
Speedcore eFPGA Datasheet (DS012)

Speedcore eFPGA



Copyrights, Trademarks and Disclaimers

Copyright © 2021 Achronix Semiconductor Corporation. All rights reserved. Achronix, Speedcore, Speedster, and ACE are trademarks of Achronix Semiconductor Corporation in the U.S. and/or other countries. All other trademarks are the property of their respective owners. All specifications subject to change without notice.

NOTICE of DISCLAIMER: The information given in this document is believed to be accurate and reliable. However, Achronix Semiconductor Corporation does not give any representations or warranties as to the completeness or accuracy of such information and shall have no liability for the use of the information contained herein. Achronix Semiconductor Corporation reserves the right to make changes to this document and the information contained herein at any time and without notice. All Achronix trademarks, registered trademarks, disclaimers and patents are listed at <http://www.achronix.com/legal>.

Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

Website: www.achronix.com
E-mail : info@achronix.com

Table of Contents

Chapter - 1: Overview 4

 Introducing Speedcore eFPGA 4

 Feature Summary 4

 Functionality 5

 Process Technology 5

 Programming 6

Chapter - 2: Speedcore eFPGA Architecture 7

 Fabric Architecture 7

 Block Floorplan 7

 Speedcore eFPGA Clock Network 8

 Speedcore eFPGA Interface Cluster 10

 Interface Timing Closure 12

 Speedcore eFPGA Logic Fabric – Reconfigurable Logic Block 12

 Block RAM 20k 13

 Logic RAM 4k 14

 Speedcore eFPGA DSP64 Block 15

Chapter - 3: Speedcore eFPGA IP Interface 17

 Interfaces 17

 Data Signals 17

 Clock Inputs 17

 Programming Interface 17

 Pins 18

Chapter - 4: Speedcore eFPGA In-System Debug 20

 Features 20

Chapter - 5: Speedcore eFPGA Integration Flow 22

 Physical Integration with Customer ASICs 22

 Simulation and Validation 23

Revision History 24

Chapter - 1: Overview

Introducing Speedcore eFPGA

Achronix's Speedcore™ embedded FPGA (eFPGA) IP includes look-up-table, memory, and DSP building blocks. Each of these blocks are designed to be modular to allow customers to define any mix of resources required for their end system.

Achronix delivers the Speedcore IP in GDSII format along with all files and documentation required for the customer to integrate their Speedcore eFPGA instance into their ASIC. Achronix also delivers the supporting ACE design tools that are used to compile designs targeting their Speedcore eFPGA.

Feature Summary

Because Speedcore eFPGA is an embeddable IP, it is designed to be completely surrounded by the end user ASIC (see the figure below). A Speedcore eFPGA includes the following features.

- Programmable core fabric, with customer defined functionality
- Core I/O ring
- FPGA configuration unit (FCU)
- Configuration memory (CMEM)
- Interfaces for debug and programming
- Interface for test (DFT)

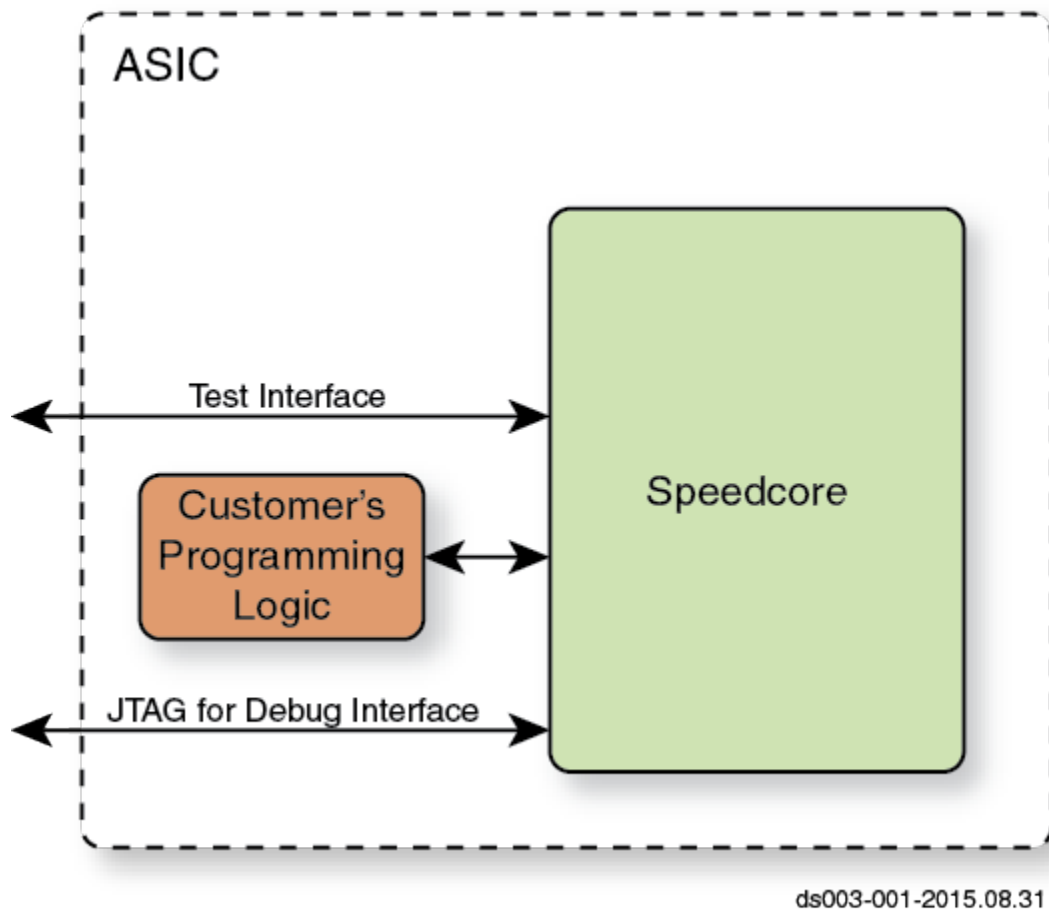


Figure 1: Embedded Speedcore

Functionality

Customers define the functionality of their Speedcore eFPGA by choosing the quantity of each of the resources listed below:

- **Logic** – 6-input look-up-tables (LUTs) plus integrated fast adders
- **Logic RAM** – up to 4kb per memory block for LRAM4k
- **Block RAM** – up to 20kb per memory block for BRAM20k
- **DSP64** – each block has a 18×27 multiplier, 64-bit accumulator and 27-bit pre-adder

Note



The number and mix of resource blocks for each Speedcore eFPGA instance is based on customer requirements.

