Copyright © 2016 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
2953 Bunker Hill Lane, Suite 101
Santa Clara, CA 95054
USA
Phone : 855.GHZ.FPGA (855.449.3742)
E-mail : info@achronix.com
Table of Contents

Chapter - 1: Overview ................................................................. 4
  Introducing Speedcore eFPGA ................................................. 4
  Feature Summary ................................................................. 4
    Functionality .................................................................. 5
    Programming .................................................................. 5
    Process Technology .......................................................... 5
    Power Supplies ............................................................... 5

Chapter - 2: Speedcore Architecture .............................................. 6
  Block Floorplan ................................................................. 6
  Interconnect ........................................................................ 7
  Clock Network .................................................................... 8
  Interface Cluster .................................................................. 9
  Logic Fabric – Reconfigurable Logic Block ....................... 11
  Memory Resources .............................................................. 12
    Block RAMs .................................................................. 12
    Logic RAM ..................................................................... 14
  DSP64 Blocks .................................................................... 16

Chapter - 3: Speedcore IP Interface .................................................. 17
  Interfaces ........................................................................... 17
    Data Signals ................................................................... 17
    Clock Inputs ..................................................................... 17
    Programming Interface .................................................... 17
  Pins .................................................................................. 18

Chapter - 4: Speedcore In-System Debug ....................................... 20

Chapter - 5: Speedcore 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 architecture includes look-up-table, memory, and DSP building blocks that are designed in a modular structure which allows customers to define any quantity of resources required for their end system.

Achronix delivers the Speedcore IP in GDSII format for the customer to integrate into their ASIC and delivers the Achronix ACE design tools that are used to compile designs into the Speedcore eFPGA.

Feature Summary

Because Speedcore eFPGA is an embeddable IP, it does not include programmable I/O — it is designed to be completely surrounded by the end user ASIC (see the figure below).

---

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** – 4-input look-up-tables (LUTs) plus integrated wide mux functions and fast adders
- **Logic RAM** – up to 4 kb per memory block
- **Block RAM** – up to 20 kb per memory block
- **DSP64** – each block has a $18 \times 27$ multiplier, 64-bit accumulator and 27-bit pre-adder

There are design rules that dictate minimum and maximum relative quantities for each of the available resources. Customers submit their requirements to Achronix and receive back a Speedcore specification document that defines the exact resource count for their Speedcore eFPGA.

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

- **JTAG**
- **Parallel CPU** ($\times 1, \times 8, \times 16, \times 32, \times 128$ data width modes)
- **Serial flash** (1 or 4 flash devices)

Process Technology
Speedcore eFPGAs can be built on any digital process technology. Achronix charges a process technology port fee if the customer needs the Speedcore IP on a process technology and metal stack not currently supported by Achronix. Please contact Achronix directly to obtain a list of process technologies and metal stacks that are currently supported.

Power Supplies
The Speedcore eFPGA requires three external power supplies:

- **$V_{DDL}$** – Power rail for the programmable logic fabric.
- **$V_{CFG}$** – Power rail for the configuration cells in the core.
- **$V_{DD}$** – Power rail for the digital power supply of the ASIC boundary ring of the core.

Please refer to the *Speedcore Power User Guide* (UG066) for details on connectivity and power rail sharing.
Chapter - 2: Speedcore Architecture

Block Floorplan

The resource types are arranged in homogeneous columns. The height of the columns, number of columns and mix of column contents are customizable as defined by the customer. Implementing the exact mix of resources is performed by Achronix using automation, based on the specification set by the customer.

The Speedcore™ fabric performance is design dependent and is typically between 300 MHz and 500 MHz ($F_{\text{MAX}}$).

Figure 2: Sample Speedcore eFPGA Floorplan
Interconnect

The reconfigurable logic blocks (RLBs), block RAMs (BRAMs), logic RAMs (LRAMs) and digital signal processing blocks (DSP64s) are connected by a uniform global interconnect. This 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.

