CoreVectorBlox Handbook

Introduction (Ask a Question)

CoreVectorBlox provides a flexible neural network accelerator. It uses an overlay approach, where one instantiation can run different networks without needing to be resynthesized.

This handbook provides details about Microchip CoreVectorBlox environment and how to use it. This document is used by Microchip FPGA designers using Libero® System-on-Chip (SoC).

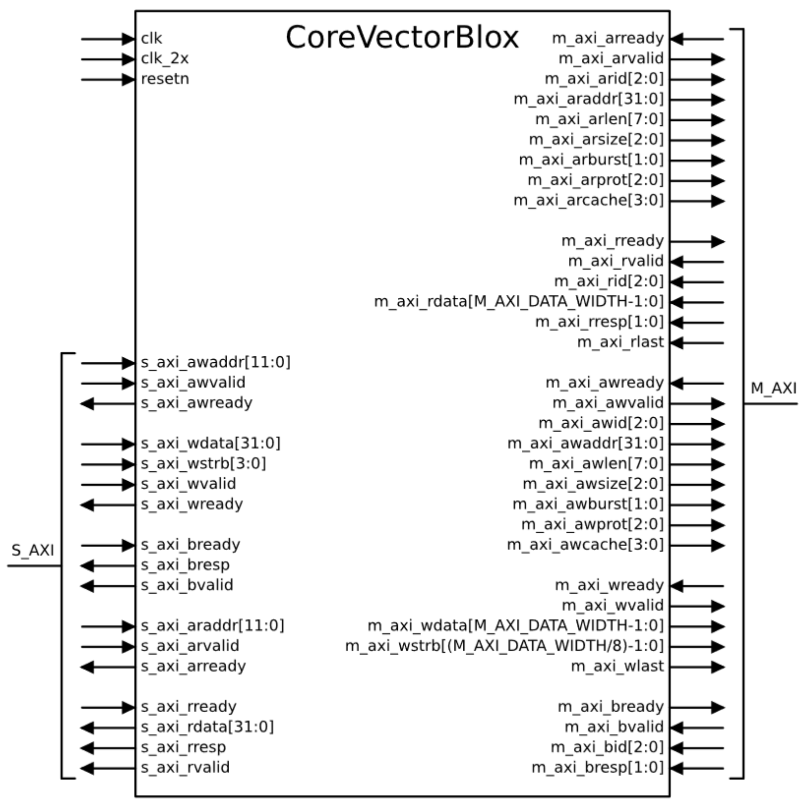
Table of Contents

1. Overview (Ask a Question)

CoreVectorBlox provides a flexible neural network accelerator. It uses an overlay approach, where one instantiation can run different networks without needing to be resynthesized. A software toolkit called VectorBlox Accelerator Software Development Kit (SDK) compiles a neural network description from a supported framework (for example, TensorFlow, Caffe and so on) into a Binary Large Object (BLOB) that is loaded into memory accessible by the CoreVectorBlox memory-mapped master. The CoreVectorBlox reads this BLOB and the network inputs from memory, processes the network and places the result into an output buffer in memory. It can switch between multiple networks dynamically because of its overlay design. The overlay features a vector processor that can handle general vector layers and a convolutional accelerator, which further accelerates Convolutional Neural Networks (CNNs). CoreVectorBlox efficiently supports most convolutional neural networks available today, such as ResNet, MobileNet, YOLO and many more. Different configurations are available, allowing the user to scale the resource utilization to the required network performance.

The following figure shows the top-level interface.

Figure 1-1. CoreVectorBlox I/O Signal Diagram



1.1 Features (Ask a Question)

The following are the key features of CoreVectorBlox.

- Multiple size configuration to trade-off performance for resource utilization.
- Overlay design, which allows multiple networks to run on the same core and even switch dynamically.
- Configurable width (64-bit to 256-bit) AXI4 memory master for data access.

- AXI4-Lite slave for control and status.
- Memory-based; reads inputs from and writes outputs to memory-mapped master.
- Internal vector processor, which can process general neural-network layers.
- CNN accelerator for convolutional layers.

1.2 Core Versions (Ask a Question)

This handbook applies to CoreVectorBlox v2.0. The release notes provided with the core list known discrepancies between this handbook and the core release associated with the release notes.

1.3 Supported Families (Ask a Question)

CoreVectorBlox supports the following devices:

