FIFO Generator
(Beta Release)

Version 0.1

August 1, 2023

Copyright

Trademarks

Disclaimers

Contents

IP Summary

Introduction

The First-In-First-Out (FIFO) principle is a foundational concept extensively used in technical systems to effectively manage item prioritization based on their chronological order of arrival. This essential technique finds application in inventory management, data structures, and computing algorithms, ensuring optimal efficiency and maintaining order in various processes.

In data structures, FIFO is achieved through the implementation of a queue, where the first item inserted is the first to be removed. This approach maintains a "first in, first out" order for data retrieval and storage.

In computing algorithms, FIFO is used in process scheduling to ensure equitable execution and avoid process starvation. The principle guarantees that the first process entering the system is the first one to be executed, promoting fairness in system operations.

By adhering to the FIFO principle, technical systems can streamline processes, optimize resource utilization, and achieve systematic organization of data and tasks. This manual provides a comprehensive understanding of FIFO's implementation in Raptor Design Suite and all the supported features and capabilities, empowering users to harness its benefits across various technical domains.

Features

- Configurable FIFO Depth from 2 - 32768.

- Configurable Data Width from 1 - 128 bits.

- Support for Synchronous and Asynchronous Clocks.

- Support for programmable flag that indicates when the FIFO is empty or full.

- Support for implementation on either Block RAM or Distributed Memory.

- Configurable support for first-word-fall-through.

- Native standard FIFO interface.

- Built-in Status Indicators.

Overview

FIFO Generator

The IP Generator for FIFO (First-In-First-Out) is a specialized hardware module designed to facilitate the seamless integration of FIFO functionality into digital circuits and systems. It supports various different configuration features such as First-Word-Fall-Through and Asynchronous Write and Reads. First-Word-Fall-Through enables the reading device the know in advance what the next chunk of data looks like without issuing a pop command, and hence increasing the robustness and data integrity of the FIFO. Asynchronous Reads and Writes enable the FIFO to work in system with different clock domains that require synchronization between the read and write pointers. This synchronization is achieved via a pair of Flip Flop Synchronizers. These synchroziners ensure that the meta-stable condition does not occur while working in a two clock domain system while also making sure that the FIFO works at the highest efficiency. There is a couple of programmable empty and programmable full signals that the user can modify accordingly. This provides a greater level of control combined with the usual signals of Empty and Full. The option to select the implementation on either Block RAM or Distributed Memory makes the FIFO generator a highly efficient and FPGA optimized IP Core. A block diagram for a top level FIFO generated from the FIFO Generator IP is shown in Figure 1.
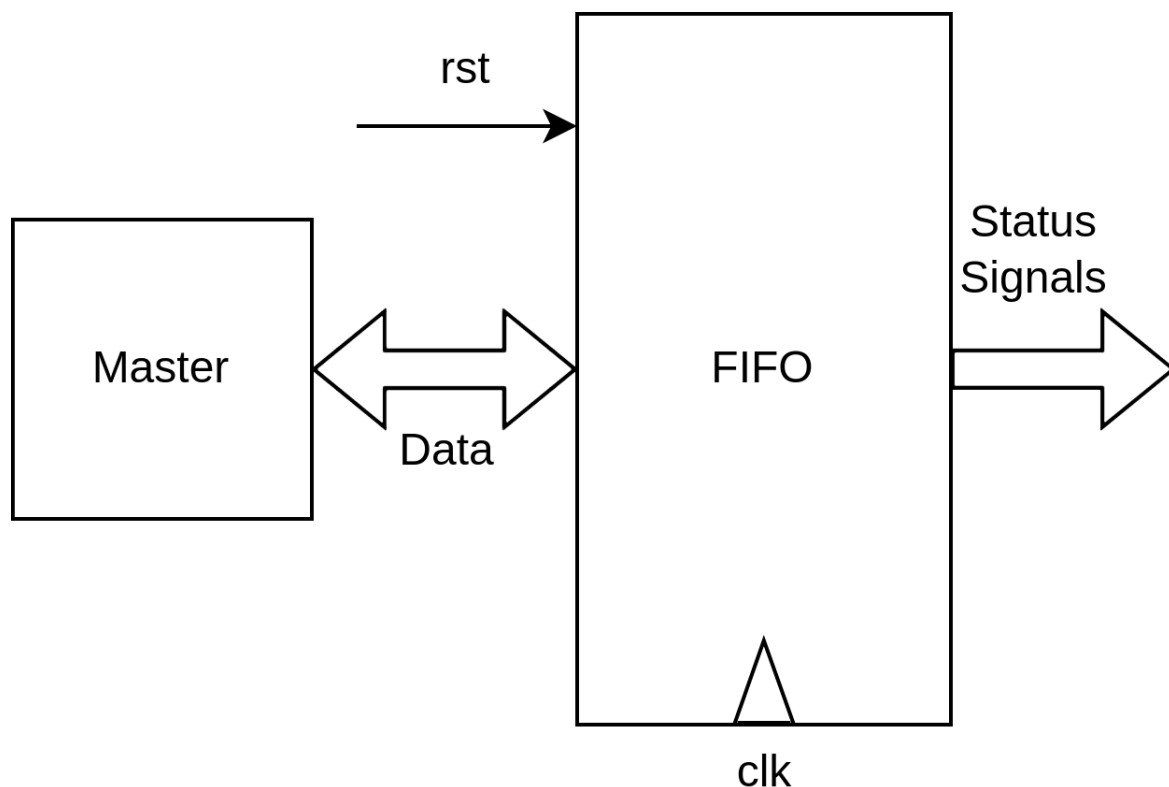