Process Technology

Speedcore eFPGA IP is currently available on the TSMC N7 process node, the TSMC 16FFC process node, and the TSMC 12FFC process node.

Programming

Customers can select the programming interface to be one or a combination of the following available options:

- JTAG
- Parallel CPU (×1, ×8, ×16, ×32, ×128 data width modes)
- Serial flash (1 or 4 flash devices)
- AXI 128-bit

Chapter - 2: Speedcore eFPGA Architecture

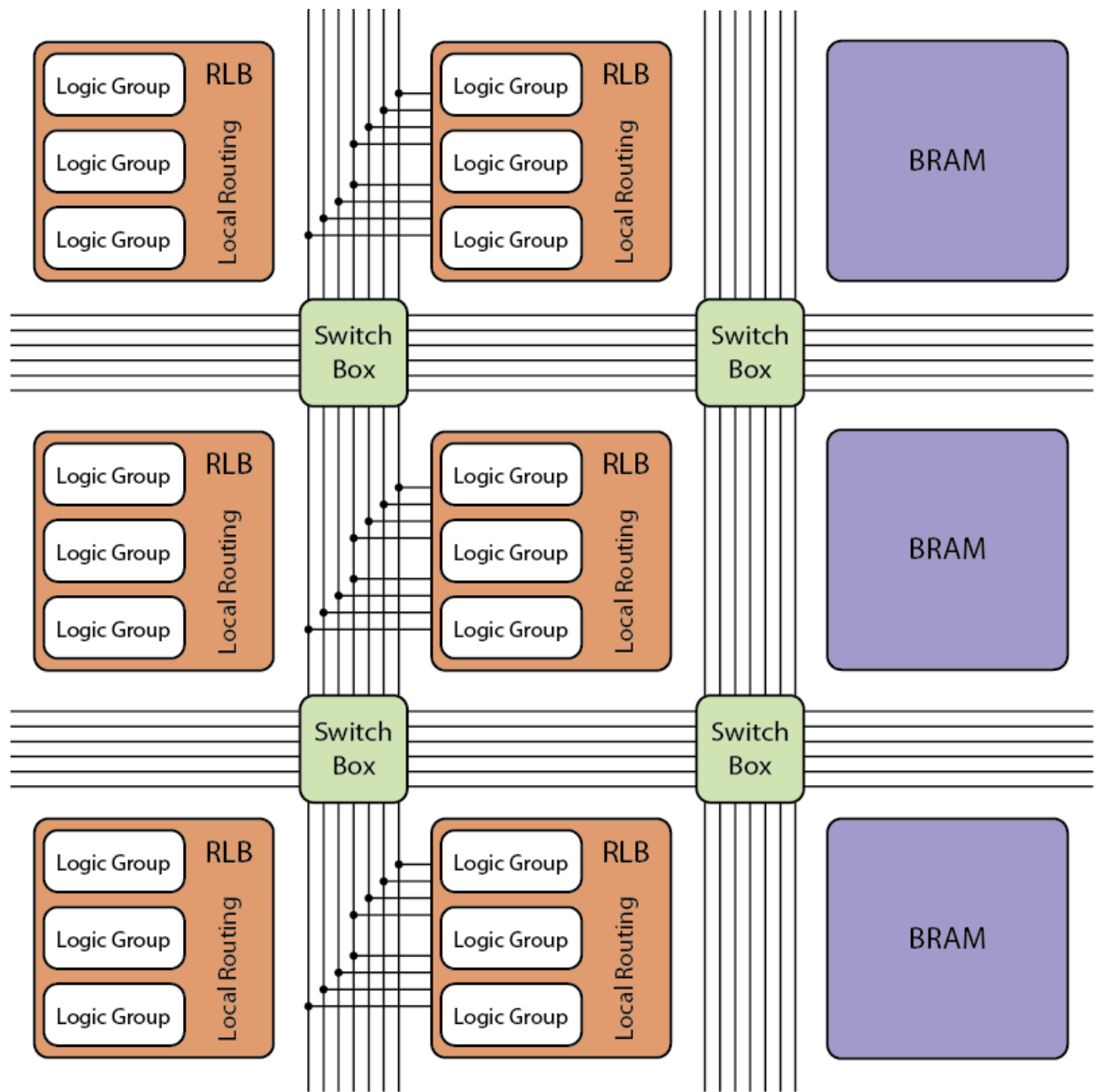
Fabric Architecture

The Speedcore eFPGA fabric is built from a library of tiles. Each tile consists of a routing switch box plus a logic block which can consist of LUTs, memory, DSPs, etc. Each type of block is designed to snap together in a grid, where abutting routing networks connect.

Various tile flavors are assembled to deliver the desired fabric size and resource mix, with all connections between tiles accomplished via abutment and guaranteed to be DRC/LVS clean for any combination of tile flavors.

Block Floorplan

The Speedcore floorplan is arranged with the various block functions arranged in columns. The block functions are connected by a uniform global interconnect, which enables the routing of signals between core elements. Switch boxes make the connection points between vertical and horizontal routing tracks. Inputs to and outputs from each of the functions connect to the global interconnect.



ds003-003.2017.01.10

Figure 2: Speedcore eFPGA Interconnect

Speedcore eFPGA Clock Network

Speedcore eFPGAs have two types of clock networks targeted to provide both a low-skew, balanced architecture as well as addressing the source-synchronous nature of data transfers with external interfaces.

The hierarchical global clock network feeds resources within the eFPGA fabric. The global clock trunk runs vertically up and down the center of the core (gray stripe in the following figure), sourced by global clock muxes at the top and bottom of the global trunk. The sources driven on the trunk are then distributed to all clock regions on both the left and right halves of the core.

Within Speedcore eFPGAs, there is a second clock network available at the periphery of core, the interface clock network. As the name implies, the intent of these clocks is to facilitate the construction of interface logic within the eFPGA core operating on the same clock domain as local logic in the surrounding host ASIC. The clocks connect to the core through the surrounding interface clusters, allowing for clock signals to be driven both into and out of the core.

These two networks are shown in the two figures below.

