RapidSilicon

FIFO Generator (Beta Release)

Version 0.1

December 5, 2023

Contents

IP Summary

Introduction

The FIFO Generator IP offers a superior alternative to custom Verilog-based designs, streamlining the development of high-quality FIFOs. It incorporates years of expertise, best practices, and optimizations for easy integration.
Beyond fundamental FIFO operations, this IP boasts configurable asymmetric data widths, adjustable depths, and synchronous/asynchronous clock domain crossing options. This flexibility empowers designers to tailor the FIFO Generator IP to specific requirements, making it indispensable across a wide range of applications.
Designers choose this IP to overcome challenges in data flow and synchronization. Its pre-established architecture expedites development, sparing engineers from intricate custom design nuances. The IP's versatility is evident in applications, from complex communication protocols to intricate memory management, prioritizing data integrity and synchronization. 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, empowering users across various technical domains.

Features

- Configurable FIFO Depth from 3 - 523264.
- Configurable Data Width from 1 - 1024 bits.
- Support for Synchronous and Asynchronous Clocks.
- Support for programmable flags that indicates when the FIFO is empty and full.
- Support for implementation on either Built-in FIFO or Distributed Memory.
- Support for First-Word-Fall-Through or Standard Mode.
- Native standard FIFO interface.
- Built-in Status Indicators.
- Limited Asymmetric Data Widths supported with Built-in FIFO.
- Adaptive utilization of either the half or full Built-in FIFO module, dependent on the selected configuration.
- Prioritizes efficient resource utilization, occasionally opting for increased clock cycles to conserve resources in specific cases.

Overview

FIFO Generator

The FIFO Generator is a dedicated hardware module for effortless integration of FIFO functionality into digital systems. It offers versatile configuration options, including First-Word-Fall-Through, Programmable Full/Empty Flags, and support for Asynchronous Write/Reads with Asymmetric Data Widths.

- First-Word-Fall-Through enables the reading device to 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 in the FIFO facilitate seamless operation in systems with distinct clock domains. Employing a pair of Flip Flop Synchronizers ensures synchronization without encountering meta-stability issues in a two-clock domain system, ensuring optimal FIFO efficiency.
- There are a couple of programmable empty and programmable full signals the thresholds of which the user can modify accordingly. This provides a greater level of control over the status of FIFO memory, combined with the usual signals of Empty and Full.
- Asymmetric data widths in FIFOs empower designers to seamlessly integrate components with varying data processing capabilities, providing adaptability to diverse data types and optimizing system-level performance.

The IP also provides the option to select the implementation utilizing either Built-in FIFO or Distributed Memory making the FIFO generator a versatile and FPGA optimized IP Core. A block diagram for a top level FIFO generated from the FIFO Generator IP is shown in Figure 1.

