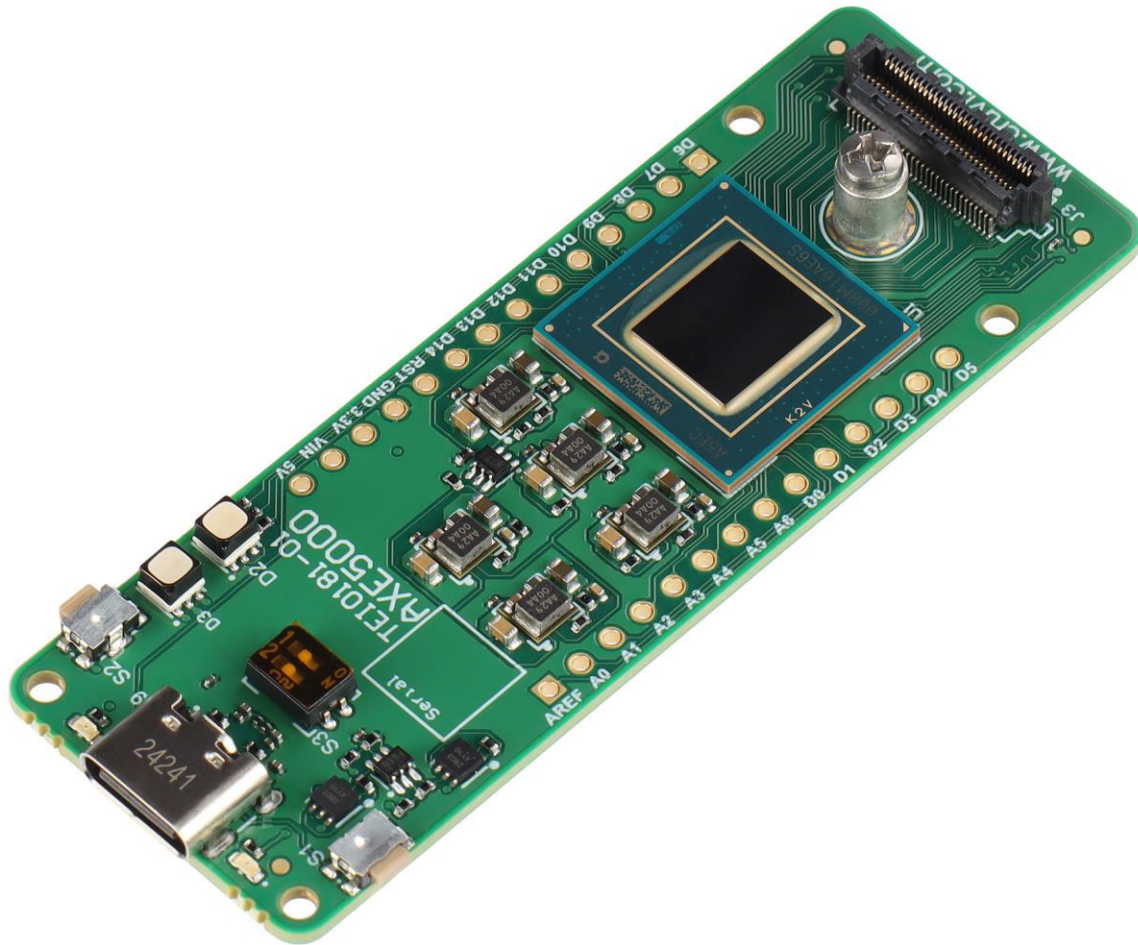


# OpenMBMC

IP Core user manual, document version 0.01



## Table of Contents

Introduction.....	3
License .....	3
Devices .....	4
HyperRAM .....	4
HyperFlash.....	4
OPI .....	4
FPGA support.....	5
Altera .....	5
AMD.....	5
Simple Test Design .....	5
Lattice .....	6
IP Configuration.....	8
Defines.....	8
Parameters .....	8
Board support.....	9
AXE5000 .....	9
Simulation.....	10
Avalon.....	10
AHBLite .....	10
Support .....	13

## Introduction

OpenMBMC (Open Multi Bus Memory Controller) is a portfolio of IP cores supporting various host busses and different Memory Devices.

