LogiCORE IP FIFO Generator v6.2

DS317 July 23, 2010 **Product Specification**

Introduction

The Xilinx LogiCORE™ IP FIFO Generator is a fully verified first-in first-out (FIFO) memory queue for applications requiring in-order storage and retrieval. The core provides an optimized solution for all FIFO configurations and delivers maximum performance (up to 500 MHz) while utilizing minimum resources. Delivered through the Xilinx CORE Generator™ software, the structure can be customized by the user including the width, depth, status flags, memory type, and the write/read port aspect ratios.

This document describes the features of the core, specifies the interfaces, and defines the input and output signals. For detailed information about designing with the core, see UG175, *FIFO Generator User Guide*.

Features

- FIFO depths up to 4,194,304 words
- FIFO data widths from 1 to 1024 bits
- Non-symmetric aspect ratios (read-to-write port ratios ranging from 1:8 to 8:1)
- Independent or common clock domains
- Selectable memory type (block RAM, distributed RAM, shift register, or built-in FIFO)
- Synchronous or asynchronous reset option
- Hamming Error Injection and Correction Checking (ECC) for built-in and block RAM FIFO
- First-word fall-through (FWFT)
- Full and empty status flags, and almost full and almost empty flags for indicating one-word-left
- Programmable full and empty status flags, set by user-defined constant(s) or dedicated input port(s)
- Configurable handshake signals
- Embedded register option for block RAM and built-in FIFO
- Fully configurable using the Xilinx CORE Generator

LogiCORE IP Facts	
Core Specifics	
Supported FPGA Device Families(1)	Virtex®-6, Virtex-5, Virtex-4, Spartan®-6, Spartan-3A/3AN/3A DSP, Spartan-3E, Spartan-3
Performance and Resources Used	See Table 10 through Table 13
Provided with Core	
Documentation	Product Specification User Guide Release Notes Migration Guide(2)
Design File Formats	NGC
Instantiation Template	VHDL, Verilog
Design Tool Requirements	
Implementation	Xilinx ISE® v12.2
Simulation	Mentor® ModelSim® v6.5c and above Cadence® Incisive Enterprise Simulator (IES) v9.2 and above
Supported Simulation Models	Verilog Behavioral(3) VHDL Behavioral(3) Verilog Structural VHDL Structural
Support	
Provided by Xilinx, Inc. @ www.xilinx.com/support	

1. For the complete list of supported devices, see Table 2, page 4 and the release notes for this core.
2. The Migration Guide provides instructions for converting legacy Asynchronous FIFO and Synchronous FIFO LogiCORE IP cores to FIFO Generator cores.
3. Behavioral models do not model synchronization delay. See the "Simulating Your Design" section of UG175, *FIFO Generator User Guide* for details.

Applications

In digital designs, FIFOs are ubiquitous constructs required for data manipulation tasks such as clock domain crossing, low-latency memory buffering, and bus width conversion. Figure 1 highlights just one of many configurations that the FIFO Generator supports. In this example, the design has two independent clock domains and the width of the write data bus is four times wider than the read data bus. Using the FIFO Generator, the user is able to rapidly generate solutions such as this one, that is customized for their specific requirements and provides a solution fully optimized for Xilinx FPGAs.

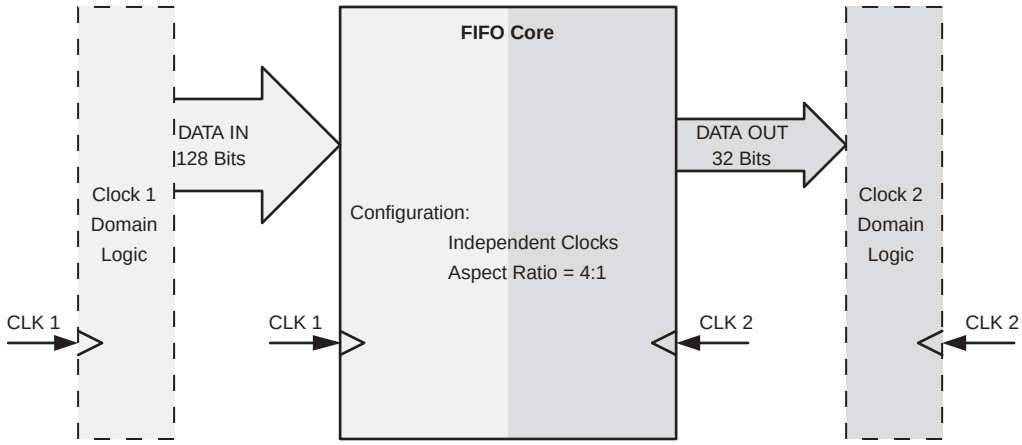

Figure 1: **FIFO Generator Application Example**

Feature Overview

Clock Implementation Operation

The FIFO Generator enables FIFOs to be configured with either independent or common clock domains for write and read operations. The independent clock configuration of the FIFO Generator enables the user to implement unique clock domains on the write and read ports. The FIFO Generator handles the synchronization between clock domains, placing no requirements on phase and frequency. When data buffering in a single clock domain is required, the FIFO Generator can be used to generate a core optimized for that single clock.

Virtex-6 and Virtex-5 FPGA Built-in FIFO Support

The FIFO Generator supports the Virtex-6 and Virtex-5 FPGA built-in FIFO modules, enabling large FIFOs to be created by cascading the built-in FIFOs in both width and depth. The core expands the capabilities of the built-in FIFOs by utilizing the FPGA fabric to create optional status flags not implemented in the built-in FIFO macro. The built-in Error Correction Checking (ECC) feature in the built-in FIFO macro is also available to the user.

See the Virtex-5 and Virtex-6 FPGA user guides for frequency requirements.

Virtex-4 FPGA Built-in FIFO Support

Support of the Virtex-4 FPGA built-in FIFO allows generation of a single FIFO primitive complete with fabric implemented flag patch, described in "Solution 1: Synchronous/Asynchronous Clock Work-Arounds," in UG070, *Virtex-4 FPGA User Guide*.

First-Word Fall-Through (FWFT)

The first-word fall-through (FWFT) feature provides the ability to look-ahead to the next word available from the FIFO without issuing a read operation. When data is available in the FIFO, the first word falls through the FIFO and appears automatically on the output bus (DOUT). FWFT is useful in applications that require low-latency access to data and to applications that require throttling based on the contents of the data that are read. FWFT support is included in FIFOs created with block RAM, distributed RAM, or built-in FIFOs in the Virtex-6 or Virtex-5 devices.

Memory Types

The FIFO Generator implements FIFOs built from block RAM, distributed RAM, shift registers, or the Virtex-6 and Virtex-5 FPGA built-in FIFOs. The core combines memory primitives in an optimal configuration based on the selected width and depth of the FIFO. The following table provides best-use recommendations for specific design requirements. The generator also creates single primitive Virtex-4 FPGA built-in FIFOs with the fabric implemented flag patch described in "Solution 1: Synchronous/Asynchronous Clock Work-Arounds," in the *Virtex-4 FPGA User Guide*.