Figure 3: Speedcore eFPGA Interconnect
Clock Network

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

The global clock network is the hierarchical network that feeds resources in the eFPGA fabric. The global clock trunk runs vertically up and down the center of the core (gray stripe in the previous figure), sourced by global clock muxes at the top and bottom of the global trunk. The sources driven down the trunk are then are channeled out the balanced clock mini-trunks to 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 surround host ASIC. The clocks connect to the core through the surround 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 4: Global Core Clock Network
The Interface Cluster is the portion of the Speedcore boundary ring that contains the registers, ACB logic and the connectivity to the Speedcore top-level pins. The figure below shows the details of an interface cluster. The upper half shows the ingress and egress path for user signals to the core. Both paths can be optionally registered.

*Figure 5: Interface Clock Network*
Figure 6: Speedcore Interface Cluster Detail
Logic Fabric – Reconfigurable Logic Block

The reconfigurable logic block (RLB) contains 4-input look-up-tables (LUTs), dedicated 4-to-1 and 8-to-1 MUXes, registers and a 4-bit fast arithmetic logic unit (ALU) to implement the eFPGA logic functionality.

**Figure 7: RLB Details**
Memory Resources

Block RAMs
The block RAM (BRAM) implements a dual-ported memory block where each port can be independently configured with respect to size and function. The BRAM 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 (per Block RAM) are summarized in the table below.

Table 1: Block RAM Key Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block RAM Size</td>
<td>20 kb</td>
</tr>
<tr>
<td>Organization</td>
<td>512 × 40(†), 1k × 20, 1k × 18, 1k × 16, 2k × 10, 2k × 9, 2k × 8, 4k × 5, 4k × 4, 8k × 2, 16k × 1 (†)</td>
</tr>
<tr>
<td>Performance</td>
<td>500 MHz</td>
</tr>
<tr>
<td>Physical Implementation</td>
<td>Columns throughout device</td>
</tr>
<tr>
<td>Number of Ports</td>
<td>Dual port (independent read and write)</td>
</tr>
<tr>
<td>Port Access</td>
<td>Synchronous</td>
</tr>
</tbody>
</table>

Note
† 512 × 40 only available as simple dual-port function.

The BRAM ports are illustrated in the following figure:
Figure 8: BRAM Ports

Organization
The organization of each BRAM port can be independently configured.

Note
Access from opposite ports is not required to have the same organization; however, the number of total memory bits on each port must be the same.
Operation
The read and write operations are both synchronous. For higher performance operation, an additional output register can be enabled, which will add an additional cycle of read latency. The initial value of the memory contents may be specified by the user from either parameters or a memory initialization file. The initial/reset values of the output registers may also be specified by the user. The reset values are independent of the initial (powerup) values. The porta_write_mode/portb_write_mode parameters define the behavior of the output data port during a write operation. When porta_write_mode/portb_write_mode is set to write_first, the douta/doutb is set to the value being written on the dina/dinb port during a write operation. Setting porta_write_mode/ portb_write_mode to no_change keeps the douta/ doutb port unchanged during a write operation to porta/portb. Conflict arises when the same memory cell is accessed by both ports within a narrow window and one or both ports are writing to memory. If this condition occurs, the contents of the memory and the output data for the colliding address may be undefined, but no damage will occur to the Speedcore eFPGA.

Built in FIFO Controller
The BRAMFIFO implements a 20 kb FIFO memory block utilizing the embedded BRAM blocks with dedicated pointer and flag circuitry. The BRAMFIFO can be configured to support a variety of widths and depths, ranging from 512-bit depth with 40-bit data down to 16k depth with 1-bit data. The read and write clocks may be either synchronous or asynchronous with respect to each other. If the user read and write clocks are the same clock, the user may set the sync_mode to 1'b1 to enable faster and synchronous generation of the status flags and FIFO pointer outputs.