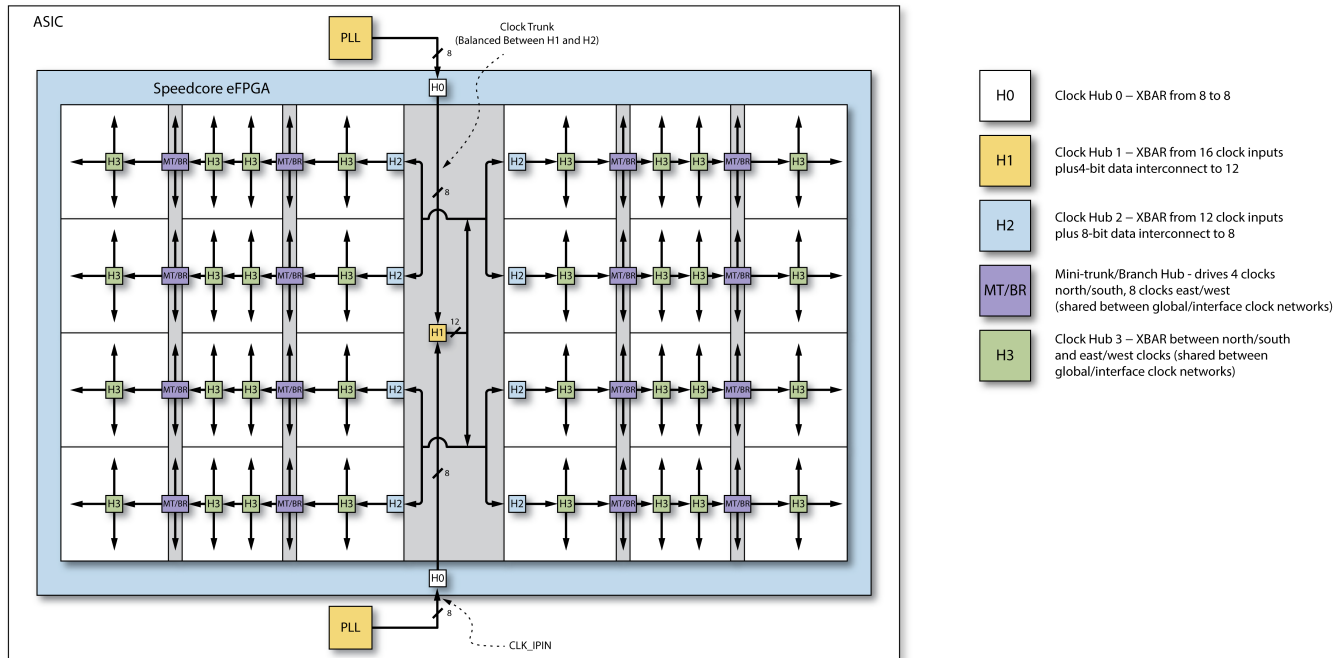


Figure 3: Global Core Clock Network

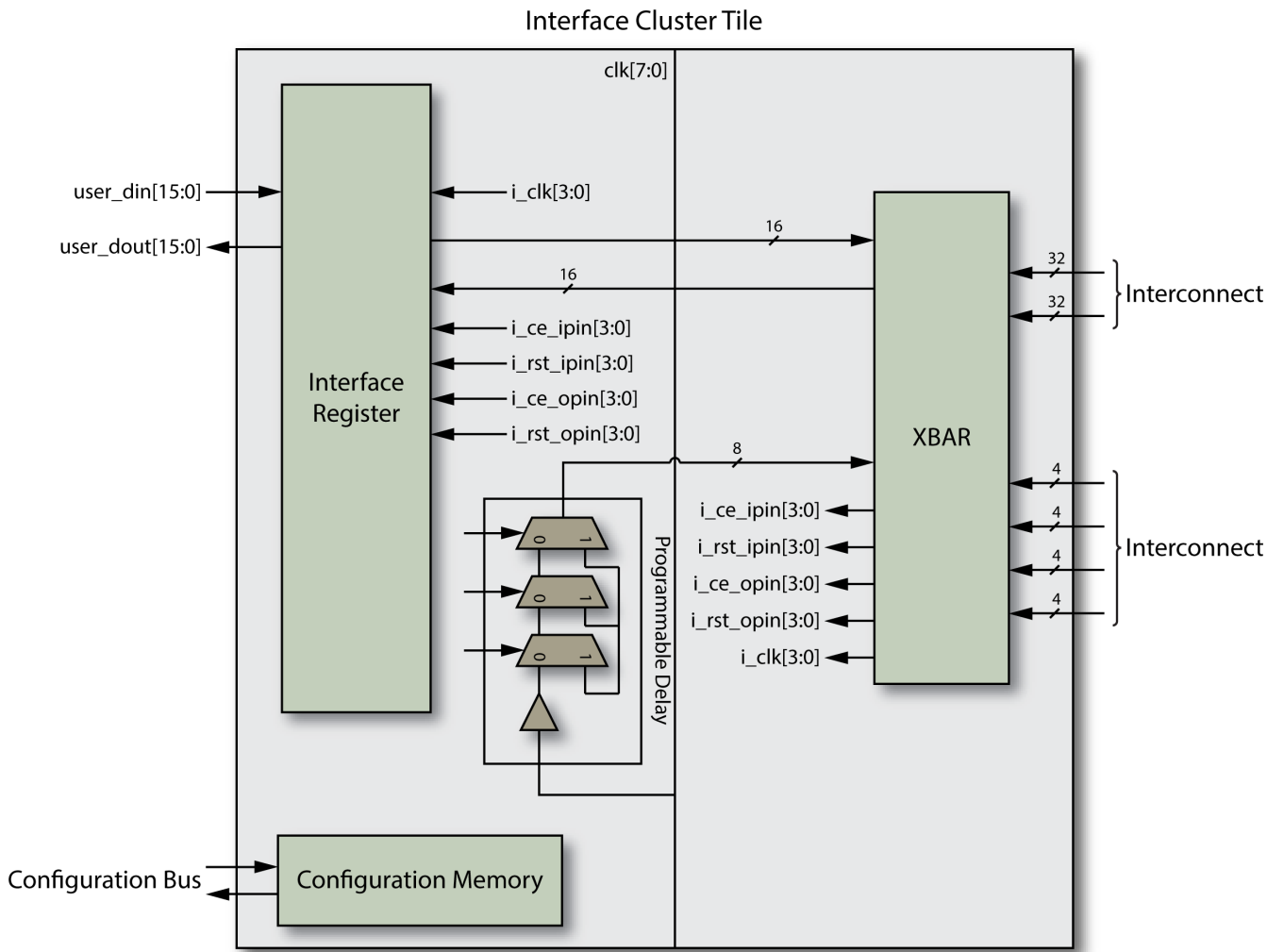
In the figure below the dots represent the clock interface clusters. At points *A* and *B* in the figure, up to four clocks can enter at the boundary of the Speedcore instance. In the clock interface cluster, these four clocks can be routed on up to eight clock signals going into the fabric. For further details, refer to the section on the [Speedcore Interface Cluster](#) (see page 10).