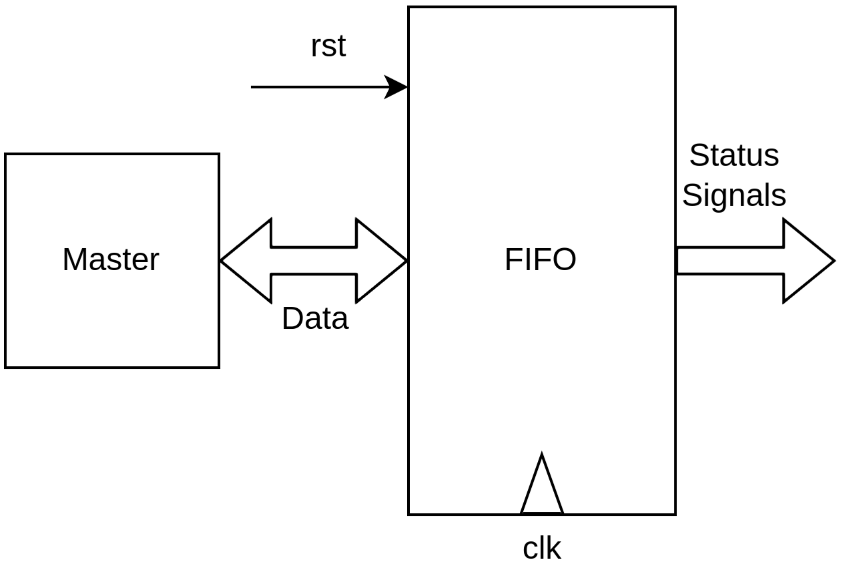


Figure 1: FIFO Block Diagram

IP Specification

The FIFO Generator employs sophisticated algorithms, transforming the FIFO into a circular buffer, guaranteeing seamless tracking of available space. Synchronous pointers operate on binary logic, while asynchronous pointers use gray code encoding to prevent metastable conditions. In asynchronous mode, the configurable depth for Distributed RAM follows powers of 2 due to the gray encoding logic, ensuring stability. However, this constraint is alleviated in the Built-in FIFO design of the asynchronous FIFO. Here, pointers are converted back to binary before reaching counters, enabling a continuous configurable depth range from 3 to 523,264 in the Symmetric Mode.
The FIFO Generator provides users with the versatility to leverage either Built-in FIFO or Distributed Memory implementations, allowing for the mapping of FIFO memory onto Built-in FIFO modules or the utilization of FPGA board