Virtex-5 FPGA Aurora 8B/10B Migration to Aurora 64B/66B

User Guide

UG519 (v1.0) June 24, 2009

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/24/09	1.0	Initial Xilinx release.

Table of Contents

Chapter 5: Aurora 64B/66B Core Advantages

About This Guide

This user guide provides instructions for migrating from the Xilinx LogiCORE™ IP Aurora 8B/10B core to the Aurora 64B/66B core in a Virtex®-5 FPGA. This guide also describes the function and operation of these cores and the differences between them, and provides information about using the cores.

About the Core

The Aurora 8B/10B core and the Aurora 64B/66B core are CORE Generator™ software cores included in the latest IP Update on the Xilinx IP Center. For detailed information about the cores, see www.xilinx.com/aurora.

Guide Contents

This manual contains the following chapters:

- Chapter 1, "Introduction"
- Chapter 2, "Difference between the Aurora Protocol Specifications"
- Chapter 3, "Differences between Aurora Cores"
- Chapter 4, "How to Migrate from the Aurora 8B/10B Core to the Aurora 64B/66B Core"
- Chapter 5, "Aurora 64B/66B Core Advantages"

Additional Resources

To find additional documentation, see the Xilinx website at:

http://www.xilinx.com/literature.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

http://www.xilinx.com/support.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *design_name*
Helvetica bold	Commands that you select from a menu	**File → Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**`ngdbuild`** *design_name*
	References to other manuals	See the *Development System Reference Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** [*`option_name`*] *`design_name`*
Braces { }	A list of items from which you must choose one or more	**`lowpwr =`**{**`on`**|**`off`**}
Vertical bar |	Separates items in a list of choices	**`lowpwr =`**{**`on`**|**`off`**}

Vertical ellipsis
.
.
.
	Repetitive material that has been omitted	`IOB #1: Name = QOUT'`
		`IOB #2: Name = CLKIN'`
		.
		.
		.

Horizontal ellipsis ...	Repetitive material that has been omitted	**`allow block`** *block_name loc1 loc2 ... locn;*

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-5 FPGA User Guide*.
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

XILINX®

Chapter 1

Introduction

Overview

There is an overwhelming trend towards serialization of data transmission from what was traditionally a parallel interface. With advances in communication technology, you can achieve gigahertz data-transfer rates in serial links without having to make trade-offs in data integrity.

Xilinx recommends the LogiCORE™ IP Aurora cores to address high-speed data rate requirements not met by other serialized protocols. The Aurora core on Xilinx FPGAs provides customers with an easy way to use serial transceivers in the FPGA and to provide an easy-to-use user application interface.

Aurora is a scalable, lightweight, link-layer protocol that is used to move data across point-to-point serial links. It is an open protocol, free of charge and can be implemented in any silicon device/technology. Aurora provides a transparent interface to the physical serial links, allowing upper layers of proprietary or industry-standard protocols, such as Ethernet and TCP/IP, to easily use these high-speed serial links. In cases where Aurora is used as a physical layer and industry standard interfaces for the upper layer protocol this can lead to higher connectivity performance while preserving software infrastructure investment.

Aurora is a very efficient low-latency protocol that uses the least possible amount of logic. Aurora also offers a rich and highly configurable feature set. Through unlimited bonded lanes Aurora can increase the bandwidth between devices. Instantiated through the CORE Generator™ software very little time is needed to integrate with existing user designs. Being lightweight, Aurora is well suited for serial point to point connectivity even in low logic count devices.

Although Aurora 8B/10B is a very efficient protocol the 8B/10B encoding imposes a 25% overhead. With Aurora 64B/66B, which imposes ~3% overhead, it is even more attractive to gain even more bandwidth out of the serial link.

About the Cores

Xilinx offers two versions of LogiCORE™ IP Aurora cores that are distinguished by the encoding performed on the serial link: the *Aurora 8B/10B* core and *Aurora 64B/66B* core. This guide discusses their differences and provides instructions to migrate an Aurora 8B/10B core design to an Aurora 64B/66B core design. This guide uses the Virtex®-5 FXT family in the example design to compare an Aurora 8B/10B core to an Aurora 64B/66B core. The cores are:

- Aurora 8B/10B core
  - Uses 8B/10B encoding/decoding

  - Supports all the FPGA families which have MGTs, Serializers-Deserializer (SerDes) that are used to realize the high-speed Aurora serial link in the FPGA
- Aurora 64B/66B core
  - Uses 64B/66B encoding/decoding
  - Supports only the Virtex-5 FXT/TXT families as the GTX gearbox block natively supports this encoding scheme.