Table 1: **Memory Configuration Benefits**

	Independent Clocks	Common Clock	Small Buffering	Medium-Large Buffering	High Performance	Minimal Resources
Virtex-6/Virtex-5 FPGA with Built-in FIFO	✓	✓		✓	✓	✓
Block RAM	✓	✓		✓	✓	✓
Shift Register		✓	✓		✓	
Distributed RAM	✓	✓	✓		✓	

Non-Symmetric Aspect Ratio

The core supports generating FIFOs with write and read ports of different widths, enabling automatic width conversion of the data width. Non-symmetric aspect ratios ranging from 1:8 to 8:1 are supported for the write and read port widths. This feature is available for FIFOs implemented with block RAM that are configured to have independent write and read clocks.

Embedded Registers in block RAM and FIFO Macros

In Virtex-6, Virtex-5 and Virtex-4 FPGA block RAM and FIFO macros, embedded output registers are available to increase performance and add a pipeline register to the macros. This feature can be leveraged to add one additional latency to the FIFO core (DOUT bus and VALID outputs) or implement the output registers for FWFT FIFOs. The embedded registers available in Virtex-6 FPGAs can be reset (DOUT) to a default or user programmed value for common clock built-in FIFOs. See Embedded Registers in block RAM and FIFO Macros in the *FIFO Generator User Guide* for more information.

Error Injection and Correction

The block RAM and FIFO macros are equipped with built-in Error Correction Checking (ECC) in the Virtex-5 FPGA architecture and built-in Error Injection and Correction Checking in the Virtex-6 FPGA architecture. This feature is available for both the common and independent clock block RAM or built-in FIFOs.

Supported Devices

Table 2 shows the families and sub-families supported by the FIFO Generator. For more details about device support, see the Release Notes.

Table 2: **Supported FPGA Families and Sub-Families**

FPGA Family	Sub-Family
Spartan-3	
Spartan-3E	
Spartan-3A	
Spartan-3AN	
Spartan-3A DSP	
Spartan-6	LX/LXT
Virtex-4	LX/FX/SX
Virtex-5	LXT/FXT/SXT/TXT
Virtex-6	CXT/HXT/LXT/SXT

FIFO Generator Configurations

Table 3 defines the supported memory and clock configurations.

Table 3: **FIFO Configurations**

Clock Domain	Memory Type	Non-symmetric Aspect Ratios	First-word Fall-Through	ECC Support	Embedded Register Support
Common	block RAM		✓	✓	✓ (1)
Common	DistributedRAM		✓		
Common	Shift Register				
Common	Built-in FIFO (2)		✓ (3)	✓	✓ (1)
Independent	block RAM	✓	✓	✓	✓ (1)
Independent	Distributed RAM		✓		
Independent	Built-in FIFO (2), (4)		✓ (3)	✓	

1. Embedded register support is only available for Virtex-6, Virtex-5 and Virtex-4 FPGA block RAM-based FIFOs, as well as Virtex-6 and Virtex-5 FPGA common clock built-in FIFOs.
2. The built-in FIFO primitive is only available in the Vortex-6, Virtex-5 and Virtex-4 architectures.
3. FWFT is supported for Built-in FIFOs in Virtex-6 and Virtex-5 devices only.
4. For non-symmetric aspect ratios, use the block RAM implementation (feature not supported in built-in FIFO primitive).

Common Clock: Block RAM, Distributed RAM, Shift Register

This implementation category allows the user to select block RAM, distributed RAM, or shift register and supports a common clock for write and read data accesses. The feature set supported for this configuration includes status flags (full, almost full, empty, and almost empty) and programmable empty and full flags generated with user-defined thresholds.

In addition, optional handshaking and error flags are supported (write acknowledge, overflow, valid, and underflow), and an optional data count provides the number of words in the FIFO. In addition, for the block RAM and distributed RAM implementations, the user has the option to select a synchronous or asynchronous reset for the core. For Virtex-6 and Virtex-5 FPGA designs, the block RAM FIFO configuration also supports ECC.

Common Clock: Virtex-6, Virtex-5 or Virtex-4 FPGA Built-in FIFO

This implementation category allows the user to select the built-in FIFO available in the Virtex-6, Virtex-5 or Virtex-4 FPGA architecture and supports a common clock for write and read data accesses. The feature set supported for this configuration includes status flags (full and empty) and optional programmable full and empty flags with user-defined thresholds.

In addition, optional handshaking and error flags are available (write acknowledge, overflow, valid, and underflow). The Virtex-6 and Virtex-5 FPGA built-in FIFO configuration also supports the built-in ECC feature.

Independent Clocks: Block RAM and Distributed RAM

This implementation category allows the user to select block RAM or distributed RAM and supports independent clock domains for write and read data accesses. Operations in the read domain are synchronous to the read clock and operations in the write domain are synchronous to the write clock.

The feature set supported for this type of FIFO includes non-symmetric aspect ratios (different write and read port widths), status flags (full, almost full, empty, and almost empty), as well as programmable full and empty flags generated with user-defined thresholds. Optional read data count and write data count indicators provide the number of words in the FIFO relative to their respective clock domains. In addition, optional handshaking and error flags are available (write acknowledge, overflow, valid, and underflow). For Virtex-6 and Virtex-5 FPGA designs, the block RAM FIFO configuration also supports ECC.

Independent Clocks: Virtex-6, Virtex-5 or Virtex-4 FPGA Built-in FIFO

This implementation category allows the user to select the built-in FIFO available in the Virtex-6, Virtex-5 or Virtex-4 FPGA architecture. Operations in the read domain are synchronous to the read clock and operations in the write domain are synchronous to the write clock.

The feature set supported for this configuration includes status flags (full and empty) and programmable full and empty flags generated with user-defined thresholds. In addition, optional handshaking and error flags are available (write acknowledge, overflow, valid, and underflow). The Virtex-6 and Virtex-5 FPGA built-in FIFO configuration also supports the built-in ECC feature.

FIFO Generator Features

Table 4 summarizes the supported FIFO Generator features for each clock configuration and memory type. For detailed information, see the *FIFO Generator User Guide*.