Error Correction
Error correction is available only in 40-bit (simple dual-port) mode. The built-in error correction logic provides single-bit error correction and dual-bit error detection on a 32-bit data bus, using eight internal overhead bits. If the internal ECC logic is not used, all 40 bits can be used for other purposes such as tagging and various control functions.

Initialization and Reset
Initial content of the block RAMs can be optionally loaded during device configuration if specified by the user. Otherwise, the block RAM initial content is undefined. On reset, the RAM contents are unchanged, but the output register, if used, assumes the specified reset value.

The initial state of the RAM read outputs can also be optionally loaded during device configuration.

Logic RAM
The logic RAM (LRAM) implements a 4,096-bit memory block with one write port and one read port. The LRAM is configured as a 128 × 32 simple dual-port (1 write port, 1 read port) RAM. The LRAM 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 FPGA fabric. A summary of LRAM features is shown in the table below.

Table 2: Logic RAM Key Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic RAM size</td>
<td>4,096 bits</td>
</tr>
<tr>
<td>Organization</td>
<td>128 × 32</td>
</tr>
</tbody>
</table>
### Table

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>500 MHz</td>
</tr>
<tr>
<td>Physical implementation</td>
<td>Dedicated columns</td>
</tr>
<tr>
<td>Number of ports</td>
<td>Simple dual port (one read, one write)</td>
</tr>
<tr>
<td>Port access</td>
<td>Synchronous writes, asynchronous reads</td>
</tr>
</tbody>
</table>

The LRAM ports are shown in the following figure:

---

**Figure 9: LRAM Ports**

**Organization**

The LRAM is configured as a 128 × 32 simple dual-port (one write port, one read port) RAM.

**Initialization and Reset**

By default, the contents of the LRAM memory are undefined. If the user wants the initial contents to be defined, he may assign them from either a file pointed to by the `mem_init_file` parameter or assign them from the value of the `mem_init` parameter.
**Operation**

The LRAM has a synchronous write port. The read port is configured for asynchronous read operations with an optional output register. The memory is organized as little-endian order with bit 0 mapped to bit 0 of parameter `mem_init_00` and bit 4095 mapped to bit 255 of parameter `mem_init_15`.

**DSP64 Blocks**

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 eFPGA logic fabric and LRAM blocks to implement math functions. Refer to the *Speedcore DSP64 User Guide* for more details.

The DSP64 blocks have the following functions:

- 27-bit preadder
- $18 \times 27$ multiplication/accumulation with programmable load value
- Add/subtract
- Saturating add/subtract support
- $(A \pm B)^2$ and $(A \pm B)^2 + constant$
- Output rounding

![Figure 10: DSP64 Block](image)
Chapter - 3: Speedcore IP Interface

Interfaces
There are three sets of interfaces to the Speedcore™ block (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 16 interface clocks per cluster, per side.

Programming Interface
There is a dedicated set of signals for programming of eFPGA block. The number of these signals depends on the programming options selected.

The table following the figure lists the interface signals of the eFPGA block.
The following table describes the input/output pins of the Speedcore eFPGA core:

**Table 3: Speedcore eFPGA Pins**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i_data_w/e/n/s[n:0]</td>
<td>Input</td>
<td>Data inputs to the programmable core. The bit width depends on size and customer requirements.</td>
</tr>
<tr>
<td>o_data_w/e/n/s[m:0]</td>
<td>Output</td>
<td>Data outputs from the programmable core. The bit width depends on size and customer requirements.</td>
</tr>
<tr>
<td>Pin Name</td>
<td>Direction</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>i_clock[c:0]</td>
<td>Input</td>
<td>Clock inputs to the programmable core.</td>
</tr>
<tr>
<td>o_clock[d:0]</td>
<td>Output</td>
<td>Clock outputs from the programmable core.</td>
</tr>
<tr>
<td>i_config[x:0]</td>
<td>Input</td>
<td>Bitstream data, control and configuration setting selection pins for Speedcore. The width of these signals depends on the selected programming option.</td>
</tr>
<tr>
<td>o_config[y:0]</td>
<td>Output</td>
<td>Status output and signaling pins for Speedcore.</td>
</tr>
</tbody>
</table>
Chapter - 4: Speedcore In-System Debug