Recommended Design Experience

Although the Aurora core is a fully verified solution, the challenge associated with implementing a complete design varies depending on the configuration and functionality of the application. For best results, previous experience building high-performance, pipelined FPGA designs using Xilinx implementation software and user constraints files (.ucf) is recommended.

Related Xilinx Documents

Prior to generating an Aurora core, users should be familiar with the following:

- Aurora documents:
  - UG352, *LogiCORE IP Aurora 8B/10B v4.1 Getting Started Guide*
  - UG353, *LogiCORE IP Aurora 8B/10B v4.1 User Guide*
  - UG237, *LogiCORE IP Aurora 64B/66B v2.1 User Guide*
  - SP002, *Aurora 8B/10B Protocol Specification*
  - SP011, *Aurora 64B/66B Protocol Specification*
- Documents located on the LocalLink product page: www.xilinx.com/locallink:
  - SP006, *LocalLink Interface Specification*
- Xilinx RocketIO Transceiver User Guide:
  - UG198, *Virtex-5 FPGA RocketIO GTX Transceiver User Guide*

Technical Support

For technical support, go to www.xilinx.com/support. Questions are routed to a team of engineers with expertise using the Aurora core.

Xilinx will provide technical support for use of this product as described in the *LogiCORE IP Aurora 8B/10B v3.1 for Virtex-4 FX FPGA User Guide* and the *LogiCORE IP Aurora 8B/10B v3.1 for Virtex-4 FX FPGA Getting Started Guide*. Xilinx cannot guarantee timing, functionality, or support of this product for designs that do not follow these guidelines, or for modifications to the source code.

Feedback

Xilinx welcomes comments and suggestions about the Aurora core and the accompanying documentation.

Core

For comments or suggestions about the Aurora core, please submit a WebCase from www.xilinx.com/support. Be sure to include the following information:

- Product name
- Core version number
- List of parameter settings
- Explanation of your comments

Document

For comments or suggestions about this document, please submit a WebCase from www.xilinx.com/support. Be sure to include the following information:

- Document title
- Document number
- Page number(s) to which your comments refer
- Explanation of your comments

Chapter 2

Difference between the Aurora Protocol Specifications

Aurora Protocol Specification

Aurora is a lightweight link-layer protocol that can be used to move data point-to-point across one or more high-speed serial lanes. The Aurora 8B/10B protocol core uses 8B/10B encoding and the Aurora 64B/66 protocol core uses 64B/66B uses 64B/66B encoding.

Figure 2-1 illustrates the Aurora channel.

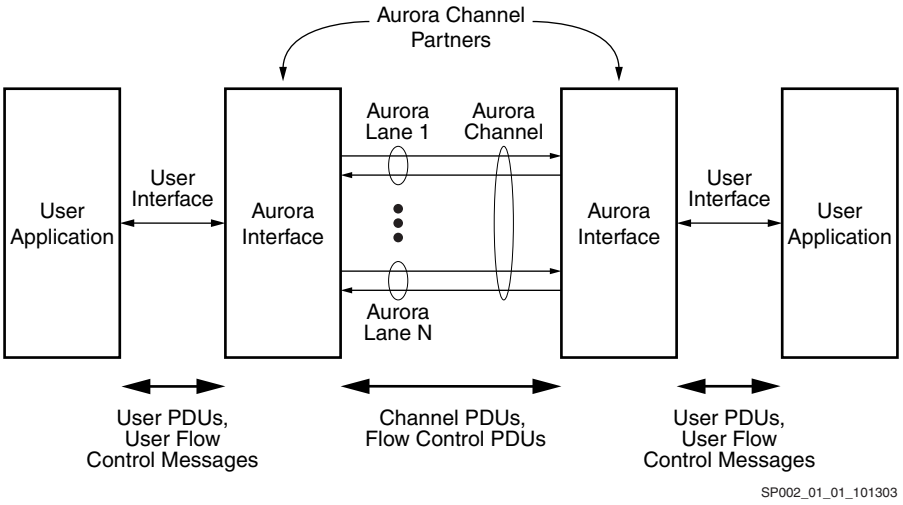


Figure 2-1: **Aurora Channel**

Table 2-1 shows a brief comparison between Aurora 8B/10B and Aurora 64B/66B.