Table 4: **FIFO Configurations Summary**

FIFO Feature	Independent Clocks			Common Clock		
	Block RAM	Distributed RAM	Built-in FIFO	Block RAM	Distributed RAM, Shift Register	Built-in FIFO
Non-symmetric Aspect Ratios(1)	✓					
Symmetric Aspect Ratios	✓	✓	✓	✓	✓	✓
Almost Full	✓	✓		✓	✓	
Almost Empty	✓	✓		✓	✓	
Handshaking	✓	✓	✓	✓	✓	✓
Data Count	✓	✓		✓	✓	
Programmable Empty/Full Thresholds	✓	✓	✓(2)	✓	✓	✓(2)
First-Word Fall-Through(3)	✓	✓	✓	✓	✓	✓
Synchronous Reset				✓	✓	
Asynchronous Reset	✓(4)	✓(4)	✓	✓(4)	✓(4)	✓
DOUT Reset Value	✓	✓		✓	✓	✓(5)
ECC	✓(6)		✓(6)	✓(6)		✓(6)
Embedded Register	✓(7)			✓(7)		✓(7)

1. For applications with a single clock that require non-symmetric ports, use the independent clock configuration and connect the write and read clocks to the same source. A dedicated solution for common clocks will be available in a future release. Contact your Xilinx representative for more details.
2. For built-in FIFOs, the range of Programmable Empty/Full threshold is limited to take advantage of the logic internal to the macro.
3. First-Word-Fall-Through is not supported for the shift RAM FIFOs and Virtex-4 built-in FIFOs.
4. Asynchronous reset is optional for all FIFOs built using distributed and block RAM.
5. DOUT Reset Value is supported only in Virtex-6 FPGA common clock built-in FIFOs.
6. ECC is only supported for the Virtex-6 and Virtex-5 FPGAs and block RAM and built-in FIFOs.
7. Embedded register option is only supported in Virtex-6, Virtex-5 and Virtex-4 FPGA block RAM FIFOs, as well as Virtex-6 and Virtex-5 FPGA common clock built-in FIFOs. See <BL Blue>Embedded Registers in block RAM and FIFO Macros.

FIFO Interfaces

The following sections define the FIFO interface signals. Figure 2 illustrates these signals (both standard and optional ports) for a FIFO core that supports independent write and read clocks.

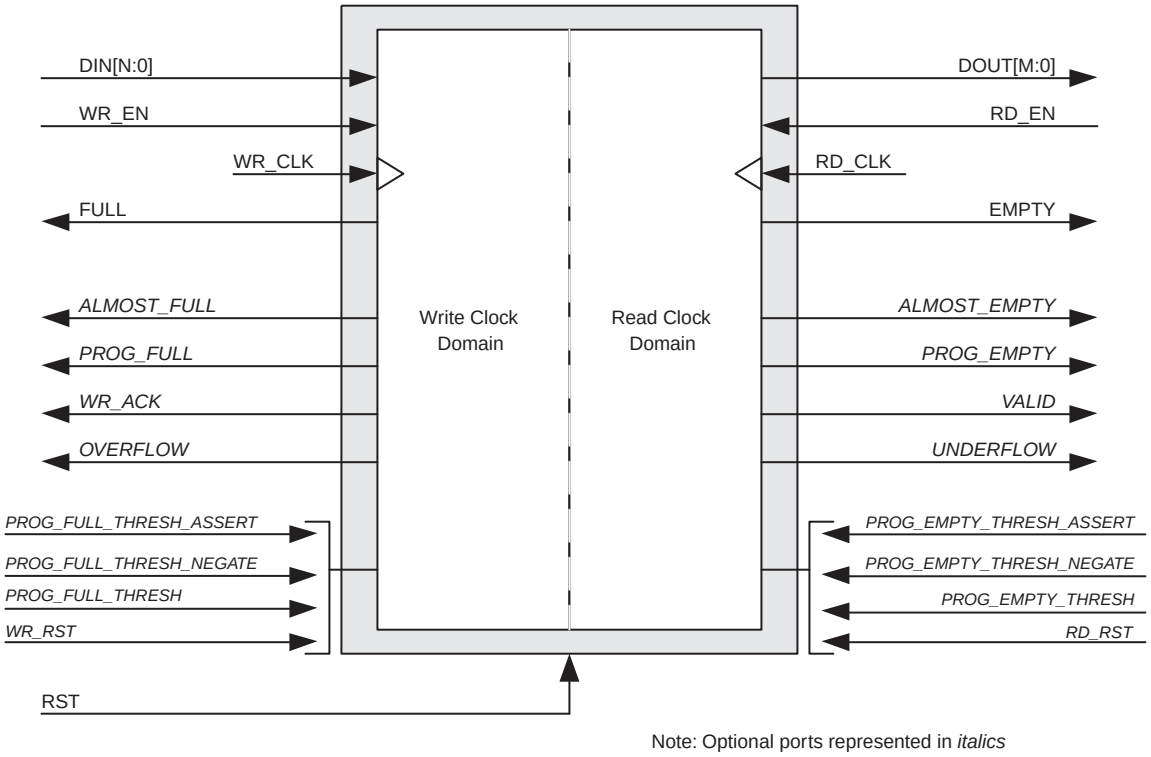

Figure 2: **FIFO with Independent Clocks: Interface Signals**

Interface Signals: FIFOs With Independent Clocks

The RST signal, as defined Table 5, causes a reset of the entire core logic (both write and read clock domains. It is an asynchronous input synchronized internally in the core before use. The initial hardware reset should be generated by the user. See the *FIFO Generator User Guide* for specific information about reset requirements and behavior.

Table 5: **Reset Signal for FIFOs with Independent Clocks**

Name	Direction	Description
RST	Input	Reset: An asynchronous reset signal that initializes all internal pointers and output registers.

Table 6 defines the write interface signals for FIFOs with independent clocks. The write interface signals are divided into required and optional signals and all signals are synchronous to the write clock (WR_CLK).

Table 6: **Write Interface Signals for FIFOs with Independent Clocks**

Name	Direction	Description
Required		
WR_CLK	Input	Write Clock: All signals on the write domain are synchronous to this clock.
DIN[N:0]	Input	Data Input: The input data bus used when writing the FIFO.
WR_EN	Input	Write Enable: If the FIFO is not full, asserting this signal causes data (on DIN) to be written to the FIFO.
FULL	Output	Full Flag: When asserted, this signal indicates that the FIFO is full. Write requests are ignored when the FIFO is full, initiating a write when the FIFO is full is non-destructive to the contents of the FIFO.
Optional		
WR_RST	Input	Write Reset: Synchronous to write clock. When asserted, initializes all internal pointers and flags of write clock domain.
ALMOST_FULL	Output	Almost Full: When asserted, this signal indicates that only one more write can be performed before the FIFO is full.
PROG_FULL	Output	Programmable Full: This signal is asserted when the number of words in the FIFO is greater than or equal to the assert threshold. It is deasserted when the number of words in the FIFO is less than the negate threshold.
WR_DATA_COUNT [D:0]	Output	Write Data Count: This bus indicates the number of words written into the FIFO. The count is guaranteed to never under-report the number of words in the FIFO, to ensure the user never overflows the FIFO. The exception to this behavior is when a write operation occurs at the rising edge of WR_CLK, that write operation will only be reflected on WR_DATA_COUNT at the next rising clock edge. If D is less than log2(FIFO depth)-1, the bus is truncated by removing the least-significant bits.
WR_ACK	Output	Write Acknowledge: This signal indicates that a write request (WR_EN) during the prior clock cycle succeeded.
OVERFLOW	Output	Overflow: This signal indicates that a write request (WR_EN) during the prior clock cycle was rejected, because the FIFO is full. Overflowing the FIFO is non-destructive to the contents of the FIFO.
PROG_FULL_THRESH	Input	Programmable Full Threshold: This signal is used to input the threshold value for the assertion and de-assertion of the programmable full (PROG_FULL) flag. The threshold can be dynamically set in-circuit during reset. The user can either choose to set the assert and negate threshold to the same value (using PROG_FULL_THRESH), or the user can control these values independently (using PROG_FULL_THRESH_ASSERT and PROG_FULL_THRESH_NEGATE).