Figure 1: FIFO Block Diagram

IP Specification

The FIFO Generator is armed with advanced algorithms making the FIFO in a circular buffer ensuring that the no empty place is gone unnoticed. The synchronous pointers work on binary logic while the asynchronous pointers work on a gray code encoding logic so that the synchronizers can work without creating a meta stable condition. Because of this reason, the configurable depth for the Distributed RAM in the asynchronous mode is not continuous but rather in the exponents of 2. This is to ensure that the gray encoding logic does not break. This limitation is mitigated in the Block RAM design of the asynchronous FIFO where the pointers are converted back to binary before giving them to their respective counters and hence the configurable depth is continuous from 3 - 32768.

By giving the support for both Block RAM and Distributed Memory implementations, the user is able to choose between mapping the FIFO on the BRAM or by utilizing logic / LUTs of the FPGA board. This makes for an extremely versatile FIFO generator IP catering to a vast variety of different application needs. Since the reset is a synchronous one, the IP generated requires a cycle between the read and write operations to ensure that no data is being lost in the transitional cycles. A more detailed description of the internal workings of the FIFO can be read from here, the publication of **Clifford E. Cummings** in his publication titled **"Simulation and Synthesis Techniques for Asynchronous FIFO Design"**. An internal macro block diagram for the FIFO can be seen in Figure 2 that is referenced from the publication of Clifford E. Cummings linked earlier.
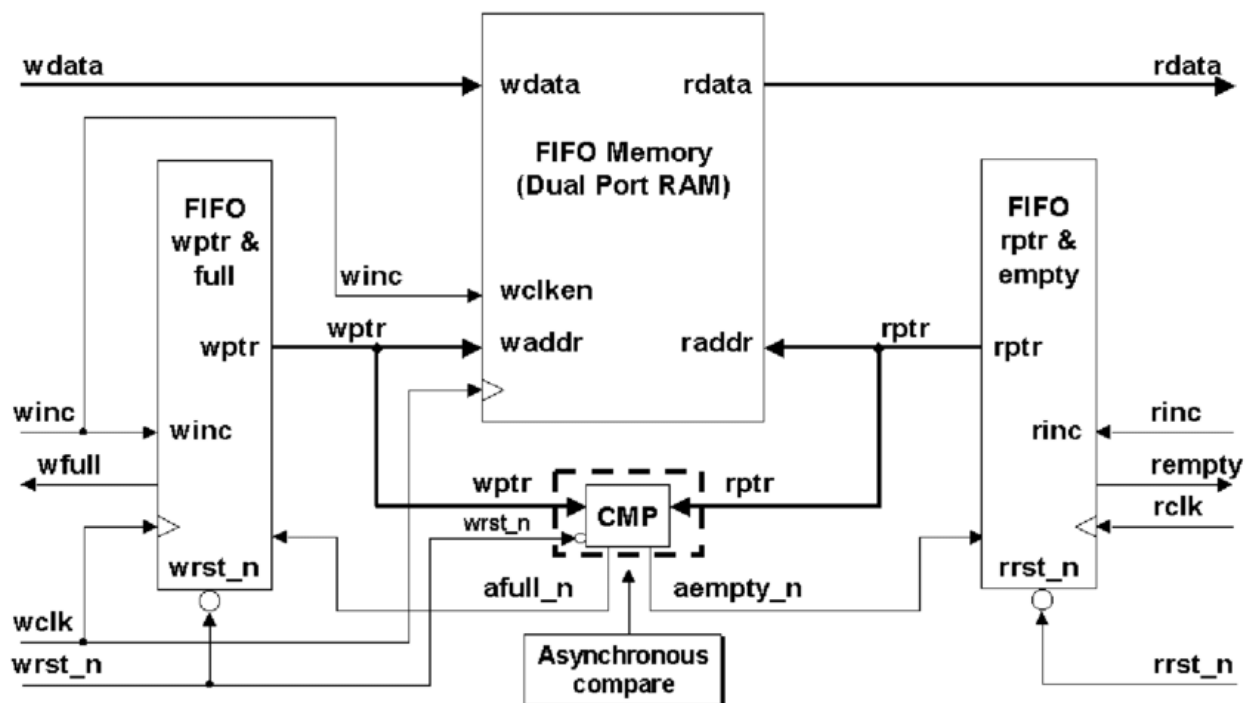

Figure 2: FIFO Internal Diagram

Synchronous versus Asynchronous Clock

Synchronous clock refers to a clock signal that is shared among all the memory blocks in the device. In contrast, an Asynchronous clock refers to a clock signal that is unique to each memory block.