Table 2-1: **Aurora 8B/10B vs. 64B/66B**

Feature	Aurora 8B/10B	Aurora 64B/66B
Overhead	25%	3.125%
Alignment	2 byte	8 byte
NFC PAUSE Value	Encoded	Not encoded
USER-K Blocks	NA	User defined
UFC Count	2 to 16 octets	1 to 256 octets
Encoding	8B/10B	64B/66B
Flow Control	Symbols	Block codes
Flow Control Interrupt	Cannot be interrupted	Can be interrupted
Streaming + Flow control	Not supported	Supported
Resource Utilization	LUT-2246 FF-2340	LUT-722 FF-1116

Main Differences

Data Transmission

The Aurora 64B/66B core has two more data types than the Aurora 8B/10B core:

- User K-Blocks:

  There are nine User K-Blocks available in Aurora 64B/66B. These control blocks are not decoded by the Aurora interface, but are instead passed directly to the user. These blocks can be used to implement application-specific control functions.

  For Aurora 8B/10B designs where control codes were implemented in-band, the customer can choose to remove them from the application design and implement them in User K-Blocks in the Aurora 64B/66B core. In short, User K-Blocks provides an easy way to control the Aurora blocks.

  Aurora 8B/10B designs can choose to use this capability or not in Aurora 64B/66B designs.

- Aurora 64B/66B Data, Data, Separator, Separator-7:

  These blocks are combined to create frames carrying user data:

  - Data blocks carry eight octets of data.
  - Separator blocks indicate the end of the current frame; the next frame begins on the next block. Separator blocks carry 0 to 6 octets of data: one byte in the block is used to indicate how many octets in the block are valid. Figure 2-2 shows the format of the Separator block code.

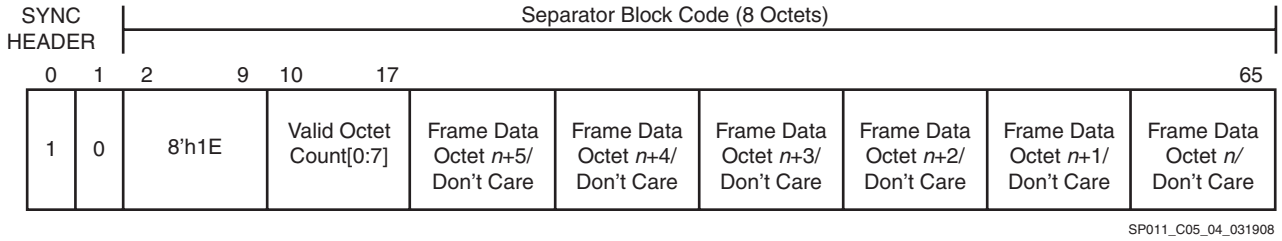


Figure 2-2: **Separator Block Code**

  - Separator-7 blocks are the same as Separator blocks, but always carry exactly seven valid octets of data.

Aurora 8B/10B Protocol Data Unit (PDU)

Used to send the data

- SCP
- Pad
- ECP (4.5 bytes)

The Aurora 64B/66B core's Separator block has lower overhead than an Aurora 8B/10B PDU. For most applications this difference should not affect the user's Aurora design because the LocalLink interfaces are the same in Aurora8B/10B and Aurora 64B/66B. One

difference that can occur is in the user application interface bus width (for details, see "LocalLink Interface," page 26.

Table 2-2 shows the relative priority of all the blocks for transmission, valid at all times, except when the transmitter is processing a flow control request.