The main goal for the IP core was to be simple to use and support all possible FPGA's.

## License

This open-source IP core is licensed under permissive MIT license.

MIT License

Copyright (c) 2025 MicroFPGA UG(h)

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.

## Devices

Currently only HyperBUS like devices are supported – this includes HyperRAM, HyperFlash, OPI RAM and xSPI Profile 2 devices. Not supported currently are Xccela and xSPI Profile 1 devices (they may be supported in the future).

### HyperRAM

The initial version of the IP core supported only HyperRAM devices.

### HyperFlash

HyperFlash was added as second device type. Following changes have been done compared to the HyperRAM only IP core:

- Latency count set to: 5
- Latency type set to: fixed
- Register initialization removed
- All writes converted to 16 bit writes with 0 latency (like HyperRAM register writes)
- RWDS strobe masking removed (not needed)

### OPI

As third device class, OPI PSRAM support was implemented. Following changes have been done compared to the HyperRAM mode:

- Latency type set to: fixed for APmemory devices (variable for Gigadevice)
- Register initialization changed
- Command bits reversed (swapped read and write commands)
- Address bits allocated at different CA bit positions

Status: in preparation, concept is ready need implement the verilog code!

## FPGA support

OpenMBMC is designed to be compatible with any FPGA technology, the IP core does not use any vendor primitives, and it is plain verilog code.

## Altera

For Altera tools packaged IP is provided, just place the IP core files to the ip directory under your project tree. All Altera FPGA's are supported. Initial testing was done on CR00100 board with MAX10, then it was again tested on AXE5000 with Agilex-5E FPGA.

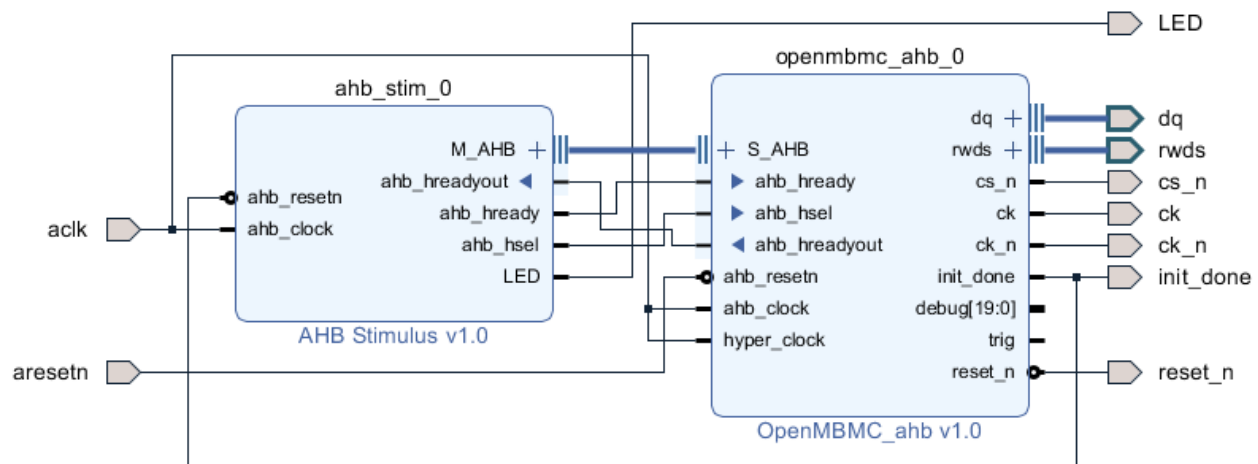
The IP core should just work, add to the IP core, connect the internal ports, export the hyperram bus and set address region, that is all that is needed.