- The use of a Synchronous clock in memory blocks has several advantages. First, it simplifies the design and implementation of the memory blocks as there is only one clock signal to operate. Secondly, it ensures that all the memory blocks are synchronized, which is critical for high-speed data transfer and avoiding errors. However, the disadvantage of using a Synchronous clock is that it may limit the flexibility of the design, as all the memory blocks must operate at the same clock frequency.

- The use of an ASynchronous clock in memory blocks allows for more flexibility in the design, as each memory block can operate at a different clock frequency. This can be useful in designs that require multiple memory blocks with different timing requirements. However, the disadvantage of using an Asynchronous clock is that it increases the complexity of the design, as each clock signal must be operated independently, and timing issues may arise.
Overall, the choice between using a Synchronous clock or an Asynchronous clock in memories using FIFO Generator depends on the specific requirements of the design. If the design requires all memory blocks to operate at the same frequency, a Synchronous clock may be the best choice. However, if the design requires flexibility in timing requirements, an Asynchronous clock may be more appropriate.

Block RAM versus Distributed Memory

Two types of memory blocks can be generated with FIFO Generator, i.e., Block RAM and Distributed RAM. Block RAM is a large, rectangular array of memory cells optimized for high-speed access, while Distributed RAM is a flexible set of smaller, distributed memory cells that are suitable for storing small chunks of data. The choice between these two types of memory blocks depends on the specific requirements of the design.

Standards

The FIFO Generator soft IP supports the native interface with the standard DATA_IN and DATA_OUT ports along with the supporting clocks and reset signals in the port list.

IP Support Details

The Table 1 gives the support details for FIFO Generator.