Table 6: **Write Interface Signals for FIFOs with Independent Clocks *(Cont'd)***

Name	Direction	Description
PROG_FULL_THRESH_ASSERT	Input	Programmable Full Threshold Assert: This signal is used to set the upper threshold value for the programmable full flag, which defines when the signal is asserted. The threshold can be dynamically set in-circuit during reset.
PROG_FULL_THRESH_NEGATE	Input	Programmable Full Threshold Negate: This signal is used to set the lower threshold value for the programmable full flag, which defines when the signal is de-asserted. The threshold can be dynamically set in-circuit during reset.
INJECTSBITERR	Input	Injects a single bit error if the ECC feature is used on a Virtex-6 FPGA block RAM or built-in FIFO macro. For detailed information, see "Chapter 4, Designing with the Core," in the *FIFO Generator User Guide*.
INJECTDBITERR	Input	Injects a double bit error if the ECC feature is used on a Virtex-6 FPGA block RAM or built-in FIFO macro. For detailed information, see "Chapter 4, Designing with the Core," in the *FIFO Generator User Guide*.

Table 7 defines the read interface signals of a FIFO with independent clocks. Read interface signals are divided into required signals and optional signals, and all signals are synchronous to the read clock (RD_CLK).

Table 7: **Read Interface Signals for FIFOs with Independent Clocks**

Name	Direction	Description
Required		
RD_RST	Input	Read Reset: Synchronous to read clock. When asserted, initializes all internal pointers, flags and output registers of read clock domain.
RD_CLK	Input	Read Clock: All signals on the read domain are synchronous to this clock.
DOUT[M:0]	Output	Data Output: The output data bus is driven when reading the FIFO.
RD_EN	Input	Read Enable: If the FIFO is not empty, asserting this signal causes data to be read from the FIFO (output on DOUT).
EMPTY	Output	Empty Flag: When asserted, this signal indicates that the FIFO is empty. Read requests are ignored when the FIFO is empty, initiating a read while empty is non-destructive to the FIFO.
Optional		
ALMOST_EMPTY	Output	Almost Empty Flag: When asserted, this signal indicates that the FIFO is almost empty and one word remains in the FIFO.
PROG_EMPTY	Output	Programmable Empty: This signal is asserted when the number of words in the FIFO is less than or equal to the programmable threshold. It is de-asserted when the number of words in the FIFO exceeds the programmable threshold.

Table 7: **Read Interface Signals for FIFOs with Independent Clocks *(Cont'd)***

Name	Direction	Description
RD_DATA_COUNT [C:0]	Output	Read Data Count: This bus indicates the number of words available for reading in the FIFO. The count is guaranteed to never over-report the number of words available for reading, to ensure that the user does not underflow the FIFO. The exception to this behavior is when the read operation occurs at the rising edge of RD_CLK, that read operation is only reflected on RD_DATA_COUNT at the next rising clock edge. If C is less than log2(FIFO depth)-1, the bus is truncated by removing the least-significant bits.
VALID	Output	Valid: This signal indicates that valid data is available on the output bus (DOUT).
UNDERFLOW	Output	Underflow: Indicates that the read request (RD_EN) during the previous clock cycle was rejected because the FIFO is empty. Underflowing the FIFO is not destructive to the FIFO.
PROG_EMPTY_THRESH	Input	Programmable Empty Threshold: This signal is used to input the threshold value for the assertion and de-assertion of the programmable empty (PROG_EMPTY) flag. The threshold can be dynamically set in-circuit during reset. The user can either choose to set the assert and negate threshold to the same value (using PROG_EMPTY_THRESH), or the user can control these values independently (using PROG_EMPTY_THRESH_ASSERT and PROG_EMPTY_THRESH_NEGATE).
PROG_EMPTY_THRESH_ASSERT	Input	Programmable Empty Threshold Assert: This signal is used to set the lower threshold value for the programmable empty flag, which defines when the signal is asserted. The threshold can be dynamically set in-circuit during reset.
PROG_EMPTY_THRESH_NEGATE	Input	Programmable Empty Threshold Negate: This signal is used to set the upper threshold value for the programmable empty flag, which defines when the signal is de-asserted. The threshold can be dynamically set in-circuit during reset.
SBITERR	Output	Single Bit Error: Indicates that the ECC decoder detected and fixed a single-bit error on a Virtex-6 or Virtex-5 FPGA block RAM or built-in FIFO macro. For detailed information, see "Chapter 4, Designing with the Core," in the *FIFO Generator User Guide*.
DBITERR	Output	Double Bit Error: Indicates that the ECC decoder detected a double-bit error on a Virtex-6 or Virtex-5 FPGA block RAM or built-in FIFO macro and data in the FIFO core is corrupted. For detailed information, see "Chapter 4, Designing with the Core," in the *FIFO Generator User Guide*.

Interface Signals: FIFOs with Common Clock

Table 8 defines the interface signals of a FIFO with a common write and read clock and is divided into standard and optional interface signals. All signals (except asynchronous reset) are synchronous to the common clock (CLK). Users have the option to select synchronous or asynchronous reset for the distributed or block RAM FIFO implementation. See the *FIFO Generator User Guide* for specific information on reset requirements and behavior.