Figure 4: Interface Clock Network

Speedcore eFPGA Interface Cluster

The interface cluster is the portion of the Speedcore boundary ring that contains the registers, Achronix configuration bus (ACB) logic, and the connectivity to the Speedcore top-level pins. The figure below shows the details of an interface cluster, illustrating the ingress and egress path for user signals to the core (both paths can be optionally registered). The second figure depicts the clock tile, showing how clocks enter and exit the fabric, including optional delays.



34018477-01.20209.09.12

Figure 5: Interface Cluster Logic Tile

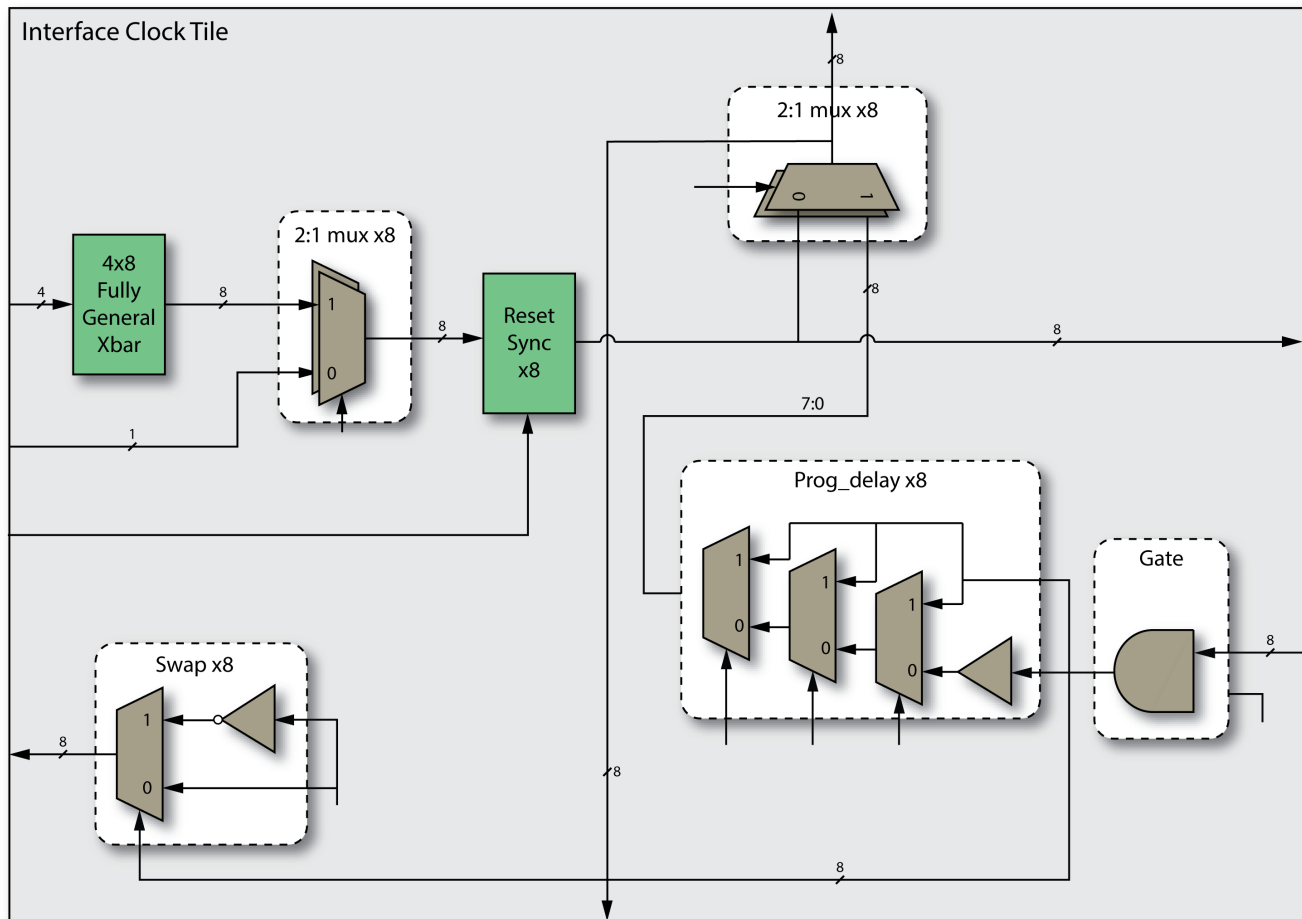


Figure 6: Interface Cluster Clock Tile

Interface Timing Closure

The Speedcore interface clusters do not have programmable I/O that connect directly to the pads on the host ASIC. Instead the Speedcore eFPGA IP supports a large number of interface clusters which connect directly to logic signals within the host ASIC. The exact number of interface I/O is dependent upon the size of the Speedcore eFPGA instance specified.

The timing of signals from the host ASIC to any embedded eFPGA is crucial in enabling signals to close timing at the high frequencies desired for 16nm or smaller technology. The variation in clock skew between the host ASIC and an embedded eFPGA's internal clock structures must be fully accounted for and carefully engineered. To enable this timing closure, Achronix has two different timing scenarios which enable customers to configure the optimum timing path architecture for different I/O groups.

Speedcore eFPGA Logic Fabric – Reconfigurable Logic Block

An RLB contains multiple 6-input look-up-tables (LUT6), a number of registers, and an 8-bit fast arithmetic logic unit (ALU8). The table below provides information on the resource counts inside an RLB in a Speedcore eFPGA.