## AMD

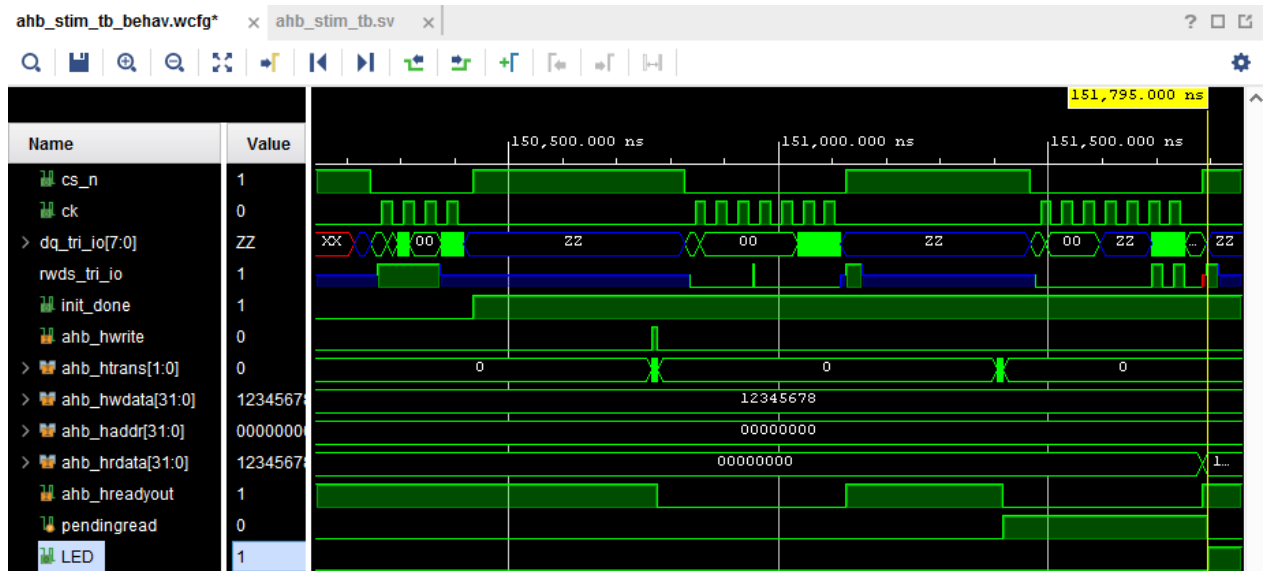
With AMD Vivado, the OpenMBMC packaged as Avalon IP core should be used. Vivado IP catalog provides free AXI to Avalon bridge IP, that should be used to connect the OpenMBMC to the system AXI bus. Vivado IP catalog ready IP core is provided, ready to be used.

## Simple Test Design

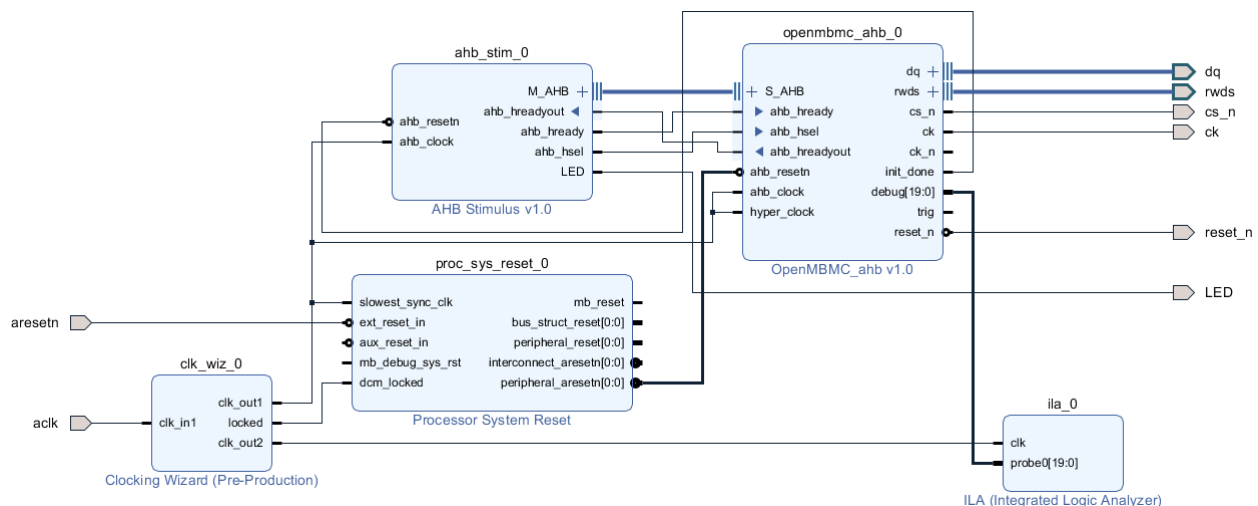
For go/no-go testing there is a simple test design that writes one 32 bit word to OpenMBMC (AHBLite version) and reads it back with verify indicating success or fail on single LED. It is possible to define the LED polarity, test address and data pattern.