- PolarFire®
- PolarFire® SoC

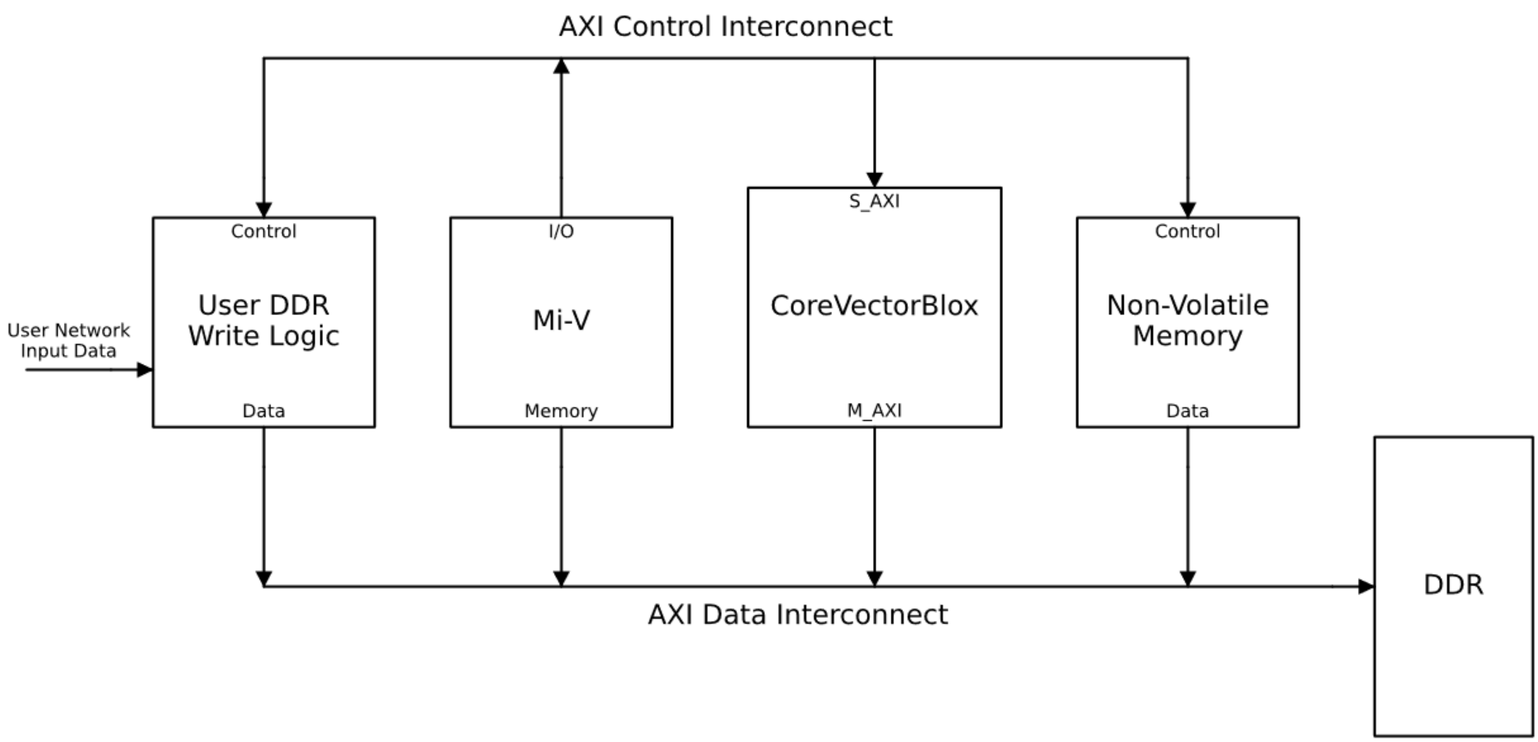
2. Functional Description (Ask a Question)

The following section shows the functional description of CoreVectorBlox.

2.1 System Level Overview (Ask a Question)

CoreVectorBlox processes neural networks residing in external memory. Its interfaces are an AXI4-Lite slave for setup and control and an AXI4 master for instruction and data memory. The following figure shows an example system of CoreVectorBlox usage.

Figure 2-1. Example of System Level Block Diagram



The Mi-V soft processor is used to control the flow of data among components. At boot up, network BLOBs (the network BLOBs are processed by the SDK and stored in non-volatile memory) are copied from non-volatile memory to DDR memory. Incoming data (for example, video frames from an image sensor) is written into DDR memory by user logic under the control of the Mi-V. When new data is available, the Mi-V instructs CoreVectorBlox to begin processing. CoreVectorBlox reads the weights and layer types from the network BLOB and data from the network inputs from DDR memory, processes the network and then writes the results to DDR memory. Finally, the Mi-V either directly reads the results or signals another module to consume them.

2.2 Memory Components (Ask a Question)

CoreVectorBlox requires the following components to be placed in memory accessible to its AXI4 master interface:

- Network BLOB(s)—BLOBs produced by the VectorBlox Accelerator SDK for each network to run.
- Network Inputs/Outputs—Network specific I/O for each run of a network.