Table 2-2: **Normal Aurora 64B/66B Block Transmission Priority**

Block Name	Priority
Clock Compensation	1 (Highest)
Not Ready	2
Channel Bonding	3
Native Flow Control	4
User Flow Control, Data blocks carrying UFC message	5
User K-Blocks	6
Data, Separator, Separator-7	7
Idle	8

During native flow control (NFC) countdown or user flow control (UFC) countdown, idles have a higher priority than data.

Flow Control

Aurora 64B/66B flow control is very similar to Aurora 8B/10B flow control with four key differences listed in Table 2-3.

Table 2-3: **Key Flow Control Differences between the Aurora Protocols**

Aurora 64B/66B	Aurora 8B/10B
Uses block codes and 64B/66B encoding.	Uses 8B/10B symbols and 8B/10B encoding.
User flow control messages can be 1 to 256 octets long.	User flow control messages are only 2 to 16 octets long.
User flow control messages can be interrupted by control characters at block boundaries. The user flow interface is independent.	Uses a multiplexed user flow control interface (with User data).
Native flow control PAUSE intervals are not encoded because the PAUSE value is 8 bits.	The Aurora 8B/10B PAUSE field is encoded and is 4 bits.

Although the flow control interfaces differ a little between Aurora 8B/10B and Aurora 64B/66B (Table 2-4, page 17), the user's designs can be easily migrated.

Table 2-4: **Additional Flow Control Differences between the Aurora Protocols**

Feature	Aurora 64B/66B	Aurora 8B/10B
Interface	Has more detailed ports.	Has fewer detailed ports.
Native Flow Control	XOFF has the port separated and the PAUSE value is no longer encoded.	Uses NFC codes. Has less PAUSE number choices.
User Flow Control	Has its own UCF TX data ports.	Shares the UCF TX data port with TX data.

Initialization

Aurora 64B/66B initialization differs from Aurora 8B/10B initialization in several key ways:

- The number of initialization messages and handshakes is reduced
- There is no channel verification stage
- The 64B/66B block lock state machine defined in IEEE 803.3ae for 10-Gigabit Ethernet is now used for block alignment and error handling
- The Aurora protocol recommends a specific CRC polynomial for compatibility with the 64B/66B scrambling polynomial

Initialization is associated with the physical layer, making it transparent to the user. The Aurora core completes the initialization process automatically.

Simplex Mode

Although both the Aurora 8B/10B and Aurora 64B/66B cores offer a simplex mode, the Aurora 64B/66B simplex TX core sends periodic Channel Bonding sequences to maintain a more reliable synchronization with the receiver. The transmitter and receiver can stay better synchronized because the Aurora 64B/66B implements a timer-based synchronization as opposed to time-out counters in Aurora 8B/10B.

Streaming Mode

Aurora 64B/66B implements the same streaming mode as Aurora 8B/10B but offers more functions. In particular Aurora 64B/66B supports flow control in the streaming mode as well.

XILINX®

Chapter 3

Differences between Aurora Cores

Xilinx provides two Aurora cores: the Aurora 8B/10B core and the Aurora 64B/66B core. This chapter compares and contrasts the features of these two cores.

Cores Features

Aurora 8B/10B Core for the Virtex-5 FPGA

The features of the Aurora 8B/10B core supported in the Virtex®-5 LXT/SXT/FXT/TXT family are:

- Supports up to 16 RocketIO™ GTP or GTX transceivers
- Supports line rates
  - In GTP transceivers: 500 Mbps to 3.75 Gbps
  - In GTX transceivers: 750 Mbps to 6.5 Gbps
- Aurora v2.0 specification compliant (8B/10B encoding)
- Low resource cost (see "Comparison of the Resource Usage between Aurora Cores," page 21)
- Easy-to-use framing and flow control
- Automatically initializes and maintains the channel
- Full-duplex or simplex operation
- LocalLink (framing) or streaming user interface
- Aurora 8B/10B core is backward compatible. For example. a core in the Virtex-4 FX family can talk to a core in the Virtex-5 family.