This is how it looks in Vivado IPI. Notice the manual wiring for some AHBLite signals, this is necessary or the design would not work. This is ready to use BD for the use in simulation testbench that is also provided.



Simulation screenshot, visible register initialization, then write command and read command and response and LED going to one!



This is how the FPGA version looks like, we added Clock Wizard, Reset IP and ILA IP Cores. Also removed unnecessary top level ports. Ready to use project for CR00107 is provided, it will lit the green LED on success. To port to any other FPGA/Board you only need to change the FPGA part and constraints. And check the reset pin polarity and clock frequency for input clock.

## Lattice

To fully support Lattice Propel design environment we only need one IP core and that should use AHBLite bus. For SoC designs with larger RISC-V core that use AXI bus, the free AXI to AHBLite bus bridge should be used. All Lattice FPGA's are supported. The IP core was initially tested with CR00103 with Certus-NX FPGA.

The AHBLite version was developed using open-source AHBLite2Avalon IP core found at [github](https://github.com). Unfortunately this IP core (being only one commit and 9 years old!) did not work initially. We needed a

lot of work changing the logic for read/write and hready generation. We also had to add avl\_byteenable support to this IP core. After those changes we had a working simulation with AHBLite bus!

## IP Configuration

The IP core is configurable using verilog defines and parameters. Defines are used for platform settings that are later not editable in IP configuration GUI. Parameters are editable in the IP configuration GUI.

### Defines

There are some verilog defines that are used to configure some basic features of the IP core. If you are working with packaged IP catalog IP core then you cannot and should not modify those defines.

Platform defines are defined in the file named *openmbmc\_platform.v*, there are multiply copies of this file with different content. This file is included from the IP core toplevel file and it does configure the IP core for selected hardware platform.

`'define VIVADO`

Should be defined for compilation into Vivado IP catalog.

`'define QUARTUS`

Should be defined for compilation as Quartus QSYS IP component.

`'define PROPEL_EMU`

Should be defined if we “emulate” Lattice Propel IP core on Vivado.

`'define PROPEL`

Should be defined if we compile for Propel IP catalog.

### Parameters

Parameters define the selected functionality of the IP core. They are editable in the IP configuration GUI. The look and feel depends on the FPGA design environment. In some cases it is needed to enter numeric values, in some cases the selection can be done with drop down menu.

**G\_DEVICE** is integer parameter that defines the device type/family being used.

G_DEVICE Value	Device type or family
0	HyperRAM – Generic HyperRAM/HyperBUS device
1	HyperFlash
2	OPI PSRAM

Currently only HyperRAM and HyperFlash are supported.

**G\_ADDRBITS** defines the number of used Address bits (in byte addressing mode).

G_ADDRBITS Value	Mbits	Mbytes
22	32	4
23	64	8
24	128	16
25	256	32
26	512	64



## Board support

OpenMBMC has been tested on many FPGA boards; reference designs are available in the github repository. Initial IP core testing was done with CR00107 AMD Spartan-7 based board using the on-board HyperRAM device.