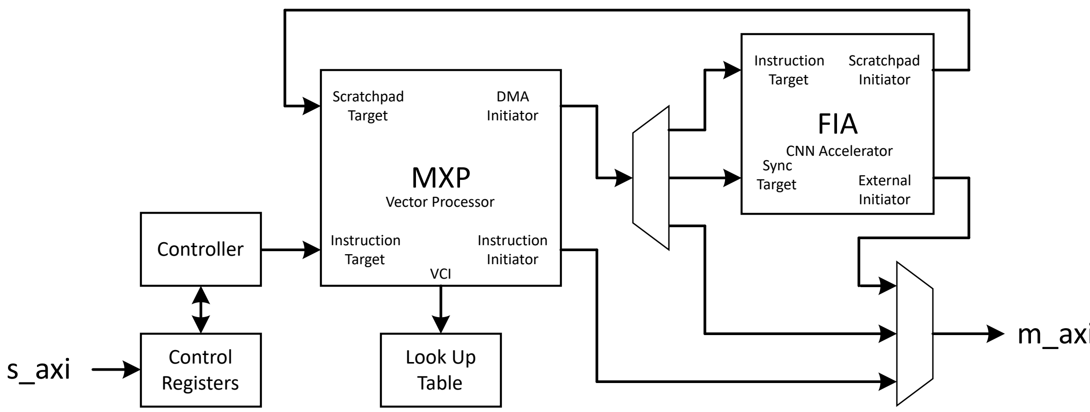
The network BLOBs are compiled by the user to target their own networks. Microchip provides examples and tutorials in the VectorBlox Accelerator SDK. A network BLOB contains information about the specific layer types in the network as well as weights and buffer space for activations.

Each network has its own BLOB; multiple BLOBs can be present in memory at the same time and each time the CoreVectorBlox is started, it can target a different BLOB. The user loads the network BLOB(s) into external memory during bootup. The network inputs and outputs are specific to each network. For more information, see the *VectorBlox Accelerator SDK*.

2.3 Hardware Architecture (Ask a Question)

The following figure shows the primary blocks of CoreVectorBlox.

Figure 2-2. CoreVectorBlox Block Diagram



The following list describes the primary blocks of CoreVectorBlox:

- Control Registers—Control, status and error registers as well as addresses for BLOBs and I/O.
- Controller—Provides control instructions to the vector processor.
- MXP Vector Processor—Processes general neural network layers.
- FIA (CNN) Accelerator—Processes convolutional network layers.

The control registers have an AXI4-Lite slave interface for external logic (for example, a Mi-V soft processor) to control the state of CoreVectorBlox. They detect and report simple errors, such as invalid BLOB addresses, but mostly they are used to initialize and communicate with the controller. Once initialized, the controller sets status and error conditions in the control registers.

The controller issues execution instructions to the MXP and sets status and error registers based on the MXP operation.

The MXP Vector Processor is a soft vector processor, which performs data-parallel operations on long vectors of data. It uses an internal scratchpad as working memory and a variable-width DMA controller for transferring data between the scratchpad and external memory. The CoreVectorBlox size configuration (see Configuration Options) configures the number of parallel ALUs and scratchpad memory size of the MXP.

The CNN Accelerator is an array of Processing Elements (PEs) that consist of a multiply-accumulate unit and small accumulation RAM. Data are retrieved from external memory and then written to the MXP scratchpad or back to external memory. The PEs are laid out in a 2D grid, which takes advantage of the multiple levels of parallelism in many neural network layers. This achieves a higher level of performance than the MXP alone, especially in convolutional neural networks and achieves a higher degree of parallelism than the MXP, when processing the convolutional layers.

2.4 Configuration Options (Ask a Question)

CoreVectorBlox can be configured to one of the various sizes listed in the following table. Detailed resource usage, device utilization and benchmark along with power and memory bandwidth usage are available in the CoreVectorBlox SDK.

Table 2-1. Processor Size Configuration Details

Size Configuration	Vector Processor Width	Vector Scratchpad	CNN Accelerator Array Size
V250	64-bit	64 kB	16 × 8
V500	128-bit	128 kB	16 × 16
V1000	256-bit	256 kB	32 × 16

3. Operation (Ask a Question)

The following section shows the operation of CoreVectorBlox.

3.1 Memory Map (Ask a Question)

The following table lists the control slave memory map. All registers are 32-bit in width. Register access is specified as a combination of readable (R), writable (W), write-to-set (WS) and write-to-clear (WC). Write-to-set (WS) bits are set to ‘1’ by a write to the specified register that has a ‘1’ in the WS bit, but can only be cleared by an internal logic (a write to the specified register with a ‘0’ in the WS bit has no effect). Write-to-clear (WC) bits are cleared to ‘0’ by a write to the specified register with a ‘1’ in the WC bit (a write to the specified register with a ‘0’ in the WC bit has no effect). Write-to-clear (WC) registers are cleared by any write to the specified register.