Aurora 64B/66B Core for the Virtex-5 FPGA

The features of the Aurora 64B/66B core supported in the Virtex-5 FXT/TXT families are:

- Supports up to16 RocketIO GTX transceivers
- Supports line rates
  - In GTX transceivers: 750 Mbps to 6.5 Gbps
- Aurora 64B/66B v1.1 specification compliant (64B/66B encoding with lower overhead)
- Low resource cost (see "Comparison of the Resource Usage between Aurora Cores," page 21)
- Easy-to-use framing interface and flow control
- Automatically initializes and maintains the channel
- Full-duplex or simplex operation
- LocalLink (framing) or streaming user interface with lower transmission overhead

Other Aurora IP

Xilinx also supports the Aurora 8B/10B core named v4fx_aurora_8b10b for the Virtex-4 FX family.

64B/66B Coding/Decoding in the Aurora Core

Overview

In data networking and transmission, 64B/66B is a line code that transforms 64-bit data to 66-bit line code to achieve DC-balance reducing data dependent jitter and bounded disparity, and yet provide enough state changes to allow clock recovery. This means that there are just as many 1s as 0s in a string of two symbols, and that there are not too many 1s or 0s in a row. This is an important attribute in a signal that needs to be sent at high rates because it helps reduce intersymbol interference.

The 8B/10B coding adds 25% overhead (10/8 = 1.25) whereas the overhead of the 64B/66B coding is only 3.125% (66/64 = 1.03125). The 64B/66B coding provides a much more efficient line coding thereby giving more bandwidth for the user's design.

64B/66B Encoder/Decoder

The GTX transceiver in the Virtex-5 FXT/TXT family contains a TX/RX gearbox to perform the 64B/66B encode/decode and scrambler/descrambler functions; all other functions are implemented in the FPGA fabric. The HDL example code is included with the Aurora core.

Figure 3-1, page 21 illustrates the structure of 64B/66B encoder/decoder in the Aurora 64B/66B core.

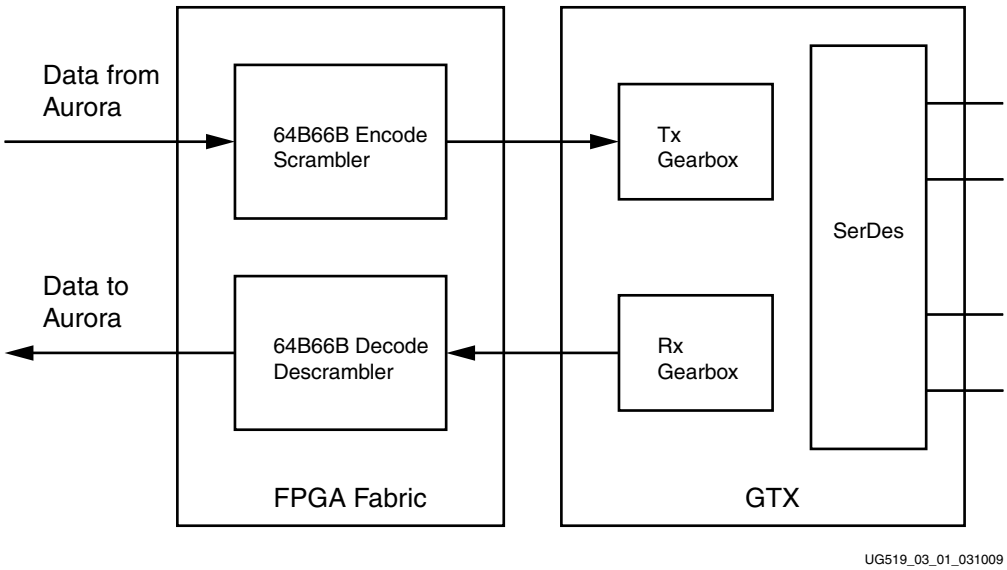


Figure 3-1: **Structure of the 64B/66B Encoder/Decoder**

The Aurora 64B/66B performs clock compensation and channel bonding in the fabric, whereas Aurora 8B/10B performs the same functions in the transceiver.

Comparison of the Resource Usage between Aurora Cores

This section compares the resource usage of the Aurora 8B/10B core to the Aurora 64B/66B core on the Virtex-5 FXT family.

Table 3-1 shows the number of look-up tables (LUTs) and flip-flops (FFs) used in selected Aurora 8B/10B framing modules.