Table 8: **Interface Signals for FIFOs with a Common Clock**

Name	Direction	Description
		Required
RST	Input	Reset: An asynchronous reset that initializes all internal pointers and output registers.
SRST	Input	Synchronous Reset: A synchronous reset that initializes all internal pointers and output registers.
CLK	Input	Clock: All signals on the write and read domains are synchronous to this clock.
DIN[N:0]	Input	Data Input: The input data bus used when writing the FIFO.
WR_EN	Input	Write Enable: If the FIFO is not full, asserting this signal causes data (on DIN) to be written to the FIFO.
FULL	Output	Full Flag: When asserted, this signal indicates that the FIFO is full. Write requests are ignored when the FIFO is full, initiating a write when the FIFO is full is non-destructive to the contents of the FIFO.
DOUT[M:0]	Output	Data Output: The output data bus driven when reading the FIFO.
RD_EN	Input	Read Enable: If the FIFO is not empty, asserting this signal causes data to be read from the FIFO (output on DOUT).
EMPTY	Output	Empty Flag: When asserted, this signal indicates that the FIFO is empty. Read requests are ignored when the FIFO is empty, initiating a read while empty is non-destructive to the FIFO.
		Optional
DATA_COUNT [C:0]	Output	Data Count: This bus indicates the number of words stored in the FIFO. If C is less than log2(FIFO depth)-1, the bus is truncated by removing the least-significant bits.
ALMOST_FULL	Output	Almost Full: When asserted, this signal indicates that only one more write can be performed before the FIFO is full.
PROG_FULL	Output	Programmable Full: This signal is asserted when the number of words in the FIFO is greater than or equal to the assert threshold. It is deasserted when the number of words in the FIFO is less than the negate threshold.
WR_ACK	Output	Write Acknowledge: This signal indicates that a write request (WR_EN) during the prior clock cycle succeeded.
OVERFLOW	Output	Overflow: This signal indicates that a write request (WR_EN) during the prior clock cycle was rejected, because the FIFO is full. Overflowing the FIFO is non-destructive to the FIFO.

Table 8: **Interface Signals for FIFOs with a Common Clock *(Cont'd)***

Name	Direction	Description
PROG_FULL_THRESH	Input	Programmable Full Threshold: This signal is used to set the threshold value for the assertion and de-assertion of the programmable full (PROG_FULL) flag. The threshold can be dynamically set in-circuit during reset. The user can either choose to set the assert and negate threshold to the same value (using PROG_FULL_THRESH), or the user can control these values independently (using PROG_FULL_THRESH_ASSERT and PROG_FULL_THRESH_NEGATE).
PROG_FULL_THRESH_ASSERT	Input	Programmable Full Threshold Assert: This signal is used to set the upper threshold value for the programmable full flag, which defines when the signal is asserted. The threshold can be dynamically set in-circuit during reset.
PROG_FULL_THRESH_NEGATE	Input	Programmable Full Threshold Negate: This signal is used to set the lower threshold value for the programmable full flag, which defines when the signal is de-asserted. The threshold can be dynamically set in-circuit during reset.
ALMOST_EMPTY	Output	Almost Empty Flag: When asserted, this signal indicates that the FIFO is almost empty and one word remains in the FIFO.
PROG_EMPTY	Output	Programmable Empty: This signal is asserted after the number of words in the FIFO is less than or equal to the programmable threshold. It is de-asserted when the number of words in the FIFO exceeds the programmable threshold.
VALID	Output	Valid: This signal indicates that valid data is available on the output bus (DOUT).
UNDERFLOW	Output	Underflow: Indicates that read request (RD_EN) during the previous clock cycle was rejected because the FIFO is empty. Underflowing the FIFO is non-destructive to the FIFO.
PROG_EMPTY_THRESH	Input	Programmable Empty Threshold: This signal is used to set the threshold value for the assertion and de-assertion of the programmable empty (PROG_EMPTY) flag. The threshold can be dynamically set in-circuit during reset. The user can either choose to set the assert and negate threshold to the same value (using PROG_EMPTY_THRESH), or the user can control these values independently (using PROG_EMPTY_THRESH_ASSERT and PROG_EMPTY_THRESH_NEGATE).
PROG_EMPTY_THRESH_ASSERT	Input	Programmable Empty Threshold Assert: This signal is used to set the lower threshold value for the programmable empty flag, which defines when the signal is asserted. The threshold can be dynamically set in-circuit during reset.
PROG_EMPTY_THRESH_NEGATE	Input	Programmable Empty Threshold Negate: This signal is used to set the upper threshold value for the programmable empty flag, which defines when the signal is de-asserted. The threshold can be dynamically set in-circuit during reset.
SBITERR	Output	Single Bit Error: Indicates that the ECC decoder detected and fixed a single-bit error on a Virtex-6 or Virtex-5 FPGA block RAM or built-in FIFO macro. For detailed information, see "Chapter 4, Designing with the Core," in the *FIFO Generator User Guide*.

Table 8: **Interface Signals for FIFOs with a Common Clock *(Cont'd)***

Name	Direction	Description
DBITERR	Output	Double Bit Error: Indicates that the ECC decoder detected a double-bit error on a Virtex-6 or Virtex-5 FPGA block RAM or built-in FIFO macro and data in the FIFO core is corrupted. For detailed information, see "Chapter 4, Designing with the Core," in the *FIFO Generator User Guide*.
INJECTSBITERR	Input	Injects a single bit error if the ECC feature is used on a Virtex-6 FPGA block RAM or built-in FIFO macro. For detailed information, see "Chapter 4, Designing with the Core," in the *FIFO Generator User Guide*.
INJECTDBITERR	Input	Injects a double bit error if the ECC feature is used on a Virtex-6 FPGA block RAM or built-in FIFO macro. For detailed information, see "Chapter 4, Designing with the Core," in the *FIFO Generator User Guide*.

Port Summary

Table 9 describes all the FIFO Generator ports. For detailed information about any of the ports, see "Chapter 3, Core Architecture," in the *FIFO Generator User Guide*.

Table 9: **FIFO Generator Ports**

Port Name	Input or Output	Optional Port	Port Available	
			Independent Clocks	Common Clock
RST	I	Yes	Yes	Yes
SRST	I	Yes	No	Yes
CLK	I	No	No	Yes
DATA_COUNT[C:0]	O	Yes	No	Yes
Write Interface Signals				
WR_CLK	I	No	Yes	No
DIN[N:0]	I	No	Yes	Yes
WR_EN	I	No	Yes	Yes
FULL	O	No	Yes	Yes
ALMOST_FULL	O	Yes	Yes	Yes
PROG_FULL	O	Yes	Yes	Yes
WR_DATA_COUNT[D:0]	O	Yes	Yes	No
WR_ACK	O	Yes	Yes	Yes
OVERFLOW	O	Yes	Yes	Yes
PROG_FULL_THRESH	I	Yes	Yes	Yes
PROG_FULL_THRESH_ASSERT	I	Yes	Yes	Yes
PROG_FULL_THRESH_NEGATE	I	Yes	Yes	Yes
WR_RST	I	Yes	Yes	No

Table 9: **FIFO Generator Ports *(Cont'd)***

Port Name	Input or Output	Optional Port	Port Available	
			Independent Clocks	Common Clock
INJECTSBITERR	I	Yes	Yes	Yes
INJECTDBITERR	I	Yes	Yes	Yes
Read Interface Signals				
RD_CLK	I	No	Yes	No
DOUT[M:0]	O	No	Yes	Yes
RD_EN	I	No	Yes	Yes
EMPTY	O	No	Yes	Yes
ALMOST_EMPTY	O	Yes	Yes	Yes
PROG_EMPTY	O	Yes	Yes	Yes
RD_DATA_COUNT[C:0]	O	Yes	Yes	No
VALID	O	Yes	Yes	Yes
UNDERFLOW	O	Yes	Yes	Yes
PROG_EMPTY_THRESH	I	Yes	Yes	Yes
PROG_EMPTY_THRESH_ASSERT	I	Yes	Yes	Yes
PROG_EMPTY_THRESH_NEGATE	I	Yes	Yes	Yes
SBITERR	O	Yes	Yes	Yes
DBITERR	O	Yes	Yes	Yes
RD_RST	I	Yes	Yes	No

Resource Utilization and Performance

Performance and resource utilization for a FIFO varies depending on the configuration and features selected during core customization. The following tables show resource utilization data and maximum performance values for a variety of sample FIFO configurations.