Table 3-1. Control Slave Memory Map

Address	Name	Description			Access
		Bit 0 = LSB	Function	Reset Value	
0x00	Control Register	0: Soft Reset	Write 1 to soft reset. All other control register bits are ignored when soft reset is activated. The error register is cleared on soft reset as well. Soft reset expects to fulfill memory transactions on all interfaces. If the modules connected to `S_AXI` or `M_AXI` are placed into reset and may not have correctly fulfilled outstanding AXI transactions, then a hard reset (through the `resetn` pin) must be performed to ensure that the `S_AXI` and `M_AXI` interfaces come up correctly.	1	R/W
		1: Start	Write 1 to set. Setting this bit causes the CoreVectorBlox to start processing a network. It begins fetching the network BLOB from the location specified in the Network Model Address register and decode the BLOB to determine how to proceed. The Network Model Address and Network I/O Address registers must be set before setting this bit. Once cleared, the hardware starts processing the current network (concurrently with setting the Running bit). While set, it is an error to Start again or to write to the Network Model Address or Network I/O Address registers.	0	R/WS
		2: Running	Set during processing (concurrently with clearing the Start bit). Cleared concurrently to setting the 'Output Valid' bit.	0	R

...........continued

Address	Name	Description			Access
		Bit 0 = LSB	Function	Reset Value	
0x00	Control Register	3 Output Valid	Write 1 to clear. Set once the network outputs are valid. It must be cleared once per network invocation. An invocation is started by writing the Start bit and is ended by clearing the Output Valid bit. The Output Valid bit is not cleared before writing the Start bit for the next network invocation, but until it is cleared, the subsequent network does not finish (the Running bit stays set). The Output Valid bit sets once and must be cleared once per setting of the Start bit. If the control slave does not clear this bit, subsequent network invocations will not finish. While this bit is clear, it is an error to write to this bit.	0	R/WC
0x00	Control Register	4: Error	Indicates the contents of the Error Register are valid and must be examined.	0	R
		31:5	Reserved	0	R
0x04	Error Register	Write any value to clear to zero, which also clears the Error bit of the Control Register. Errors will cause a soft reset, which is identical to setting the Soft Reset bit of the Control Register except that the Error bit and this register are not cleared. See Error Codes for more information.			R/WC
0x08	Reserved	—			—
0x10	Network Model Address	Address Pointing to Model BLOB. Must be aligned to an 8 byte boundary and be greater than or equal to 0x0020_0000 when setting the 'Start' bit or an error will be raised. Writing while the Start bit is set raises an error.			R/W
0x18	Network I/O Address	Address pointing to the I/O data structure for the network. Must be aligned to an 8 byte boundary and be greater than or equal to 0x0020_0000 when setting the 'Start' bit or an error will be raised. Writing while the Start bit is set raises an error.			R/W
0x28	Version	See the following table for version information.			R

Table 3-2. Version

Bits	Name	Function
7:0	Product ID	Reserved; Reads 0 for CoreVectorBlox.
15:8	Size Configuration	Size configuration: 0≥V250, 1≥V500 , 2≥V1000
19:16	Reserved	Reserved core version information
27:20	Minor Version	Minor version number (For example, 0x05 for CoreVectorBlox 1.5).
31:28	Major Version	Major version number (For example, 0x01 for CoreVectorBlox 1.5).

Table 3-3. Error Codes

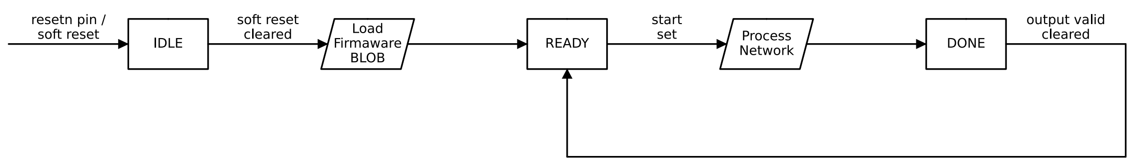
Value	Code	Description
1	INVALID_INSTRUCTION_ADDRESS	The instruction address is within the reserved memory range (first 2 MB).

Value	Code	Description
...........continued		
2	START_NOT_CLEAR	The Start bit of the Control Register, the Network Model Address, or the Network I/O Address were written while the Start bit of the Control Register was still set.
3	OUTPUT_VALID_NOT_SET	The Output Valid bit of the Control register was cleared when it was not set.
4	NETWORK_BLOB_INVALID	The network BLOB has an invalid format.
5	NETWORK_BLOB_VERSION_MISMATCH	The network BLOB was built for the wrong version of the instruction BLOB.
6	NETWORK_BLOB_PRESET_MISMATCH	The network BLOB was built for a different preset configuration than the one in use.
7	INSTRUCTION_BLOB_STALE	The instruction BLOB was not reloaded between resets of the accelerator.

3.2 Network Processing (Ask a Question)

The following figure shows the flow of processing networks.

Figure 3-1. Network Processing Flow



The CoreVectorBlox SDK provides a C API.

When coming out of reset, CoreVectorBlox is held in an IDLE state.

Processing a network is started by setting the network BLOB address and then setting the ‘start’ bit of the control register. CoreVectorBlox will then begin parsing the network BLOB and read in the inputs as it starts processing. Exact steps of processing depend on the network BLOB. During processing, the memory master may also access temporary buffer memory. The temporary buffers are pre-allocated inside the network BLOB; CoreVectorBlox only accesses memory inside the network BLOB and network input and output buffers.

When network processing completes, the ‘Output Valid’ bit of the control register is set and the network outputs can be read. The next network run might start as soon as the current run is finished.