logic/LUTs. This flexibility caters to diverse application needs. The synchronous reset mechanism guarantees a cycle between read and write operations, effectively preventing data loss during transitional cycles. The FIFO Generator further enhances its flexibility with support for asymmetric data widths, accommodating scenarios where the read and write data widths differ. This feature allows users to tailor the FIFO configuration to meet specific application demands, enabling seamless integration into systems with varying data width requirements. 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 IP can be seen in Figure 2 that is referenced from the publication of Clifford E. Cummings linked earlier.

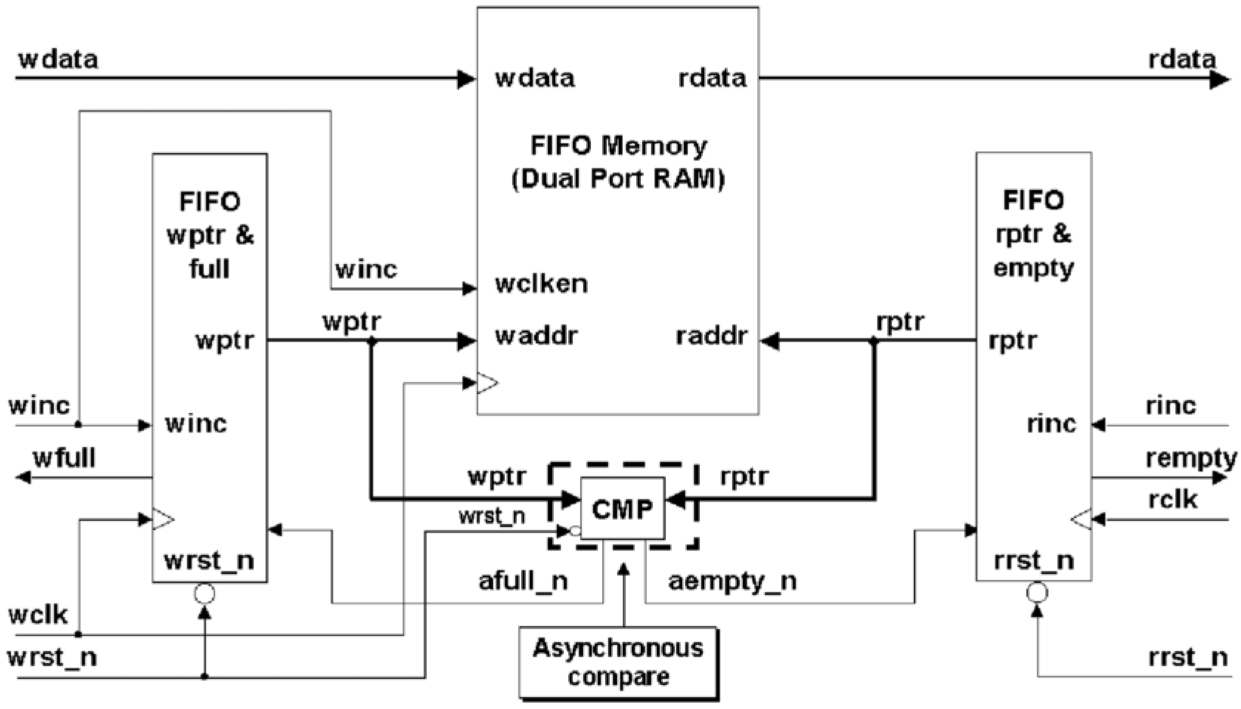


Figure 2: FIFO Internal Diagram

Synchronous versus Asynchronous Clock

A synchronous clock is a shared clock signal among all memory blocks within the device, while an asynchronous clock is specific to each individual memory block.