The benchmarks were performed while adding two levels of registers on all inputs (except clock) and outputs having only the period constraints in the UCF. To achieve the performance shown in the following tables, ensure that all inputs to the FIFO are registered and that the outputs are not passed through many logic levels.

Note: The Shift Register FIFO is more suitable in terms of resource and performance compared to the Distributed Memory FIFO, where the depth of the FIFO is around 16 or 32.

Table 10 identifies the results for a FIFO configured without optional features. Benchmarks were performed using Virtex-4 (XC4VLX15-FF668-10), Virtex-5 (XC5VLX50-FF324-1), Virtex-6 (XC6VLX760-FF1760-1) and Spartan-6 (XC6SLX150T-FGG484-2) FPGAs.

Table 10: **Benchmarks: FIFO Configured without Optional Features**

FIFO Type	Depth x Width	FPGA Family	Performance (MHz)	Resources				
				LUTs	FFs	Block RAM	Shift Register	Distributed RAM
Synchronous FIFO (Block RAM)	512 x 16	Virtex-6	325	48	50	1	0	0
		Virtex-5	300	44	50	1	0	0
		Virtex-4	250	29	50	1	0	0
		Spartan-6	175	46	50	1	0	0
	4096 x 16	Virtex-6	325	56	62	2	0	0
		Virtex-5	325	54	62	2	0	0
		Virtex-4	275	33	62	4	0	0
		Spartan-6	200	57	62	4	0	0
Synchronous FIFO (Distributed RAM)	64 x 16	Virtex-6	400	25	54	0	0	22
		Virtex-5	400	27	54	0	0	22
		Virtex-4	350	70	64	0	0	128
		Spartan-6	175	27	54	0	0	22
	512 x 16	Virtex-6	325	109	66	0	0	176
		Virtex-5	300	83	66	0	0	256
		Virtex-4	300	326	202	0	0	1024
		Spartan-6	150	104	66	0	0	176
Independent Clocks FIFO (Block RAM)	512 x 16	Virtex-6	325	79	132	1	0	0
		Virtex-5	325	75	132	1	0	0
		Virtex-4	300	63	132	1	0	0
		Spartan-6	175	70	132	1	0	0
	4096 x 16	Virtex-6	325	109	171	2	0	0
		Virtex-5	300	106	171	2	0	0
		Virtex-4	275	91	171	4	0	0
		Spartan-6	200	98	171	4	0	0
Independent Clocks FIFO (Distributed RAM	64 x 16	Virtex-6	425	121	183	0	0	22
		Virtex-5	400	46	109	0	0	22
		Virtex-4	350	92	112	0	0	128
		Spartan-6	200	42	109	0	0	22
	512 x 16	Virtex-6	350	135	148	0	0	176
		Virtex-5	300	114	148	0	0	256
		Virtex-4	200	352	280	0	0	1024
		Spartan-6	150	129	148	0	0	176

Table 10: **Benchmarks: FIFO Configured without Optional Features *(Cont'd)***

FIFO Type	Depth x Width	FPGA Family	Performance (MHz)	Resources				
				LUTs	FFs	Block RAM	Shift Register	Distributed RAM
Shift Register FIFO(1)	64 x 16	Virtex-5	425	56	44	0	32	0
		Virtex-4	300	66	44	0	64	0
		Spartan-6	175	55	44	0	48	0
	512 x 16	Virtex-5	275	134	53	0	256	0
		Virtex-4	225	324	57	0	512	0
		Spartan-6	75	339	56	0	496	0

1. Virtex-6 benchmarking data for the Shift Register FIFO will be available in the ISE 12.3 release.

Table 11 provides results for FIFOs configured with multiple programmable thresholds. Benchmarks were performed using Virtex-4 (XC4VLX15-FF668-10), Virtex-5 (XC5VLX50-FF324-1), Virtex-6 (XC6VLX760-FF1760-1) and Spartan-6 (XC6SLX150T-FGG484-2) FPGAs.

Table 11: **Benchmarks: FIFO Configured with Multiple Programmable Thresholds**

FIFO Type	Depth x Width	FPGA Family	Performance (MHz)	Resources				
				LUTs	FFs	Block RAM	Shift Register	Distributed RAM
Synchronous FIFO (Block RAM)	512 x 16	Virtex-6	325	80	74	1	0	0
		Virtex-5	325	75	74	1	0	0
		Virtex-4	250	66	74	1	0	0
		Spartan-6	200	78	74	1	0	0
	4096 x 16	Virtex-6	325	94	92	2	0	0
		Virtex-5	300	94	92	2	0	0
		Virtex-4	250	77	92	4	0	0
		Spartan-6	175	93	92	4	0	0
Synchronous FIFO (Distributed RAM)	64 x 16	Virtex-6	425	47	72	0	0	22
		Virtex-5	400	48	72	0	0	22
		Virtex-4	300	97	73	0	0	128
		Spartan-6	175	45	72	0	0	22
	512 x 16	Virtex-6	300	139	90	0	0	176
		Virtex-5	300	114	90	0	0	176
		Virtex-4	225	358	226	0	0	1024
		Spartan-6	150	132	90	0	0	176

Table 11: **Benchmarks: FIFO Configured with Multiple Programmable Thresholds *(Cont'd)***

FIFO Type	Depth x Width	FPGA Family	Performance (MHz)	Resources				
				LUTs	FFs	Block RAM	Shift Register	Distributed RAM
Independent Clocks FIFO (Block RAM)	512 x 16	Virtex-6	325	110	153	1	0	0
		Virtex-5	300	103	153	1	0	0
		Virtex-4	300	97	153	1	0	0
		Spartan-6	200	101	153	1	0	0
	4096 x 16	Virtex-6	325	149	198	2	0	0
		Virtex-5	325	144	198	2	0	0
		Virtex-4	275	125	198	4	0	0
		Spartan-6	200	141	198	4	0	0
Independent Clocks FIFO (Distributed RAM	64 x 16	Virtex-6	450	65	124	0	0	22
		Virtex-5	400	66	124	0	0	22
		Virtex-4	325	116	128	0	0	128
		Spartan-6	200	63	124	0	0	22
	512 x 16	Virtex-6	350	148	169	0	0	176
		Virtex-5	275	142	169	0	0	176
		Virtex-4	225	388	301	0	0	1024
		Spartan-6	150	153	169	0	0	176
Shift Register FIFO(1)	64 x 16	Virtex-5	400	78	60	0	32	0
		Virtex-4	300	121	60	0	64	0
		Spartan-6	175	77	60	0	48	0
	512 x 16	Virtex-5	275	167	75	0	256	0
		Virtex-4	225	374	79	0	512	0
		Spartan-6	75	370	78	0	496	0

1. Virtex-6 benchmarking data for the Shift Register FIFO will be available in the ISE 12.3 release.

Table 12 provides results for FIFOs configured to use the Virtex-5 FPGA built-in FIFO. The benchmarks were performed using a Virtex-5 (XC5VLX50-FF324-1) and Virtex-6 (XC6VLX760-FF1760-1) FPGA.