Table 1: RLB Resource Counts

Resource	Count
LUT6	12
Registers	24
8-bit ALU	1


The following features are available using the resources in the RLB:

- 8-bit ALU for adders, counters, and comparators
- MAX function that efficiently compares two 8-bit numbers and chooses the maximum or minimum result
- 8-to-1 MUX with single-level delay
- Support for LUT chaining within the same RLB and between RLBs
- Dedicated connections for high-efficiency shift registers
- Ability to fan-out a clock enable or reset signal to multiple tiles without using general routing resources
- 6-input LUT configurable to function as two 5-input LUTs using shared inputs and two outputs
- Support for combining two 6-input LUTs with a dynamic select to provide 7-input LUT functionality

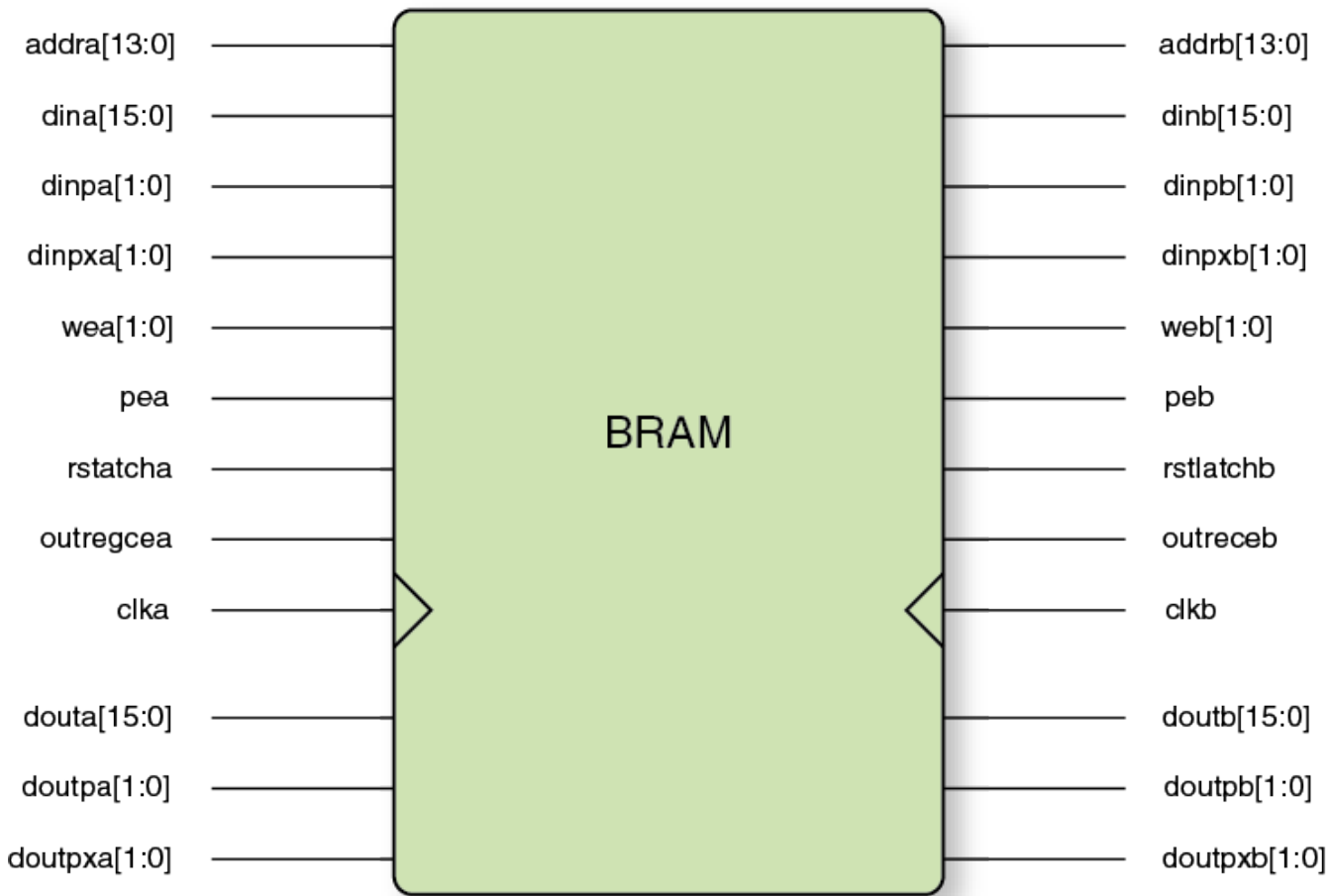
Block RAM 20k

The BRAM20k implements a dual-ported memory block where each port can be independently configured with respect to size and function. The BRAM20k can be configured as a single-port (one read/write port), dual-port (two read/write ports with independent clocks), or ROM memory. The key features of the BRAM20k are summarized in the table below.

Table 2: BRAM20k Key Features

Feature	Value
Block RAM Size	20 kb
Organization	512 × 40 ^(†) , 1k × 20, 1k × 18, 1k × 16, 2k × 10, 2k × 9, 2k × 8, 4k × 5, 4k × 4, 8k × 2, 16k × 1
Physical Implementation	Columns throughout device
Number of Ports	Dual port (independent read and write)
Port Access	Synchronous
<div>  Note † 512 × 40 only available as simple dual-port function. </div>	

The BRAM20k ports are illustrated in the following figure:



ds003-005-2016.01.28

Figure 7: BRAM20k Ports

Logic RAM 4k

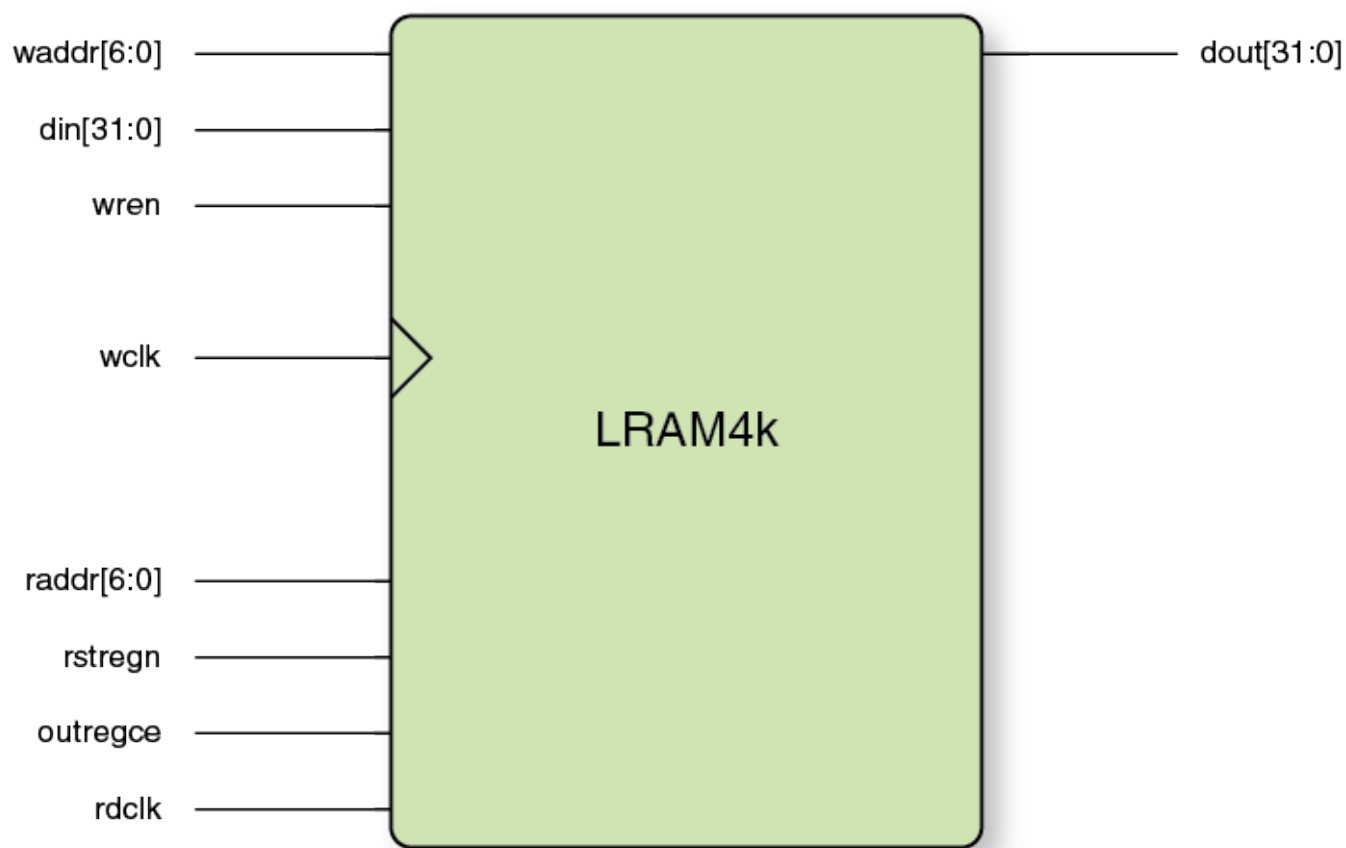
The LRAM4k implements a 4,096-bit memory block configured as a 128 × 32 simple dual-port (one write port, one read port) RAM. The LRAM4k has a synchronous write port. The read port is configured for asynchronous read operations with an optional output register. This memory block is distributed in the eFPGA fabric. A summary of LRAM4k features is shown in the table below.

Table 3: LRAM4k Key Features

Feature	Value
Logic RAM size	4,096 bits
Organization	128 × 32
Physical implementation	Dedicated columns

Feature	Value
Number of ports	Simple dual port (one read, one write)
Port access	Synchronous writes, asynchronous reads

The LRAM4k ports are shown in the following figure:



ds003-006-2019.02.06

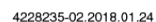
Figure 8: LRAM4k Ports

Speedcore eFPGA DSP64 Block

The DSP64 blocks include multiple/accumulate and associated logic to efficiently implement math functions such as finite impulse response (FIR) filters, fast Fourier transforms (FFT), and infinite impulse response (IIR) filters. The DSP64 blocks are optimized to operate with the logic fabric and LRAM blocks to implement math functions. Refer to the *Speedcore IP Component Library User Guide* (UG065) for more details.

The DSP64 blocks have the following functions:

- 27-bit preadder
- 18×27 multiplication/accumulation with programmable load value
- Add/subtract



Chapter - 3: Speedcore eFPGA IP Interface

Interfaces

There are three sets of interfaces to a Speedcore instance, data, clock and programming interface (see the figure below).

Data Signals

Data signals (inputs and outputs) can be on all four sides or only on two opposite sides. At the boundary, there is an option to either register the signals or send the signals directly to the programmable logic core.

Clock Inputs

Clock inputs follow the same pattern as data. These can be on all four sides or on two opposite sides. There are four interface clocks per cluster on the north and south sides, and four interface clocks per cluster on the east and west sides.

Programming Interface

There is a dedicated set of signals for programming of eFPGA instance. The number of these signals depends on the programming options selected. The table following the figure lists the interface signals of an eFPGA instance.