- Employing a synchronous clock in memory blocks offers distinct advantages. Firstly, it streamlines the design and implementation process by utilizing a single clock signal. Secondly, it guarantees synchronization across all memory blocks, crucial for high-speed data transfer and error prevention. However, a drawback of using a synchronous clock lies in its potential limitation on design flexibility, as all memory blocks must operate at the same clock frequency.

- Utilizing an asynchronous clock in memory blocks introduces greater flexibility into the design, enabling each memory block to operate at a distinct clock frequency. This proves beneficial in designs involving multiple memory blocks with diverse timing requirements. However, the drawback of employing an asynchronous clock lies in the increased complexity of the design. Each clock signal must operate independently, potentially giving rise to timing issues.

The decision to employ either a Synchronous or Asynchronous clock in memories using the FIFO Generator hinges on the unique requirements of the design. Opting for a Synchronous clock is ideal when uniform operating frequencies across all memory blocks are necessary. Conversely, if the design demands flexibility in timing requirements, choosing an Asynchronous clock becomes a more suitable option.

Built-in FIFO versus Distributed Memory

The FIFO Generator provides two distinct types of memory blocks: Built-in FIFO and Distributed RAM. Built-in FIFO utilizes Block RAM, a large, optimized array of memory cells designed for high-speed access. On the other hand, Distributed RAM consists of smaller, flexible memory cells suitable for storing smaller data chunks. The selection between these memory block types is contingent upon the specific requirements of the design.

Symmetric versus Asymmetric Data Width

Symmetric and asymmetric data widths in FIFOs represent two distinct approaches to managing data flow in digital systems. Symmetric data widths involve uniform bit sizes for both the read and write interfaces, providing simplicity and ease of integration. This uniformity ensures straightforward data handling, but it might lead to under-utilization or over-utilization of resources when the application demands different widths for reading and writing. On the other hand, asymmetric data widths allow flexibility by accommodating varying bit sizes for read and write interfaces. This versatility optimizes resource utilization, making it suitable for scenarios where the data width requirements differ. However, the complexity of managing asymmetry introduces additional considerations in design and synchronization. The choice between symmetric and asymmetric data widths depends on the specific needs of the application, weighing simplicity against flexibility and resource efficiency. Due to this complexity, the asymmetric data width mode only supports limited number of data widths, i.e., 9, 18, 36, 72, 144, 288 and 576 whereas the data width is continuous from 3 to 523264 in the symmetric mode. The introduction of asymmetry in data widths in FIFOs introduces a more intricate operational structure, particularly in scenarios where the read data width exceeds the write data width. This asymmetry leads to a pipelined operation, and it comes at the cost of requiring additional clock cycles, the actual amount of clock cycles required can be seen in the Raptor Design Suite while configuring the IP. In practical terms, when utilizing FIFOs with asymmetric data widths in cases where the read data width surpasses the write data width, the operation becomes inherently more complex and necessitates a higher number of clock cycles for effective execution. While this extended operation demands more clock cycles, it offers the advantage of optimizing hardware resources.

Pessimistic Full and Empty Conditions

The empty signal of the FIFO is de-asserted when at least one chunk of data is available to read. When using the Built-in FIFO module, it requires one clock cycle for processing this data and making it available for output. Thus, an additional clock cycle is needed after the first data chunk is written into the memory. The asynchronous mode inherently introduces this delay due to synchronizers, resulting in a two-cycle delay. Additionally, in asymmetric operation, the empty signal de-asserts when one complete word, considering the read data width, is written into the memory. Similarly, the Full signal de-asserts when one complete word is read from the FIFO, taking into account the write data width. This calls for the implementation of "Pessimistic Empty and Pessismistic Full" implementation of these status signals in accordance with the workings of *Clifford E. Cummings* in his publication *Simulation and Synthesis Techniques for Asynchronous FIFO Design*. This implies that the Full and Empty signals are asserted promptly but experience a two-clock cycle delay during de-assertion. This delay is consistent across both Symmetric and Asymmetric modes. Even in Symmetric mode, where the delay is introduced for the memory to process data before it's ready for reading, which also aligns it with the asynchronous mode.