At any point, CoreVectorBlox can be put back into the Soft Reset state by setting the ‘soft reset’ bit of the control register. Soft reset attempts to finish any outstanding AXI transactions on the master and slave interfaces, as opposed to a hard reset using the `resetn` pin, which will immediately cease all transactions and reset those interfaces. For more details on signals, see Memory Map.

4. Generics (Ask a Question)

Customers can define the generics listed in the following table as needed in the source code.

Table 4-1. CoreVectorBlox Generics

Generic	Default Setting	Valid Values	Description
Size Configuration	V1000	[V250, V500, V1000]	Size Configuration; see Processor Size Configuration Details table.
M_AXI_DATA_WIDTH	256	[64, 128, 256]	AXI4 Data Master data width in bits.

5. Interface Description (Ask a Question)

The port signals for CoreVectorBlox are defined in the following tables and are also described in I/O Signal Diagram.

5.1 Clocks and Resets (Ask a Question)

The following table lists the clocks and the resets.

Table 5-1. Clocks and Resets

Signal	Function	I/O	Description
clk	Clock	Input	System clock. The control slave and data master are synchronous to this clock.
clk_2x	2X Clock	Input	Double-frequency clock. Clocks MathBlocks and LSRAM resources at twice the frequency of the system clock. It must be synchronous to and in phase with clk. See the following notes.
resetn	Reset	Input	System reset (active low). Resets all core functions as well as the control slave and data master.

Note:

1. To minimize skew between clk and clk_2x, the clocks must be created from the same CCC on PolarFire devices. Additionally, the clock outputs are paired. Either the OUT0 and OUT1 or the OUT2 and OUT3 pair must be used, but not one output from each pair.
2. The Libero Place and Route tool must be configured with the 'Repair Minimum Delay Violations' option selected to repair any hold violations caused by skew between the two clocks.

5.2 Control Slave Signals (Ask a Question)

The Control Slave is an AXI4-Lite compliant interface with memory map described in Memory Map. It is synchronous to clk and reset by `resetn` pin. The following table lists the signals.

Table 5-2. Control Slave Signals

Signal	Function	I/O	Description
s_axi_awaddr	Write Address	Input	AXI4-Lite slave write address.
s_axi_awvalid	Write Address Valid	Input	AXI4-Lite slave write address valid.
s_axi_awready	Write Address Ready	Output	AXI4-Lite slave write address ready.
s_axi_wdata	Write Data	Input	AXI4-Lite slave write data.
s_axi_wstrb	Write Data Strobe	Input	AXI4-Lite slave write data strobe (byte enable). CoreVectorBlox expects all write strobe signals to be all high or all low; partial register writes results in undefined results.
s_axi_wvalid	Write Data Valid	Input	AXI4-Lite slave write data valid.
s_axi_wready	Write Data Ready	Output	AXI4-Lite slave write data ready.
s_axi_bready	Write Response Ready	Input	AXI4-Lite slave write response ready.
s_axi_bresp	Write Response	Output	AXI4-Lite slave write response code.
s_axi_bvalid	Write Response Valid	Output	AXI4-Lite slave write response valid.
s_axi_araddr	Read Address	Input	AXI4-Lite slave read address.
s_axi_arvalid	Read Address Valid	Input	AXI4-Lite slave read address valid.
s_axi_arready	Read Address Ready	Output	AXI4-Lite slave read address ready.
s_axi_rready	Read Data Ready	Input	AXI4-Lite slave read data ready.

...........continued			
Signal	Function	I/O	Description
s_axi_rdata	Read Data	Output	AXI4-Lite slave read data.
s_axi_rresp	Read Data Response	Output	AXI4-Lite slave read data response code.
s_axi_rvalid	Read Data Valid	Output	AXI4-Lite slave read data valid.

Note:

1. All control slave signals are synchronous to clk.
2. The control slave is reset by `resetn` pin.

5.3 Data Master Signals (Ask a Question)

The Data Master is an AXI4 compliant interface. It is synchronous to clk and reset by `resetn` pin. The following table lists the signals.

Table 5-3. Data Master Signals

Signal	Function	I/O	Description
m_axi_arready	Read Address Ready	Input	AXI4 master read address ready.
m_axi_arvalid	Read Address Valid	Output	AXI4 master read address valid.
m_axi_arid	Read Address ID	Output	AXI4 master read address ID. CoreVectorBlox uses multiple IDs; these must be propagated correctly through any interconnect attached to the Data Master.
m_axi_araddr	Read Address	Output	AXI4 master read address.
m_axi_arlen	Read Length	Output	AXI4 master read length (beats per burst minus 1).
m_axi_arsize	Read Size	Output	AXI4 master read size. CoreVectorBlox does not issue narrow reads; this will be fixed to the data width size.
m_axi_arburst	Read Burst Type	Output	AXI4 master read burst type. CoreVectorBlox only issues incrementing bursts.
m_axi_arprot	Read Protection	Output	AXI4 master read protection. CoreVectorBlox only issues unprivileged, secure data acceses.
m_axi_arcache	Read Transaction Attributes	Output	AXI4 master read transaction attributes. CoreVectorBlox issues modifiable and bufferable transactions. It does not set allocation bits and assumes that transactions can be read from memory in systems with caches.
m_axi_rready	Read Data Ready	Output	AXI4 master read data ready.
m_axi_rvalid	Read Data Valid	Input	AXI4 master read data.
m_axi_rid	Read Data ID	Input	AXI4 master read data ID. CoreVectorBlox uses multiple IDs.
m_axi_rdata	Read Data	Input	AXI4 master read data.
m_axi_rresp	Read Data Response	Input	AXI4 master read data response code.
m_axi_rlast	Read Data Last	Input	AXI4 master read data last (end of burst).