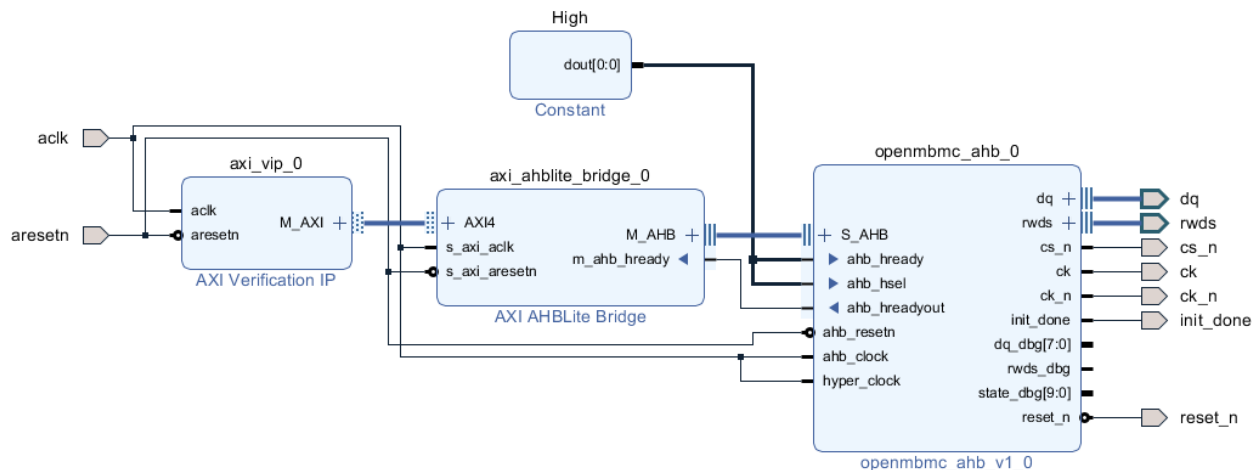
### AXE5000

This was the second Altera board the IP core was tested on, it worked on first try with the simple rtl test design. With the NIOSV design there was a problem that was solved by removing readdatavalid from the IP core.

To configure the IP core for AXE5000 on-board HyperRAM set the Address bits to 24 to support 16Mbyte.