Note:

- *The pessimistic implementation of the empty and full signals holds true exclusively in cases where the Built-in FIFO is employed or when Distributed Memory is used in the Synchronous Mode.*

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 (Surelog)	Raptor (Icarus)	Raptor

Table 1: IP Details

Parameters

Table 2 lists the parameters of the FIFO Generator.

Parameter	Values	Default Value	Description
DEPTH	3 - 523264	1024	FIFO Depth
DATA WIDTH	1 - 1024	36	FIFO Write / Read Data Width in Symmetric Mode
FULL VALUE	2 - (DEPTH - 1)	1000	Programmable Full Value
EMPTY VALUE	1 - (DEPTH - 1)	20	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
BUILTIN FIFO	0 / 1	1	Distributed RAM / Built-in FIFO
ASYMMETRIC	0 / 1	0	Symmetric or Asymmetric Read and Write Data Widths
DATA WIDTH READ	9, 18, 36, 72, 144, 288, 576	36	Read Data Width for Asymmetric Access
DATA WIDTH WRITE	9, 18, 36, 72, 144, 288, 576	36	Write Data Width for Asymmetric Access
IP TYPE	-	FIFOGEN	Type of Peripheral
IP VERSION	-	<ip_version>	Version of Peripheral
IP ID	-	<date_and_time>	Date and Time of the generated Peripheral

Table 2: Parameters

Note:

- *Both the FULL VALUE and EMPTY VALUE parameters are dependent on the DEPTH of the FIFO and are configured accordingly.*
- *The DATA WIDTH READ and DATA WIDTH WRITE parameters are only accessible when in Built-in FIFO with ASYMMETRIC mode.*
- *The ranges for DEPTH and DATA WIDTHs are dependent on each other.*

Port List

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

Signal Name	I / O	Description
din {DATA_WIDTH_WRITE}	I	Data Input
dout {DATA_WIDTH_READ}	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.

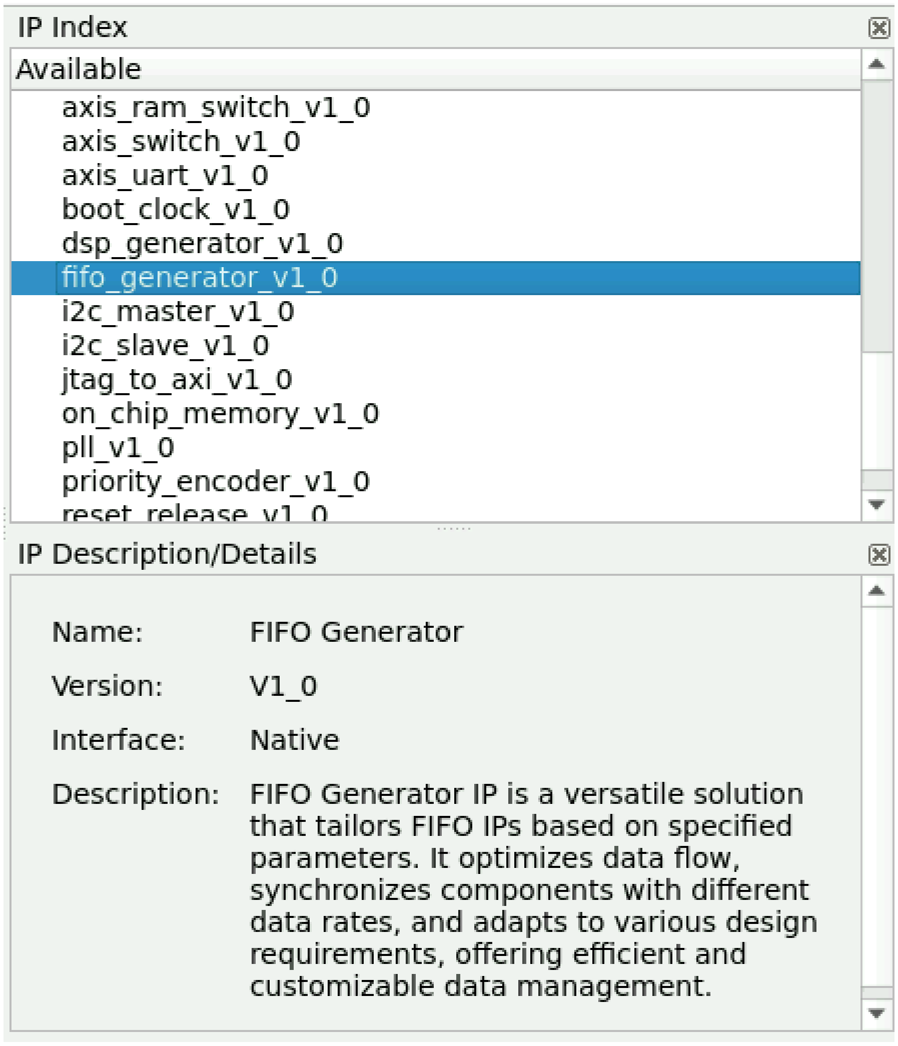
Tool	Raptor Design Suite			
FPGA Device	GEMINI			
Configuration			**Resource Utilization**	
Minimum Resource	**Options**	**Configuration**	**Resources**	**Utilized**
	DATA WIDTH	1	FIFO	0.5
	DEPTH	3	LUTs	17
	FULL THRESHOLD	0	Registers	11
	EMPTY THRESHOLD	0		
	BUILT-IN FIFO	1		
	SYNCHRONOUS	1		
	FIRST WORD FALL THROUGH	0		
	ASYMMETRIC	0		
Maximum Resource	**Options**	**Configuration**	**Resources**	**Utilized**
	DATA WIDTH	128	FIFO	127.5
	DEPTH	34816	LUTs	6158
	FULL VALUE	1000	Registers	469
	EMPTY VALUE	20	Carry Adders	34
	BUILT-IN FIFO	1		
	SYNCHRONOUS	0		
	FIRST WORD FALL THROUGH	1		
	ASYMMETRIC	0		