Table 12: **Benchmarks: FIFO Configured with Virtex-5 and Virtex-6 FIFO36 Resources**

FIFO Type	Depth x Width	FPGA Family	Read Mode	Performance (MHz)	LUTs	FFs	FIFO36
Synchronous FIFO36 (basic)	512 x 72	Virtex-5	Standard	300	0	2	1
			FWFT	300	2	4	1
		Virtex-6	Standard	325	1	2	1
			FWFT	325	4	5	1
	16k x 8(1)	Virtex-5	Standard	275	10	6	4
			FWFT	275	13	10	4
Synchronous FIFO36 (with handshaking)	512 x 72	Virtex-5	Standard	300	2	6	1
			FWFT	300	5	6	1
		Virtex-6	Standard	325	4	6	1
			FWFT	325	8	8	1
	16k x 8(1)	Virtex-5	Standard	300	12	12	4
			FWFT	300	16	13	4
Independent Clocks FIFO36 (basic)	512 x 72	Virtex-5	Standard	440	0	2	1
			FWFT	440	0	2	1
		Virtex-6	Standard	450	1	3	1
			FWFT	450	1	3	1
	16k x 8	Virtex-5	Standard	300	6	2	4
			FWFT	300	6	2	4
		Virtex-6	Standard	350	6	4	4
			FWFT	350	6	7	4
Independent Clocks FIFO36 (with handshaking)	512 x 72	Virtex-5	Standard	440	2	6	1
			FWFT	440	2	3	1
		Virtex-6	Standard	450	4	7	1
			FWFT	450	3	4	1
	16k x 8	Virtex-5	Standard	280	8	8	4
			FWFT	320	9	5	4
		Virtex-6	Standard	325	10	10	4
			FWFT	325	9	7	4

1. Virtex-6 benchmarking data will be available in the ISE 12.3 release.

Table 13 provides results for FIFOs configured to use the Virtex-4 built-in FIFO with patch. The benchmarks were performed using a Virtex-4 (XC4VLX15-FF668-10) FPGA.

Table 13: **Benchmarks: FIFO Configured with Virtex-4 FIFO16 Patch**

FIFO Type	Depth x Width	Clock Ratios	Performance (MHz)	LUTs	FFs	Distributed RAMs	FIFO16s
Built-in FIFO (basic)	512x36	WR_CLK ≥ RD_CLK	250	115	264	72	1
		RD_CLK > WR_CLK	225	118	269	72	1
Built-in FIFO (Handshaking)	512x36	WR_CLK ≥ RD_CLK	250	117	277	72	1
		RD_CLK > WR_CLK	225	121	282	72	1

Supplemental Information

The following sections provide additional information about working with the FIFO Generator core.

Compatibility with Older FIFO Cores

The FIFO Generator Migration Kit can be used to migrate from legacy FIFO cores (Asynchronous FIFO and Synchronous FIFO cores) and older versions of the FIFO Generator core to the latest version of the FIFO Generator core.

SIM Parameters

Table 14 defines the SIM parameters used to specify the configuration of the core. These parameters are only used while instantiating the core in HDL manually or while calling the CORE Generator dynamically. This parameter list does not apply a core generated using the CORE Generator GUI.

Table 14: **SIM Parameters**

	SIM Parameter	Type	Description
1	C_COMMON_CLOCK	Integer	• 0: Independent Clock • 1: Common Clock
2	C_DATA_COUNT_WIDTH	Integer	Width of DATA_COUNT bus (1 – 23)
3	C_DIN_WIDTH	Integer	Width of DIN bus (1 – 1024)
4	C_DOUT_RST_VAL	String	Reset value of DOUT Hexadecimal value, 0 - 'F's equal to C_DOUT_WIDTH
5	C_DOUT_WIDTH	Integer	Width of DOUT bus (1 – 1024)
6	C_ENABLE_RST_SYNC	Integer	• 0: Do not synchronize the reset (WR_RST/RD_RST is directly used, available only for independent clock) • 1: Synchronize the reset
7	C_ERROR_INJECTION_TYPE	Integer	• 0: No error injection • 1: Single bit error injection • 2: Double bit error injection • 3: Single and double bit error injection
8	C_FAMILY	String	Device family (Ex.Virtex-5,Virtex-6, etc)
9	C_FULL_FLAGS_RST_VAL	Integer	Full flags rst val (0 or 1)

Table 14: **SIM Parameters *(Cont'd)***

	SIM Parameter	Type	Description
10	C_HAS_ALMOST_EMPTY	Integer	• 0: Core does not have ALMOST_EMPTY flag • 1: Core has ALMOST_EMPTY flag
11	C_HAS_ALMOST_FULL	Integer	• 0: Core does not have ALMOST_FULL flag • 1: Core has ALMOST_ FULL flag
12	C_HAS_DATA_COUNT	Integer	• 0: Core does not have DATA_COUNT bus • 1: Core has DATA_COUNT bus
13	C_HAS_OVERFLOW	Integer	• 0: Core does not have OVERFLOW flag • 1: Core has OVERFLOW flag
14	C_HAS_RD_DATA_COUNT	Integer	• 0: Core does not have RD_DATA_COUNT bus • 1: Core has RD_DATA_COUNT bus
15	C_HAS_RST	Integer	• 0: Core does not have asynchronous reset (RST) • 1: Core has asynchronous reset (RST)
16	C_HAS_SRST	Integer	• 0: Core does not have synchronous reset (SRST) • 1: Core has synchronous reset (SRST)
17	C_HAS_UNDERFLOW	Integer	• 0: Core does not have UNDERFLOW flag • 1: Core has UNDERFLOW flag
18	C_HAS_VALID	Integer	• 0: Core does not have VALID flag • 1: Core has VALID flag
19	C_HAS_WR_ACK	Integer	• 0: Core does not have WR_ACK flag • 1: Core has WR_ACK flag
20	C_HAS_WR_DATA_COUNT	Integer	• 0: Core does not have WR_DATA_COUNT bus • 1: Core has WR_DATA_COUNT bus
21	C_IMPLEMENTATION_TYPE	Integer	• 0: Common-Clock Block RAM/Distributed RAM FIFO • 1: Common-Clock Shift RAM FIFO • 2: Independent Clocks Block RAM/Distributed RAM FIFO • 3: Virtex-4 Built-in FIFO • 4: Virtex-5 Built-in FIFO • 5: Virtex-6 Built-in FIFO
22	C_MEMORY_TYPE	Integer	• 1: Block RAM • 2: Distributed RAM • 3: Shift RAM • 4: Built-in FIFO
23	C_MSGON_VAL	Integer	• 0: Disables timing violation on cross clock domain registers • 1: Enables timing violation on cross clock domain registers
24	C_OVERFLOW_LOW	Integer	• 0: OVERFLOW active high • 1: OVERFLOW active low
25	C_PRELOAD_LATENCY	Integer	• 0: First-Word Fall-Through with or without Embedded Register • 1: Standard FIFO without Embedded Register • 2: Standard FIFO with Embedded Register