Table 3-1: **Virtex-5 FXT Family Resource Usage for 8B/10B Framing**

Virtex-5 FXT Family			Framing			
			Duplex	Simplex		
Lanes	Lane Width	Resource Type	Full-Duplex	TX Only	RX Only	Both
1	4	LUTs	533	225	308	534
		FFs	633	260	393	645
2	4	LUTs	997	316	668	992
		FFs	1243	500	778	1272
4	4	LUTs	2246	521	1683	2198
		FFs	2340	907	1483	2385
8	4	LUTs	5866	928	4850	5768
		FFs	5228	1719	3587	5301
16	4	LUTs	17915	1745	15944	17672
		FFs	13450	3368	10219	13581

Table 3-2 shows the number of look-up tables (LUTs) and flip-flops (FFs) used in selected Aurora 64B/66B framing modules.

Table 3-2: **Virtex-5 FXT FPGA Resource Usage for 64B/66B Framing**

Virtex 5 FXT Family			Framing			
			Duplex	Simplex		
Lanes	Lane Width	Resource Type	Full-Duplex	TX Only	RX Only	RX/TX
1	4	LUTs	385	197	269	455
		FFs	705	272	494	760
2	4	LUTs	1041	275	907	1176
		FFs	1508	449	1158	1600
4	4	LUTs	2000	411	1759	2167
		FFs	2937	793	2274	3063
8	4	LUTs	3834	630	3364	3949
		FFs	5794	1489	4424	5883
16	4	LUTs	7567	1133	6696	7781
		FFs	11508	2875	8816	11661
24	4	LUTs	11298	1582	10018	11599
		FFs	17222	4242	13208	17443

From the comparison, although it appears that these two cores have almost the same resource usage though Aurora 64B/66B core has wider LocalLink width of 64 bit. Since the Aurora 64B/66B core is more efficient on the line more data is also seen on the User Application interface, hence a wider user interface is used compared to that of the Aurora 8B/10B core.

XILINX®

Chapter 4

How to Migrate from the Aurora 8B/10B Core to the Aurora 64B/66B Core

Generating the Aurora Cores

The Aurora 8B/10B and Aurora 64B/66B cores are produced by of the CORE Generator™ tool. The two main differences in the Wizard's of these two cores are:

- User K-Blocks
  - The Aurora 8B/10B core does not have these blocks.

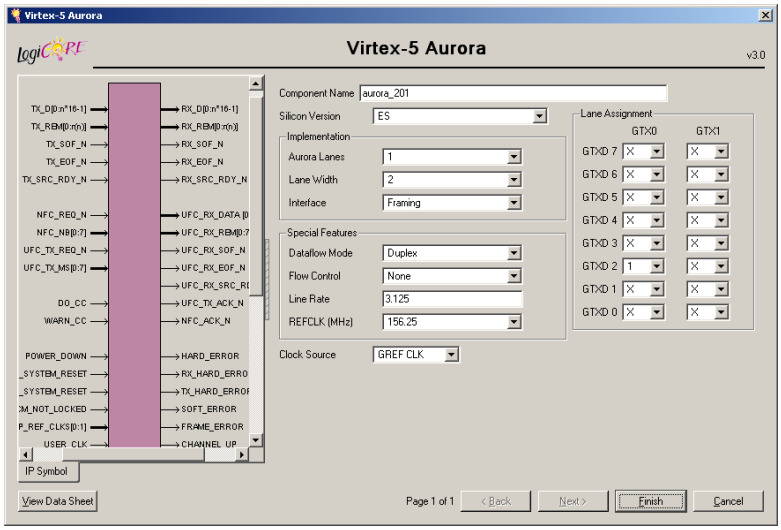
    The Aurora 64B/66B has nine available User K-Blocks. These control blocks are not decoded by the Aurora interface, but are instead passed directly to the user. These blocks can be used to implement application-specific control functions.

    These blocks have priority next to UFC but higher than user data. When a design is migrated from Aurora 8B/10B to Aurora 64B/66B, the designer can choose to ignore or take advantage of these capabilities. To minimize changes to the user design, the designer can simply ignore this function and deselect it in the CORE Generator software GUI.
- Lane Width
  - The RocketIO™ GTX transceivers support 2- and 4-byte lane width.
    - The Aurora 64B/66B core can use only the 4-byte option
    - The Aurora 8B/10B core can use either 2- or 4-byte option
  - The RocketIO GTP transceivers support only the 2-byte lane width.