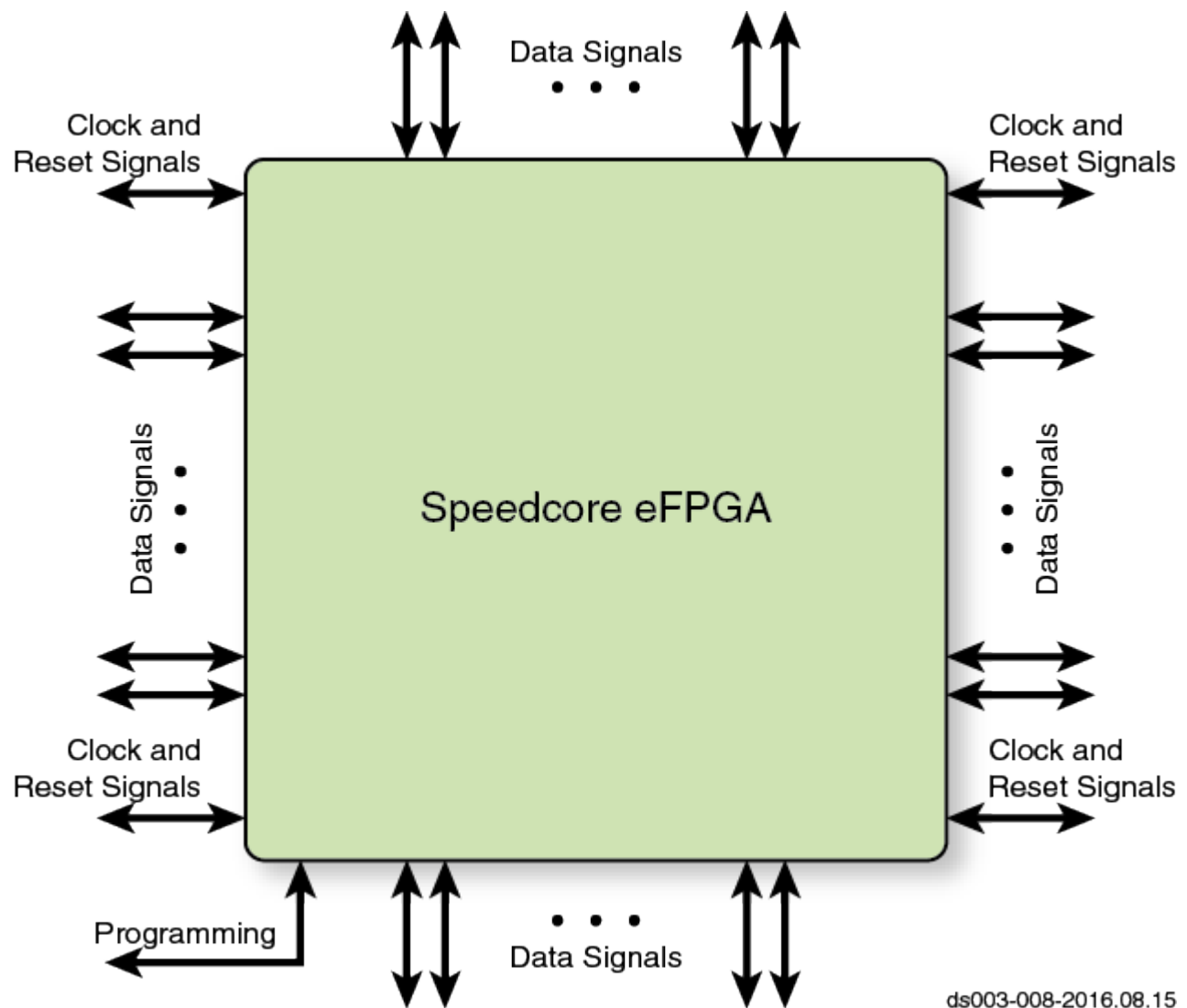


Figure 10: Speedcore eFPGA Interfaces

Pins

The following table describes the input/output pins of a Speedcore eFPGA instance:

Table 4: Speedcore eFPGA Pins

Pin Name	Direction	Description
i_user_*_lut/bram /lram/dsp_*[n:0]	Input	Data inputs to the programmable core. The bit width depends on size and customer requirements.
o_user_*_lut/bram /lram/dsp_*[m:0]	Output	Data outputs from the programmable core. The bit width depends on size and customer requirements.

Pin Name	Direction	Description
i_user_*_trunk/mt/ branch_*[c:0]	Input	Clock inputs to the programmable core.
o_user_*_trunk/mt/ branch_*[d:0]	Output	Clock outputs from the programmable core.
i_config_*[x:0]	Input	Bitstream data, control and configuration setting selection pins for a Speedcore instance. The width of these signals depends on the selected programming option.
o_config_*[y:0]	Output	Status output and signaling pins for a Speedcore instance.

Chapter - 4: Speedcore eFPGA In-System Debug

Snapshot is the real-time design debugging tool for Achronix FPGAs and eFPGA cores. The Snapshot debugger, which is embedded in ACE software, delivers a practical platform to observe the signals of a user's design in real-time. To use the Snapshot debugger, the Snapshot macro needs to be instantiated inside the user's RTL. After instantiating the macro and programming the device, the user will be able to debug the design through the Snapshot Debugger GUI within ACE, or via the `run_snapshot` TCL command API.

The Snapshot macro can be connected to any logic signal mapped to the Achronix core, to monitor and potentially trigger on that signal. Monitored signal data is collected in real time in regular BRAMs prior to being transferred to the ACE Snapshot GUI. The Snapshot macro has configurable monitor width and depth, as well as other configuration parameters, to allow user control over resource usage.

The ACE Snapshot GUI interacts with the hardware via the JTAG interface. Interactively specified trigger conditions are transferred to the design, and collected monitor data is transferred back to the GUI, which displays the data using a builtin waveform viewer. The figure below shows the components involved in a Snapshot debug session.

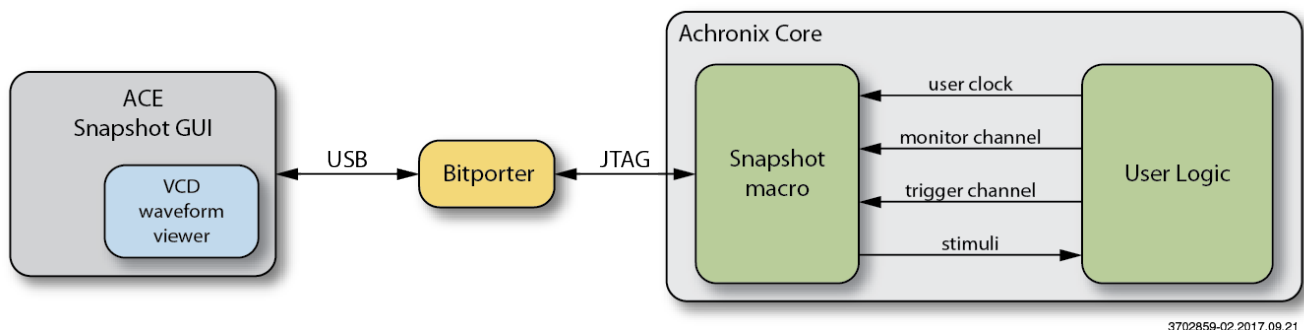


Figure 11: Snapshot Overview

Features

The Snapshot macro samples user signals in real time, storing the captured data in one or more BRAMs. The captured data is then communicated through the JTAG interface to the ACE Snapshot GUI. The implementation supports the following features:

- Monitor channel capture width of 1 to 4064 bits of data.
- Monitor channel capture depth of 512 to 16384 samples of data at the user clock frequency.
- Trigger channel width of 1 to 40 bits.
- Supports up to three separate sequential trigger conditions. Each trigger condition allows for the selection of a subset of the trigger channel, with AND or OR functionality.
- Bit-wise support for edge- (rise/fall) or level-sensitive triggers.
- The ACE Snapshot GUI allows specification of trigger conditions and circuit stimuli at runtime.
- An optional initial trigger condition, specified in RTL parameters, to allow capture of data immediately after startup, before interaction with the ACE Snapshot GUI.