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.

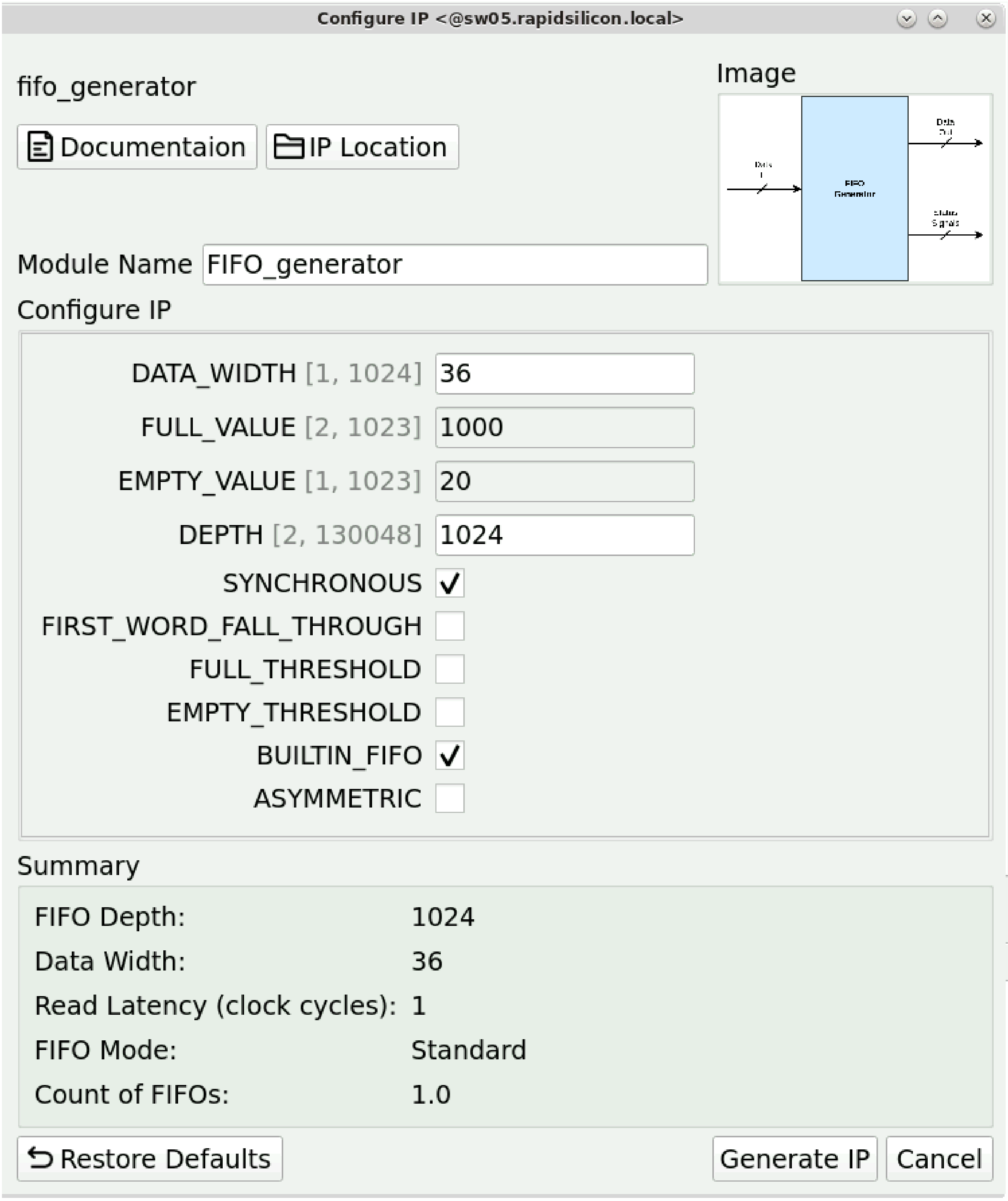


Name:	FIFO Generator
Version:	V1_0
Interface:	Native
Description:	FIFO Generator IP is a versatile solution that tailors FIFO IPs based on specified parameters. It optimizes data flow, synchronizes components with different data rates, and adapts to various design requirements, offering efficient and customizable data management.

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.



Module Name: FIFO_generator

Configure IP

DATA_WIDTH [1, 1024]	36
FULL_VALUE [2, 1023]	1000
EMPTY_VALUE [1, 1023]	20
DEPTH [2, 130048]	1024
SYNCHRONOUS	✓
FIRST_WORD_FALL_THROUGH	☐
FULL_THRESHOLD	☐
EMPTY_THRESHOLD	☐
BUILTIN_FIFO	✓
ASYMMETRIC	☐

Summary

FIFO Depth:	1024
Data Width:	36
Read Latency (clock cycles):	1
FIFO Mode:	Standard
Count of FIFOs:	1.0