  For details on the relationship between lane width and the LocalLink user interface width, see "LocalLink Interface," page 26.

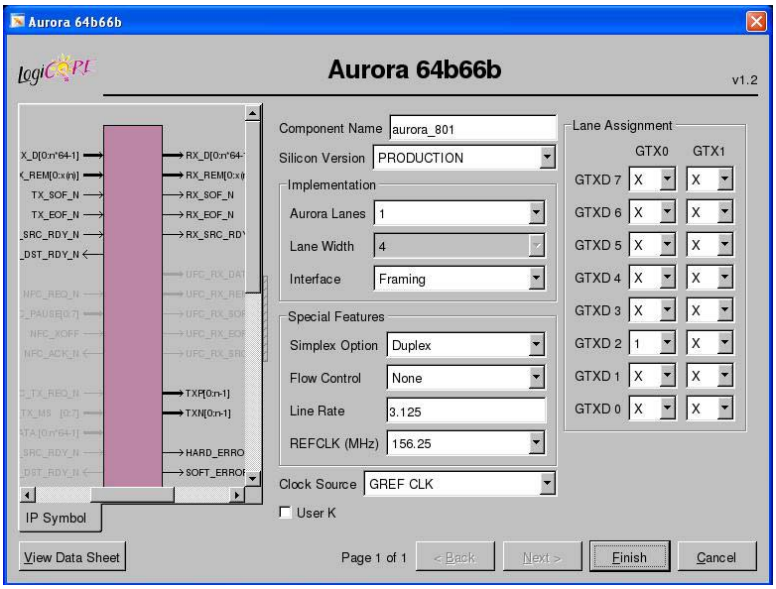
Figure 4-1, page 24 shows the CORE Generator software Wizard for the Aurora 8B/10B core.

Figure 4-2, page 24 shows the CORE Generator software Wizard for the Aurora 64B/66B core.



UG352_03_02_041608

Figure 4-1: Wizard for the Aurora 8B/10B Core



UG237_C3_01_050908

Figure 4-2: Wizard for the Aurora 64B/66B Core

Generated Files

The generated files are also similar between the Aurora 8B/10B and Aurora 64B/66B cores.

Example Design Hierarchy

- **<project directory>**
  Top-level project directory; name is user-defined.
  - <project directory>/<component name>
    Core readme file
    - <component name>/doc
      Product documentation
    - <component name>/example_design
      Example design files
      - /example_design/cc_manager
        Verilog/VHDL design files for the clock management block
      - /example_design/clock_module
        Verilog/VHDL design files for the clocking blocks
      - /example_design/gtx
        Verilog/VHDL design files for the RocketIO transceiver
      - /example_design/traffic_gen_and_check
        Verilog/VHDL design files for the frame generator and checker
      - /example_design/ucf
        Example design UCF files
    - <component name>/implement
      Implementation scripts and support files
      - /implement/results
        Implement script results
    - <component name>/simulation
      Simulation test bench and simulation script files
      - /simulation/functional
        Functional simulation files
    - <component name>/src
      Verilog/VHDL files for the core

Transferring an existing Aurora 8B/10B design to an Aurora 64B/66B directory structure is a simple process. Documentation, simulation files, implementation files and source code are in folders of the same names.

The source codes for the example designs are also similar for the two cores. For consistency, the HDL code has similar names and functions. The only difference is that Aurora 64B/66B has more files in the GTX modules than Aurora 8B/10B because Aurora 64B/66B has additional files for the 64B/66B encoder/decoder and other GTX logic functions.

LocalLink Interface

The Aurora 8B/10B and Aurora 644B/66B cores both use the LocalLink interfaces. The data transfer timing is the same so it is easy to migrate from an Aurora 8B/10B design to an Aurora 64B/66B design. The main difference is that the data width and the user clock are a little different.

For Aurora 8B/10B, the data width is [0:(8*n*-1)]. For Aurora 64B/66B, the data width is [0: (64*n*-1)].