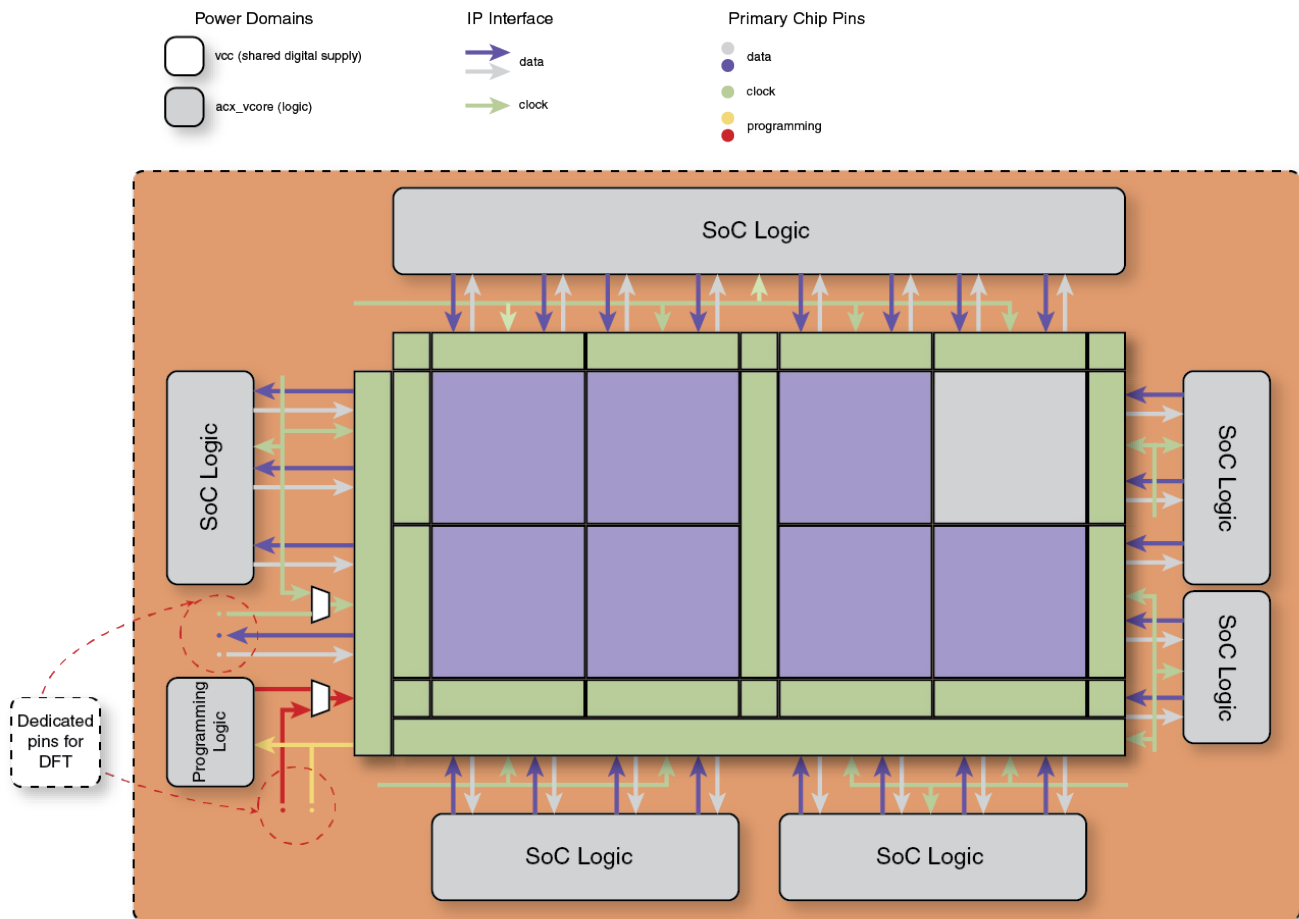
- A stimuli interface, 0 to 512 bits wide, that allows the user to drive values into the Achronix core logic from Snapshot. Stimuli values are specified with the ACE Snapshot GUI and made available before data capture.
- Optionally, the data capture can include values before the trigger occurred. This "pre-store" amount can be specified in increments of 25% of the depth.
- Captured data is saved in a standard VCD waveform file. The ACE Snapshot GUI includes a waveform viewer for immediate feedback.
- The VCD waveform file includes a timestamp for when the Snapshot was taken.
- ACE automatically extracts the names of the monitored signals from the netlist, for easy interpretation of the waveform.
- A repetitive trigger mode, in which repeated Snapshots are taken and collected in the same VCD file.
- The JTAG interface can be shared with the user design.
- A TCL batch/script mode interface is provided via the `run_snapshot` TCL command

Chapter - 5: Speedcore eFPGA Integration Flow

Physical Integration with Customer ASICs

The Speedcore™ eFPGA is provided as a fixed-transistor-layout building block that integrates with industry-standard ASIC flows such as Synopsys Design Compiler and IC Compiler. The following collateral will be provided:

- Verilog definition of logical connectivity at boundary
- Liberty timing library for timing closure at the boundary
- LEF defining the physical floorplan, pins, and metal blockages
- GDS/Oasis physical database



340223-01.2019.01.15

Figure 12: Sample eFPGA Instantiation

The data inputs/outputs and clock inputs can come from the ASIC logic or can come directly from the package pins (balls) of the ASIC. The programming interface must have access to the package pins of the ASIC to enable Speedcore programming. In addition, a certain number of Simulation and Validation data inputs/outputs must be

accessible through the package pins for eFPGA IP standalone testing. Details on the number of pins and connectivity will be provided in the *Design and Integration Manual*.

Simulation and Validation

The Speedcore eFPGA will be supplied with ACE (Achronix CAD Environment) software that provides a complete solution for simulating, synthesizing, mapping, and timing any user logic in the eFPGA fabric. The behavioral models or gate-level netlists representing the logic mapped inside the FPGA can then be directly integrated into the user's simulation/verification flow. In addition, SDF-annotated simulation models and standard Liberty timing models of the user logic can be emitted and integrated into the user's system-level timing validation flow.

Revision History

The following table lists the revision history of this document.

Version	Date	Description
1.0	07 Feb 2019	<ul style="list-style-type: none">Initial release.
1.1	29 Jul 2019	<ul style="list-style-type: none">First public release: removed confidential markings.Speedcore eFPGA Architecture (see page 7): updated the key feature table for the LRAM2k.
2.0	02 Oct 2020	<ul style="list-style-type: none">Updates for new Speedcore devices.
2.0.1	27 Apr 2021	<ul style="list-style-type: none">Document made public (removed confidential marking).