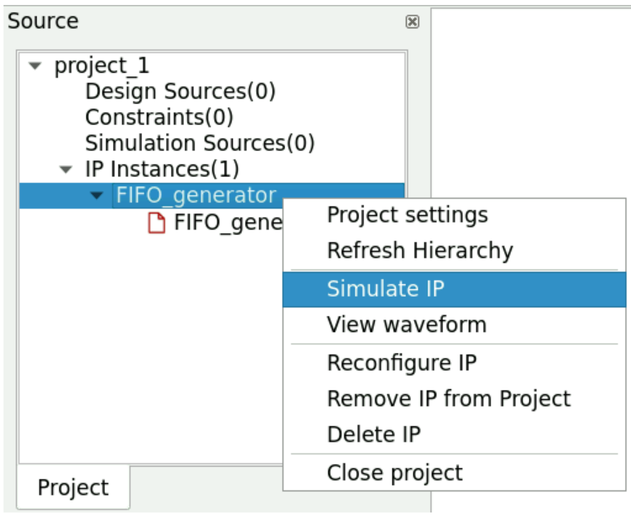
Figure 4: IP Configuration

Synthesis and PR

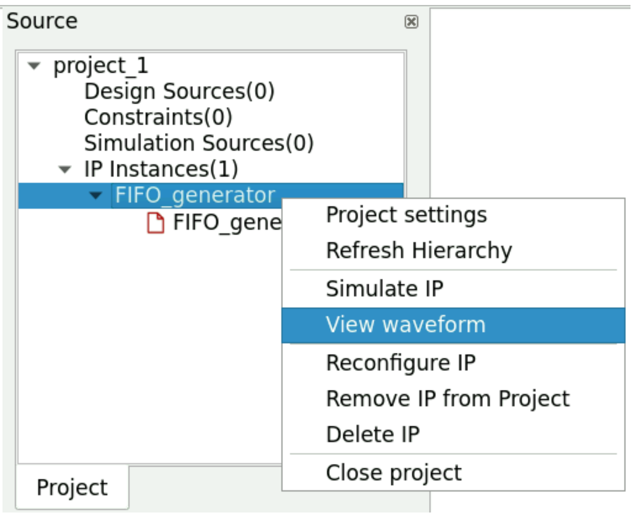
Raptor Suite is armed with tools for Synthesis and the generated post-synthesis net-lists can be viewed and analyzed from within the Raptor. The generated bit-stream can then be uploaded on an FPGA device to be utilized in hardware applications.

Test Bench

The FIFO IP, developed in Verilog HDL, can be efficiently stimulated through various industry-standard methods. These methods encompass using simple Verilog test benches or employing more sophisticated approaches such as stimulating the FIFO through an operating system or via bare-metal firmware. The included test bench for this IP is Verilog-based and can be customized to align with the specific 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 by clicking on the "Simulate IP" button as shown in figure 5a. The waveform can then be viewed in the integrated wave-viewer by clicking the "View waveform" as shown in the figure 5b.



(a) Simulate IP Window



(b) View Waveform Window

Figure 5: IP Source Window

To validate the functionality of the generated FIFO IP, the testbench follows a rigorous procedure. Initially, it writes a set of data into a dedicated memory array created within the testbench. Subsequently, during the read operation from the FIFO, the output data is meticulously compared with the initially written data in the memory array. This thorough verification process ensures the proper operation and accuracy of the generated FIFO IP.

A sample waveform can be seen in the Figure 6 with the First Word Fall Through mode enabled providing the first value at the output before a Pop command is issued.

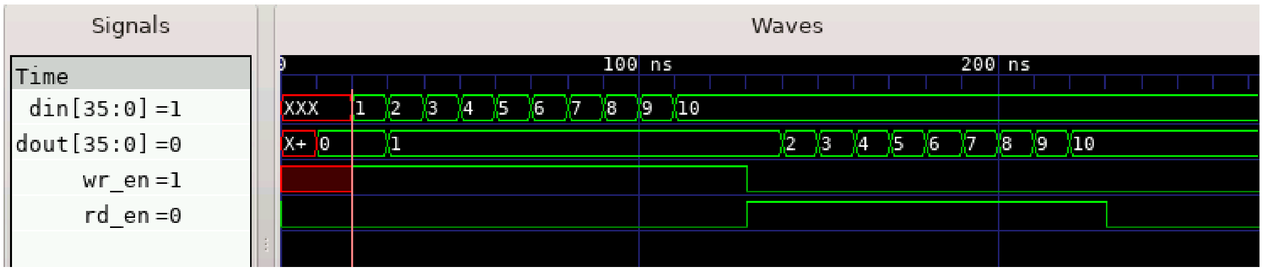


Figure 6: Test Results

The full, overflow and programmable full status indicators can be seen in Figure 7. Here, the Pessimistic approach to these signals is evident, where the full is asserted promptly but deasserted after two clock cycles. It's important to highlight that the output is 0 due to the inactive state of the First Word Fall Through mode.