...........continued

Signal	Function	I/O	Description
m_axi_awready	Write Address Ready	Input	AXI4 master write address ready.
m_axi_awvalid	Write Address Valid	Output	AXI4 master write address valid.
m_axi_awid	Write Address ID	Output	AXI4 master write address ID. CoreVectorBlox uses multiple IDs; these must be propagated correctly through any interconnect attached to the Data Master.
m_axi_awaddr	Write Address	Output	AXI4 master write address.
m_axi_awlen	Write Length	Output	AXI4 master write length (beats per burst minus 1).
m_axi_awsize	Write Size	Output	AXI4 master write size. CoreVectorBlox does not issue narrow writes; this will be fixed to the data width size.
m_axi_awburst	Write Burst Type	Output	AXI4 master write burst type. CoreVectorBlox only issues incrementing bursts.
m_axi_awprot	Write Protection	Output	AXI4 master write protection. CoreVectorBlox only issues unprivileged, secure data acceses.
m_axi_awcache	Write Transaction Attributes	Output	AXI4 master write transaction attributes. CoreVectorBlox issues modifiable, bufferable transactions. It does not set allocation bits.
m_axi_wready	Write Data Ready	Input	AXI4 master write data ready.
m_axi_wvalid	Write Data Valid	Output	AXI4 master write data valid.
m_axi_wdata	Write Data	Output	AXI4 master write data.
m_axi_wstrb	Write Data Strobe	Output	AXI4 master write data strobe (byte enable).
m_axi_wlast	Write Data Last	Output	AXI4 master write last (end of burst).
m_axi_bready	Write Response Ready	Output	AXI4 master write response ready.
m_axi_bvalid	Write Response Valid	Input	AXI4 master write response valid.
m_axi_bid	Write Response ID	Input	AXI4 master write response ID.
m_axi_bresp	Write Response Code	Input	AXI4 master write response code.

Note:

1. All data master signals are synchronous to clk.
2. The data master is reset by `resetn` pin.

5.4 Interrupt Signals (Ask a Question)

The following table lists the interrupt signals.

Table 5-4. Interrupt Signals

Signal	Function	I/O	Description
output_valid	Interrupt Output	Output	Mirrors the Output Valid bit in the Control Register (see Memory Map) for use as an interrupt for needed systems.

6. Tool Flows (Ask a Question)

The following section shows the tool flows of CoreVectorBlox.

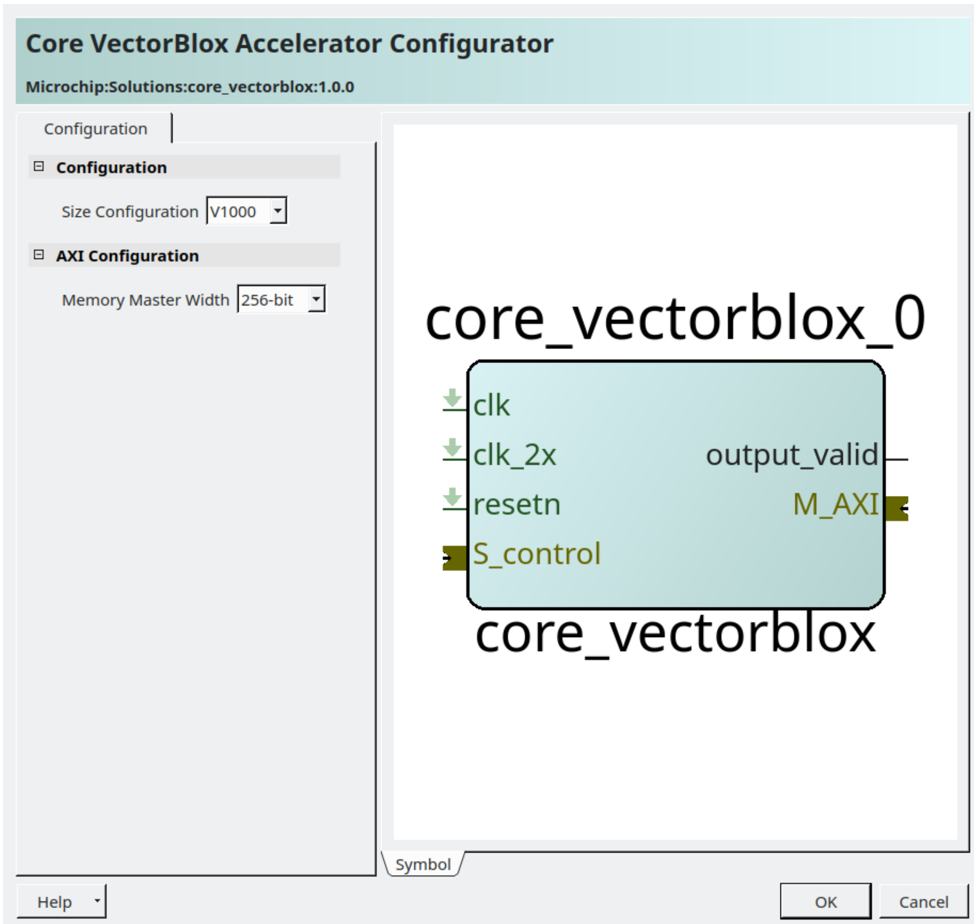
6.1 Licenses (Ask a Question)