Compliance		IP Resources					Tool Flow		
Device	Interface	Source Files	Constraint File	Testbench	Simulation Model	Software Driver	Analyze and Elaboration	Simulation	Synthesis
GEMINI	Native	Verilog	-	Verilog	VVP	Iverilog	Raptor (Verific)	Raptor (Icarus)	Raptor

Table 1: IP Details

Parameters

Table 2 lists the parameters of the FIFO Generator.

Parameter	Values	Default Value	Description
DEPTH	3 - 32768	1024	FIFO Depth
DATA WIDTH	1 - 128	8	FIFO Write / Read Data Width
FULL VALUE	2 - (DEPTH - 1)	2	Programmable Full Value
EMPTY VALUE	1 - (DEPTH - 1)	1	Programmable Empty Value
SYNCHRONOUS	0 / 1	1	Synchronous / Asynchronous Implementation
FIRST WORD FALL THROUGH	0 / 1	0	First Word Fall Through Support
FULL THRESHOLD	0 / 1	0	Programmable Full Enable
EMPTY THRESHOLD	0 / 1	0	Programmable Empty Enable
BRAM	0 / 1	1	Block / Distributed RAM

Table 2: Parameters

Note: Both the FULL VALUE and EMPTY VALUE parameters are dependent on the DEPTH of the FIFO and are configured accordingly.

Port List

Table 3 lists the top interface ports of the FIFO Generator.

Signal Name	I / O	Description
din {DATA_WIDTH}	I	Data Input
dout {DATA_WIDTH}	O	Data Output
rst	I	Reset
wr_en	I	Write Enable
rd_en	I	Read Enable
full	O	FIFO Full
empty	O	FIFO Empty
underflow	O	FIFO Underflow
overflow	O	FIFO Overflow
prog_full	O	FIFO Programmable Full
prog_empty	O	FIFO Programmable Empty
rd_clk	I	Read Clock for Asynchronous Operation
wrt_clk	I	Write Clock for Asynchronous Operation
clk	I	Common Clock for Synchronous Operation

Table 3: FIFO Generator Interface

Note:

- *DATA_WIDTH is the bit-width of the data bus of the FIFO generated.*

- *prog_full and prog_empty depends on the FULL THRESHOLD and EMPTY THRESHOLD parameters respectively.*

- *rd_clk and wrt_clk depends on SYNCHRONOUS parameter being off, otherwise clk would be in the portlist for SYNCHRONOUS mode.*

Resource Utilization

The parameters for computing the maximum and minimum resource utilization are given in Table 4, remaining parameters have been kept at their default values.

Tool	Raptor Design Suite				
FPGA Device	GEMINI				
Configuration				**Resource Utilization**	
	Options	**Configuration**	**Resources**	**Utilized**	
Minimum Resource	DATA WIDTH	1	FIFO18K	1	
	DEPTH	3			
	FULL THRESHOLD	0	LUTs	4	
	EMPTY THRESHOLD	0			
	BRAM	1	Registers	2	
	SYNCHRONOUS	1			
	FIRST WORD FALL THROUGH	0			
Maximum Resource	**Options**	**Configuration**	**Resources**	**Utilized**	
	DATA WIDTH	128	FIFO36K	128	
	DEPTH	32768			
	FULL VALUE	32500	LUTs	2492	
	EMPTY VALUE	500			
	BRAM	1	Registers	294	
	SYNCHRONOUS	0			
	FIRST WORD FALL THROUGH	1			