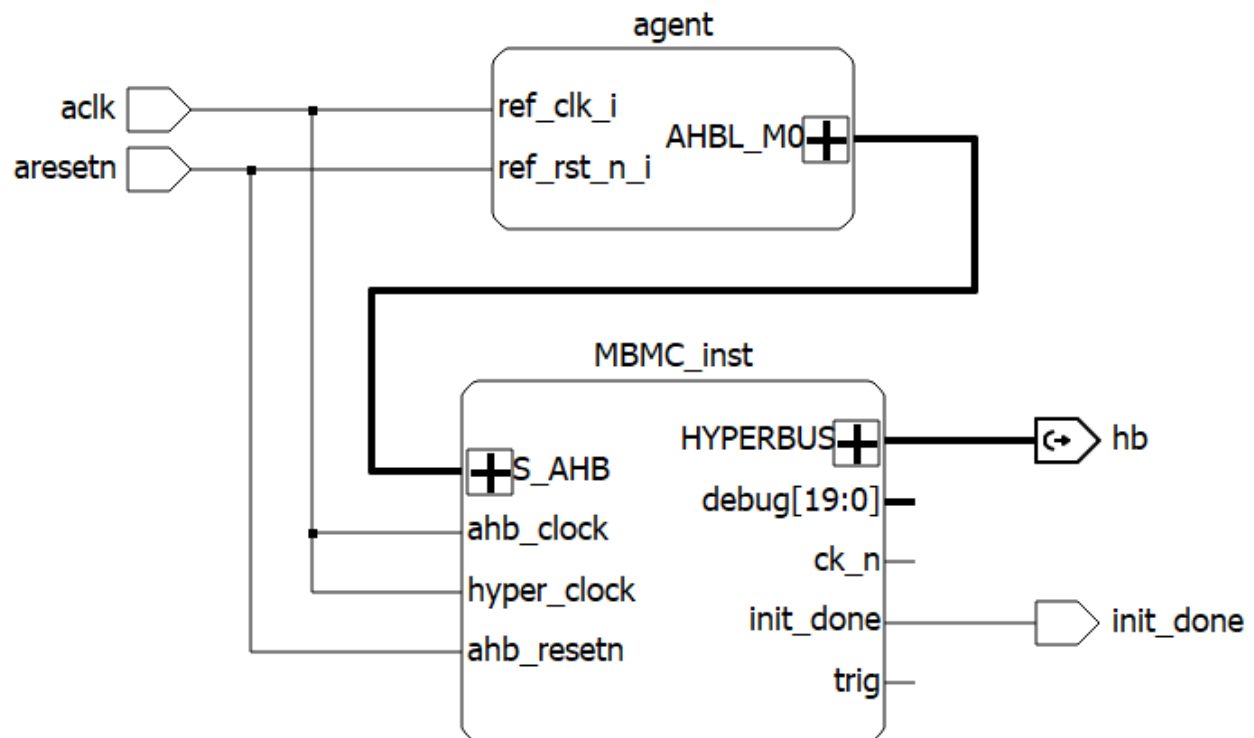


Now the next step is to combine the modules for single AHBLite component. The interesting thing was that the AHBLite slave interface requires HSEL and HREADYOUT signals but the AXI AHBLite bridge does not have those signals! In addition, the design did not work in simulation! A look of generated files revealed that bridge hready is tied to 0 and openmbmc hreadyout is not used at all. So sure the simulation could not work!



This is how it looks when it started to work. As you can see there are manual connections related to the AHBLite bus, without those the design would not work. Therefore, it is a BUG in Vivado IPI/IP cores.

After simulation of the “emulated” Propel IP core in Vivado, the IP core was packaged for Propel and test design was created using AHB BFM simulation model. This is how it looks in Propel Builder:



Unfortunately, the simulation was not possible as wrong file was generated for the AHB BFM! A workaround for this issue is manual patching of the generated toplevel verilog file, the AHB BFM instance name has to be changed to *"AHBL\_Lite\_Master\_IF"*. After this fix the simulation can be started. We have pending support ticket on this issue at Lattice Technical Support Center. Lattice has confirmed the issue and promising a fix in 25.1 release of the tools.

Unfortunately there is another problem with the AHB BFM, namely it fetches hrddata too early, at the beginning of the data phase not at the end of it. So if you simulate OpenMBMC with Questasim you will always get 0 as read data, this is problem of the AHB BFM not OpenMBMC. We have pending support ticket on this issue at Lattice Technical Support Center.



There is another issue (not a bug, but missing feature): AHB BFM only supports 32 bit read and write transactions! OpenMBMC requires 16 bit writes to work with HyperFlash. Therefore, simulation with AHB BFM should be limited to HyperRAM/OPI devices.

## Support

Support for this IP core is provided by the means of [github issues](#). If you cannot create a new github issue, you can also send email to <mailto:antti.lukats@gmail.com> , please provide as much information as you can about your issue or request.

Shareware payment for this IP Core suite is possible via paypal link: <https://paypal.me/MicroFPGA>

Or you can scan the QR code:

	
Single user fee 30 EUR, paypal <a href="#">link</a>	Company/University fee 150 EUR, <a href="#">link</a>

Recommended fee for single user is 30 USD/EUR, for company/university 150 USD/EUR.

For commercial invoice please send email with your data to [antti.lukats@gmail.com](mailto:antti.lukats@gmail.com), you will receive an invoice from MicroFPGA UGh.

Sponsoring the development is also possible via Github sponsors.