CoreVectorBlox is licensed using encrypted RTL.

6.2 Smart Design (Ask a Question)

The following figure shows the core configured using the configuration GUI within SmartDesign.

Figure 6-1. CoreVectorBlox Configurator within SmartDesign with V1000 Configuration



6.3 Simulation (Ask a Question)

CoreVectorBlox can be functionally simulated as well as simulated at the RTL level. For functional simulation, see the SDK. Functional simulation is the preferred way of verifying network functionality and accuracy.

The RTL can be simulated as part of a higher level design. The user has to load the model BLOBs into memory attached to the Data Master port. See the documentation for the specific memory interface used. The user must also write to the control registers, either using state machine logic or by instantiating a control processor, such as a Mi-V and loads a program for it to set the control registers.

6.4 Synthesis (Ask a Question)

Set the design root appropriately and click the Synthesis icon in Libero. To perform synthesis, right-click and select Run. CoreVectorBlox requires no special synthesis settings.

6.5 Place and Route (Ask a Question)

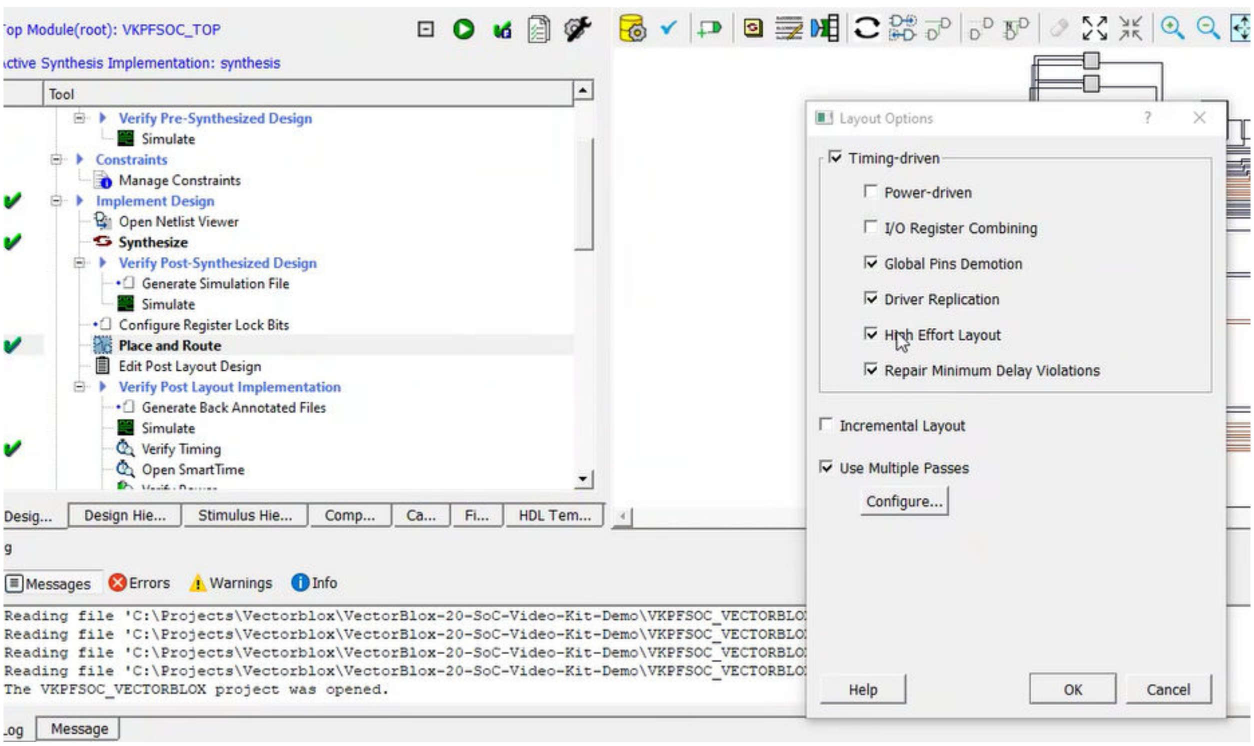
The “Repair Minimum Delay Violations” option must be turned on in the Libero **Place and Route** tool to fix any hold violations between clk and clk_2x caused by clock skew through the fabric (see Clocks and Resets).