Table 4: Resource Utilization

Design Flow

IP Customization and Generation

FIFO Generator IP core is a part of the Raptor Design Suite Software. A customized FIFO IP can be generated from the Raptor's IP configurator window as shown in Figure 3.
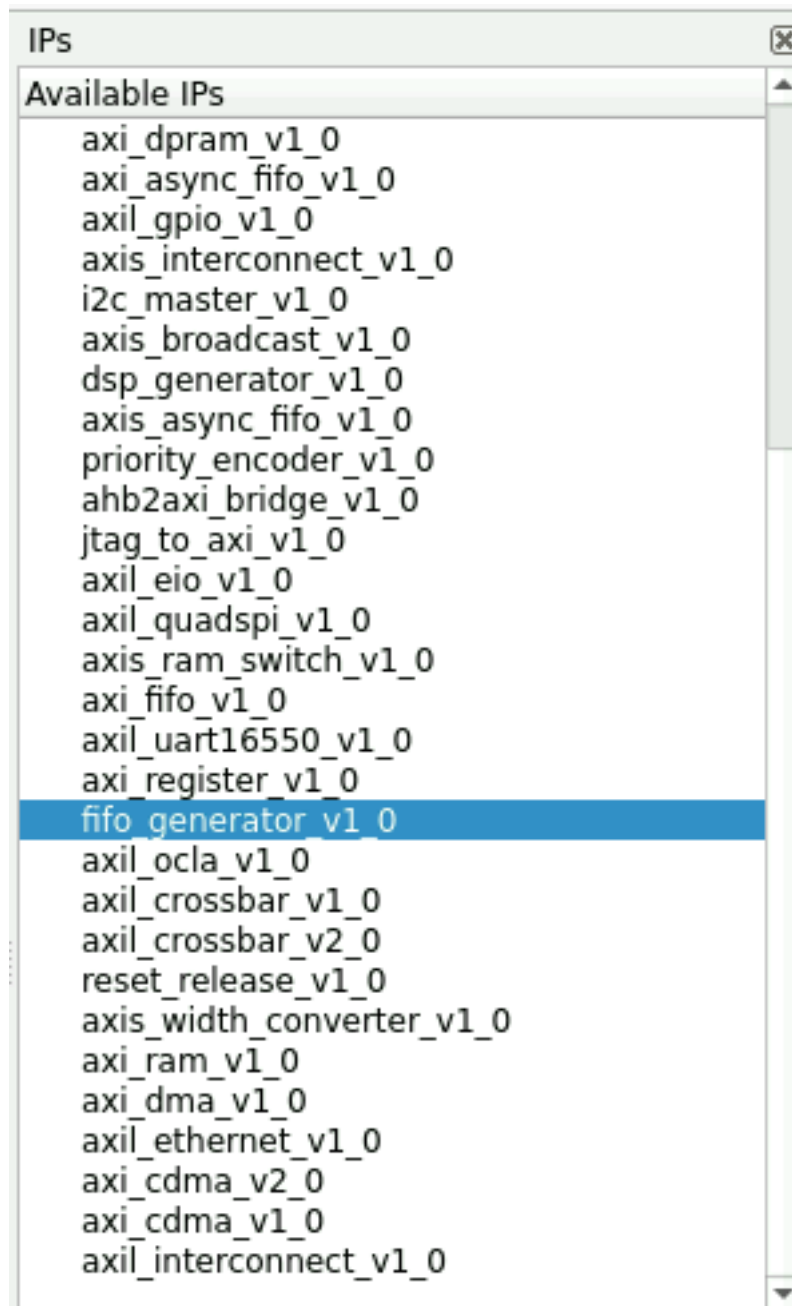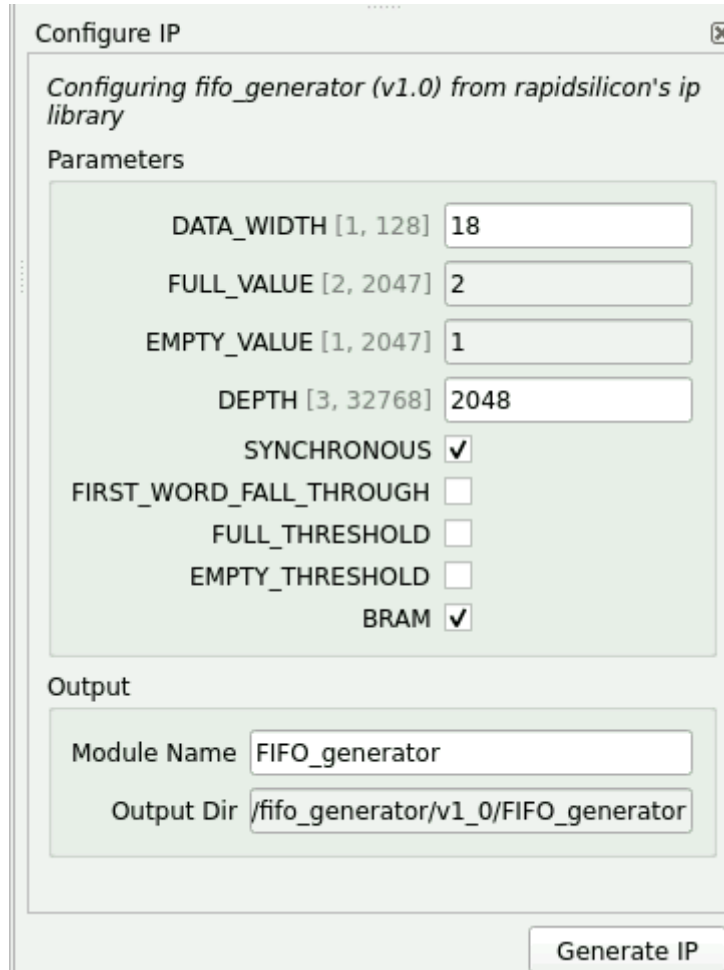