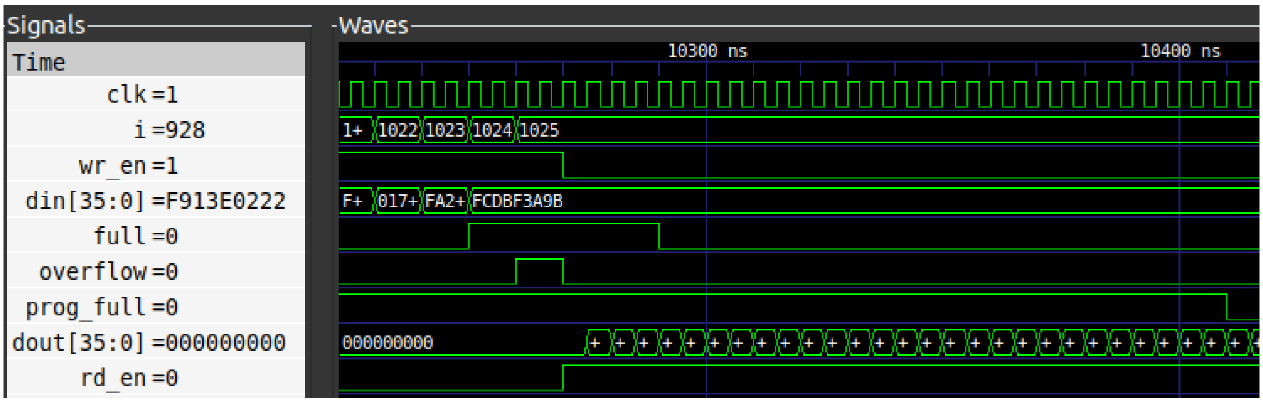


Figure 7: Full conditions

The empty and programmable empty conditions can be seen in Figure 8. Again, the Pessimistic approach to these signals is evident, where the empty is asserted promptly but deasserted after two clock cycles. It's important to highlight that the output is 0 due to the inactive state of the First Word Fall Through mode.

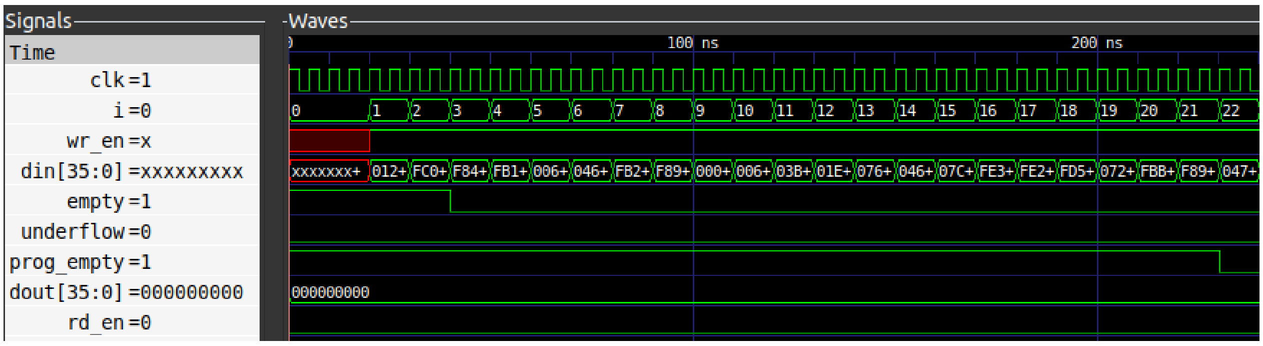
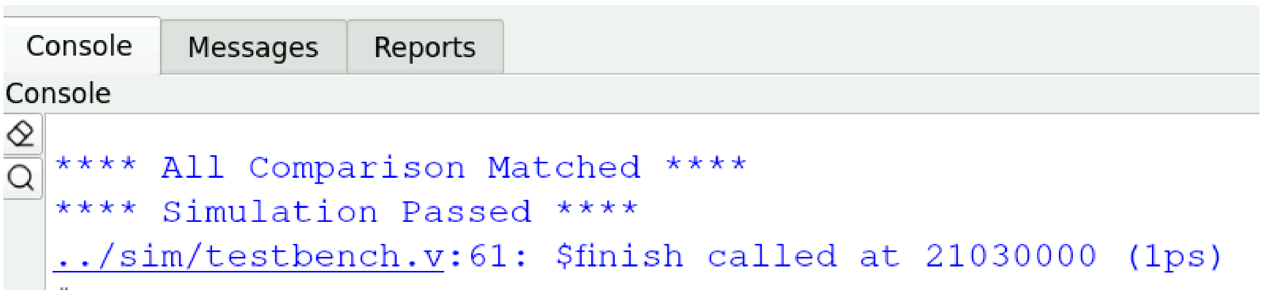


Figure 8: Empty conditions

The simulation results are then shown on the console window as shown in the figure 9.



```
**** All Comparison Matched ****
**** Simulation Passed ****
../sim/testbench.v:61: $finish called at 21030000 (1ps)
```

Figure 9: Simulation Results

Release

Release History

Date	Version	Revisions
December 5, 2023	0.2	Asymmetric Feature added with resource utilization improvements
August 01, 2023	0.1	Initial version FIFO Generator User Guide Document