For the same line rate, the user clock in Aurora 64B/66B is a little faster. The customer can obtain more bandwidth when using Aurora 64B/66B due to the different coding.

Table 4-1 shows an example of data width and user clock with different settings in the Aurora cores. The line rate is fixed to 5G.

In Table 4-1, for Aurora 8B/10B, *n* refers to the number of IP core user data bytes (*n* = Aurora lanes * lane width). For Aurora 64B/66B, the lane width can only be 4 bytes, and *n* is just equal to the number of Aurora lanes. Aurora lanes decide the number of GTP/GTX transceivers used in the channel. The user can get more bandwidth to select more lanes used in the design. However for a given line rate, the more lane width chosen has the effect of reducing the required clock rate of application logic connected to the Aurora core.

Table 4-1: **Example Data Width and User Clock with Different Settings**

Core	Aurora Lanes	Lane Width	*n*	Data Width	User Clock	GTP/GTX
Aurora 8B/10B	1	2	2	16	250 MHz	1 GTP
	1	4	4	32	125 MHz	1 GTP
	2	2	4	32	250 MHz	2 GTP
Aurora 64B/66B	1	4	1	64	78.125 MHz	1 GTX
	2	4	2	128	78.125 MHz	2 GTX

XILINX®

Chapter 5

Aurora 64B/66B Core Advantages

Easy to Use

Aurora 64B/66B is a new version of the Aurora protocol to use 64B/66B encoding instead of 8B/10B which is lower overhead. The Aurora 64B/66B protocol also provides more detailed functions than Aurora 8B/10B.

For example:

- Users can define their own control codes by using User K-Blocks
- In native flow control (NFC) the PAUSE value is not encoded and can be any number from 0 to 255
- User flow control (UFC) count has been enlarged from 16 octets to 256 octets.

Aurora 64B/66B also keeps the main functions and interfaces of Aurora 8B/10B. For example they use the same LocalLink interface. The flow control is stronger but the use is similar as before.

Other aspects such as initialization, error handling, and clock compensation are almost the same, thereby allowing for an easy migration from Aurora 8B/10B design to Aurora 64B/66B.

More Efficient

Aurora transmission efficiency includes two parts: *framing* efficiency and *coding* efficiency.

The clock compensation sequence, SCP, ECP, and separator for the frame consume the channel bandwidth. Aurora 64B/66B uses a Separator block for the frame and saves more bandwidth than Aurora 8B/10B. Table 5-1 is an example calculated after including overhead for clock compensation. It shows the efficiency for a single-lane channel and illustrates that the efficiency increases as frame length increases.

Table 5-1: **Efficiency Example**

User Data Bytes	Framing Efficiency %
100	96.12
1,000	99.18
10,000	99.89

To reduce the overhead further, 64B/66B encoding is used. This scheme encodes 64 bits of data into 66 bits, providing a 3% overhead compared to the 25% overhead of 8B/10B encoding.

Conclusion

Aurora 64B/66B is new version Aurora core to replace Aurora 8B/10B. It provides more functions and more efficient data transmission to the customer. If the user has an existing design using Aurora 8B/10B, it is easy to migrate that design to Aurora 64B/66B. The customers can generate the IP core by using the CORE Generator™ tool. The IP core will initialize the lanes automatically and the data, status, control interfaces are just as before.

For the user design the core can be easily migrated because the design code and structure are similar.

Table 5-2 shows the brief comparison between Aurora 8B/10B and Aurora 64B/66B cores.

Table 5-2: **Aurora 8B/10B vs. 64B/66B**

Feature	Aurora 8B/10B	Aurora 64B/66B
Overhead	25%	3.125%
Alignment	2 byte	8 byte
NFC PAUSE Value	Encoded	Not encoded
USER-K Blocks	NA	User defined
UFC Count	2 to 16 octets	1 to 256 octets
Encoding	8B/10B	64B/66B
Flow Control	Symbols	Block codes
Flow Control Interrupt	Cannot be interrupted	Can be interrupted
Streaming + Flow control	Not supported	Supported
Resource Utilization	LUT-2246 FF-2340	LUT-722 FF-1116