Table 14: **SIM Parameters *(Cont'd)***

	SIM Parameter	Type	Description
26	C_PRELOAD_REGS	Integer	• 0: Standard FIFO without Embedded Register • 1: Standard FIFO with Embedded Register or First-Word Fall-Through with or without Embedded Register
27	C_PRIM_FIFO_TYPE	String	Primitive used to build a FIFO (Ex. "512x36")
28	C_PROG_EMPTY_THRESH_ASSERT_VAL	Integer	PROG_EMPTY assert threshold(1)
29	C_PROG_EMPTY_THRESH_NEGATE_VAL	Integer	PROG_EMPTY negate threshold(1)
30	C_PROG_EMPTY_TYPE	Integer	• 0: No programmable empty • 1: Single programmable empty thresh constant • 2: Multiple programmable empty thresh constants • 3: Single programmable empty thresh input • 4: Multiple programmable empty thresh inputs
31	C_PROG_FULL_THRESH_ASSERT_VAL	Integer	PROG_FULL assert threshold(1)
32	C_PROG_FULL_THRESH_NEGATE_VAL	Integer	PROG_FULL negate threshold(1)
33	C_PROG_FULL_TYPE	Integer	• 0: No programmable full • 1: Single programmable full thresh constant • 2: Multiple programmable full thresh constants • 3: Single programmable full thresh input • 4: Multiple programmable full thresh inputs
34	C_RD_DATA_COUNT_WIDTH	Integer	Width of RD_DATA_COUNT bus (1 - 23)
35	C_RD_DEPTH	Integer	Depth of read interface (16 – 4194305)
36	C_RD_FREQ	Integer	Read clock frequency (1 MHz - 1000 MHz)
37	C_RD_PNTR_WIDTH	Integer	log2(C_RD_DEPTH)
38	C_UNDERFLOW_LOW	Integer	• 0: UNDERFLOW active high • 1: UNDERFLOW active low
39	C_USE_DOUT_RST	Integer	• 0: Does not reset DOUT on RST • 1: Resets DOUT on RST
40	C_USE_ECC	Integer	• 0: Does not use ECC feature • 1: Uses ECC feature
41	C_USE_EMBEDDED_REG	Integer	• 0: Does not use BRAM embedded output register • 1: Uses BRAM embedded output register
42	C_USE_FWFT_DATA_COUNT	Integer	• 0: Does not use extra logic for FWFT data count • 1: Uses extra logic for FWFT data count
43	C_VALID_LOW	Integer	• 0: VALID active high • 1: VALID active low
44	C_WR_ACK_LOW	Integer	• 0: WR_ACK active high • 1: WR_ACK active low
45	C_WR_DATA_COUNT_WIDTH	Integer	Width of WR_DATA_COUNT bus (1 – 23)
46	C_WR_DEPTH	Integer	Depth of write interface (16 – 4194305)
47	C_WR_FREQ	Integer	Write clock frequency (1 MHz - 1000 MHz)
48	C_WR_PNTR_WIDTH	Integer	log2(C_WR_DEPTH)

1. See the FIFO Generator GUI for the allowable range of values.

Verification

Xilinx has verified the FIFO Generator core in a proprietary test environment, using an internally developed bus functional model. Tens of thousands of test vectors were generated and verified, including both valid and invalid write and read data accesses.

References

1. *FIFO Generator User Guide* (UG175)

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Ordering Information

This Xilinx LogiCORE IP module is included at no additional charge with the Xilinx ISE® Design Suite software and is provided under the terms of the Xilinx End User License Agreement. The core is generated using the Xilinx ISE CORE Generator™ software, which is a standard component of the Xilinx ISE software.

For more information, please visit the FIFO Generator core page.

Information about additional LogiCORE IP modules can be found on the Xilinx.com Intellectual Property page. Contact your local Xilinx sales representative for pricing and availability.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
4/23/04	1.0	Initial Xilinx release.
5/21/04	1.1	Support for Virtex-4 built-in FIFO implementation.
11/11/04	2.0	Updated for Xilinx software v6.3i.
04/28/05	2.1	The original product specification has been divided into two documents - a data sheet and a user guide. The document has also been updated to indicate core support of first-word fall-through feature and Xilinx software v7.1i.
8/31/05	2.2	Updated Xilinx v7.1i to SP3, removed references to first-word fall-through as new in this release. Updated basic FIFO benchmark value to reflect increased performance.
1/18/06	2.3	Minor updates for release v2.3, advanced ISE support to 8.1i.
7/13/06	3.0	Added support for Virtex-5, ISE to v8.2i, core version to 3.1
9/21/06	4.0	Core version updated to 3.2, support for Spartan-3A added to Facts table.
2/15/07	5.0	Updates to Xilinx tools 9.1i, core version 3.3.
4/02/07	5.5	Added support for Spartan-3A DSP devices, upgraded Cadence IUS version to 5.7
8/8/07	5.6	Updated to Xilinx tools v9.2i; Cadence IUS v5.8.
10/10/07	6.0	Updated for IP2 Jade Minor release.
3/24/08	7.0	Updated core to version 4.3; Xilinx tools to 10.1.
9/19/08	8.0	Updated core to version 4.4; Xilinx tools 10.1, SP3.
12/17/08	8.0.1	Early access documentation.
4/24/09	9.0	Updated core to version 5.1 and Xilinx tools to version 11.1.
6/24/09	10.0	Updated core to version 5.2 and Xilinx tools to version 11.2.
6/24/09	10.1	Updated "Resource Utilization and Performance," page 14.
9/16/09	11.0	Updated core to version 5.3 and Xilinx tools to version 11.3.
4/19/10	12.0	Updated core to version 6.1 and Xilinx tools to version 12.1.
7/23/10	13.0	Updated core to version 6.2 and Xilinx tools to version 12.2.

Notice of Disclaimer