Figure 3: IP list

Parameters Customization

From the IP configuration window, the parameters of FIFO Generator can be configured and IP features can be enabled for generating a FIFO IP core that suits the user application requirement as shown in Figure 4. After IP Customization, the generated IP is made available to the user to be used in applications.

Figure 4: IP Configuration

Synthesis and PR

Raptor Suite is armed with tools for Synthesis along with Post and Route capabilities and the generated post-synthesis and post-route and place net-lists can be viewed and analyzed from within the Raptor. The generated bit-stream can then be uploaded on an FPGA device to be utilized in hardware applications.

Test Bench

The FIFO IP, based on Verilog HDL, can be stimulated by any number of industry standard means. These may include simple Verilog test benches or stimulating the DSP via an OS or via bare-metal firmwares. The bundled test-bench for this IP is a Verilog based testbench that can manipulated according to the configuration of the generated FIFO IP. After the generation of the IP, the source files and the simulation files are made available to the user along with the steps to simulate it via the bundled simulator. To make sure that the generated FIFO IP works as intended, the testbench compares the output of the generated FIFO IP to that of a basic memory array and hence makes sure that the generated IP works as intended. A sample waveform can be seen in the Figure 5.
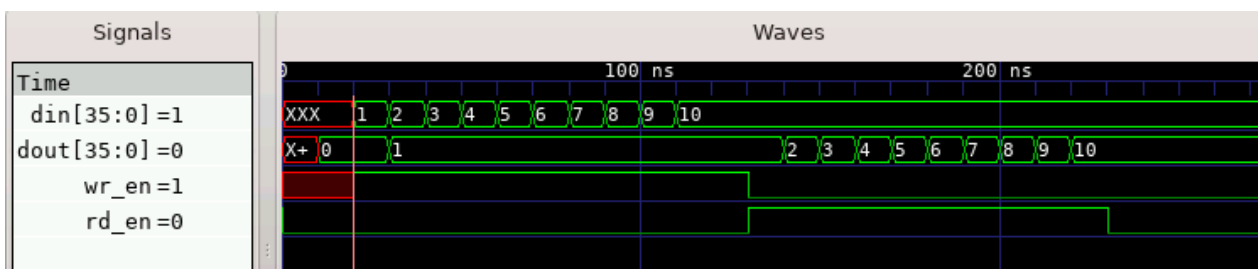
Figure 5: Test Results

Release

Release History

Date	Version	Revisions
August 1, 2023	0.1	Initial version FIFO Generator User Guide Document