Set the design route appropriately and run Synthesis. Click the **Place and Route** icon in Libero to invoke the Designer software.

For the best results, make the following settings in the **Place and Route** options in Libero.

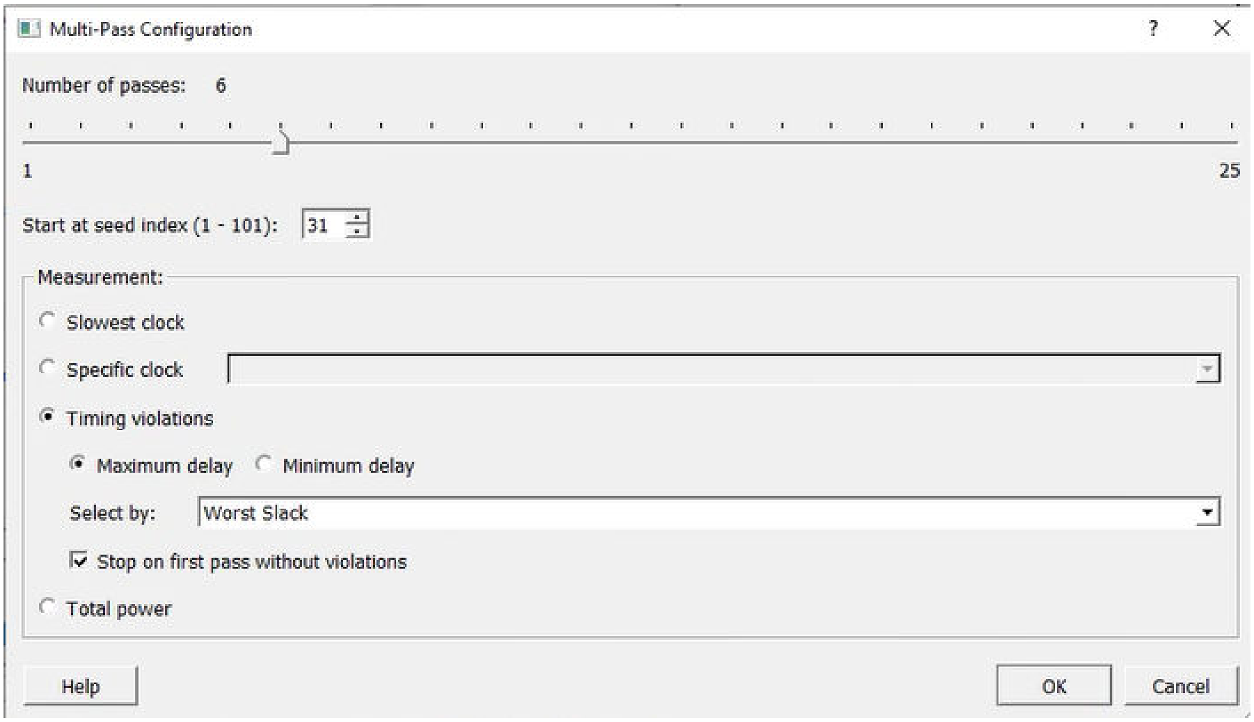
The following figure shows the Place and Route settings.

Figure 6-2. Place and Route Settings



The following figure shows the Multi-Pass configuration settings.

Figure 6-3. Multi-Pass Configuration Settings



7. Revision History (Ask a Question)

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the current publication.

Revision	Date	Description
B	11/2024	The following is the list of changes in revision B: • Updated "CoreVectorBlox v2.0" from "CoreVectorBlox v1.1" in Core Versions section. • Updated Supported Families section. • Removed "1.4 Device Utilization and Performance" section. • Updated Memory Components section. • "Controller" is added and updated Hardware Architecture section. • Updated Table 2-1 in Configuration Options section. • Updated Table 3-1 and Table 3-3 in Memory Map section. • Updated Network Processingsection. • Updated Table 4-1 in Generics section. • Updated Simulation section. • Added Figure 6-2 and Figure 6-3 in Place and Route sections.
A	01/2021	The following is the list of changes in revision A. • The document was updated to Microchip template and document number was changed from 50200919 to DS50003112A.
2.0	11/2020	The following is the list of changes in revision 2.0. • The Overview section was updated. • "Device Utilization and Performance" table was updated. • Added Interrupt Signals section. • Updated Figure 6-1.
1.0	06/2020	Revision 1.0 is the first publication of this document.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

- From North America, call **800.262.1060**
- From the rest of the world, call **650.318.4460**
- Fax, from anywhere in the world, **650.318.8044**

Microchip Information

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 979-8-3371-0072-2

Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip’s intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable.” Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.