Snapshot™ is the realtime design debugging tool, which is embedded in the ACE software and delivers a practical platform to evaluate the signals of a user’s design in realtime. To run Snapshot debugger tool, the Snapshot macro needs to be instantiated inside the RTL that is compiled and downloaded to the Speedcore™ eFPGA. After instantiating the macro and programming the device, the user will be able to debug the design through the ACE Snapshot Debugger GUI.

The Snapshot macro, when instantiated in the design, can be used to interface with any logic mapped to the Achronix FPGA core. Snapshot provides a JTAG interface at the FPGA pins to control/observe debug logic mapped to the core. This allows the ACE ChipTap Debugger GUI, which drives the JTAG interface, to control/observe the signals associated with the debug logic.

The Snapshot macro samples user signals in real time, sending the captured data back through JTAG interface. The Snapshot architecture is implemented to support the following features:

- Capture up to 144-bit wide data
- Capture always 1,024 samples of data at the user clock frequency

Figure 12: Snapshot Debugger
- Supports up to three separate 36-bit trigger conditions, each capable of operating on any user signal. Each trigger condition supports "don't care" feature (masking).
- All captured data will be read back serially with respect to TCK.
Chapter - 5: Speedcore 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

![Figure 13: Sample eFGPA Instantiation](image)
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, 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.

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>February 1, 2016</td>
<td>Initial released version.</td>
</tr>
</tbody>
</table>
| 1.1     | February 8, 2016 | Reorganized the structure:  
  - Sections Highlights and Introduction combined into a new chapter, Chapter 1 – "Overview".  
  - Sections Logic Fabric, Memory Sources and DSP18x10 Blocks moved to the chapter, Chapter 2 – "Architecture".  
  - Section Speedcore IP Interface moved to a separate chapter, Chapter 3 – "Speedcore IP Interface".  
  Added Chapter 6 – "Product Specification" DSP block nomenclature changed from DSP18x19 to DSP64.  
  All figures updated.  
  Other minor edits and corrections. |
| 1.2     | February 21, 2016|  
  - Update voltages listed in "Power Supplies" in Chapter 1 – Overview.  
  - Added section "Temperature Range" in Chapter 1 – Overview.  
  - BRAM size updated from 80 kb to 20 kb; port widths adjusted accordingly.  
  - Other updates and edits. |
| 1.3     | March 8, 2016    |  
  - Converted to Confluence |
| 1.4     | March 31, 2016   |  
  - Clarification regarding ECC mode.  
  - Minor edits and corrections. |
| 1.5     | April 4, 2016    |  
  - Updated DSP multiplier size to $18 \times 27$.  
  - Clarified information on process technology.  
  - Corrected information on clock counts. |
| 1.6     | April 6, 2016    |  
  - Changed the process description from "TSMC 16FPGL(FF16+ GL option)" to "TSMC 16FFplus-GL". |
| 1.7     | April 6, 2016    |  
  - Made corrections for BRAM/LRAM organization and operation. |
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8</td>
<td>July 25, 2016</td>
<td>• Re-structured content and moved device specific information, including resource counts, into an Appendix section.</td>
</tr>
<tr>
<td>1.9</td>
<td>August 5, 2016</td>
<td>• Corrected naming convention for LRAM and a connection in the RLB Details diagram.</td>
</tr>
</tbody>
</table>
| 1.10    | August 19, 2016 | • Updated wording for DSP64 pre-adder in text and block diagram.  
                     • Put in sections on the Clock Architecture and Interface Cluster.  
                     • Corrected eFPGA interfaces figure and modified description of eFPGA pins.  
                     • Standardized configuration memory cell power supply name to $V_{\text{CFG}}$. |