FIFO Generator
(Beta Release)

Version 0.1

November 21, 2023

Copyright

Trademarks

Disclaimers

Contents

IP Summary **3**
Introduction 3
Features 3

Overview **4**
FIFO Generator 4

IP Specification **5**
Synchronous versus Asynchronous Clock 6
Built-in FIFO versus Distributed Memory 6
Standards 6
IP Support Details 6
Parameters 7
Port List 8
Resource Utilization 9

Design Flow **10**
IP Customization and Generation 10
Parameters Customization 11
Synthesis and PR 12

Test Bench **12**

Release **15**
Release History 15

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.
- Configurable support for first-word-fall-through.
- Native standard FIFO interface.
- Built-in Status Indicators.
- Limited Asymmetric Data Widths supported with built-in FIFO.
- Dynamic selection of half or full built-in FIFO bank based on the chosen configuration.

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, Programmable Full / Empty Flags and Asynchronous Write and Reads.

- 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 enable the FIFO to work in systems with different clock domains that require synchronization between the read and write operations. This synchronization is achieved via a pair of Flip Flop Synchronizers. These synchrnoziners ensure that the meta-stable conditions do not occur when working in a two clock domain system while also making sure that the FIFO works at the highest 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.

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.

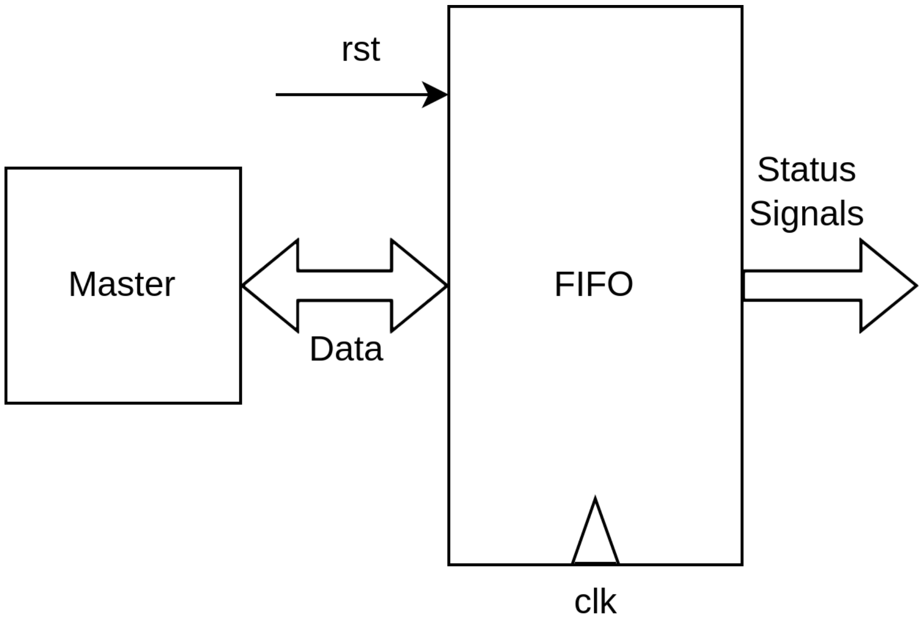

Figure 1: FIFO Block Diagram

IP Specification

The FIFO Generator 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 Built-in FIFO 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 - 52326.
By giving the support for both Built-in FIFO and Distributed Memory implementations, the user is able to choose between mapping the FIFO on the Built-in FIFO 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.

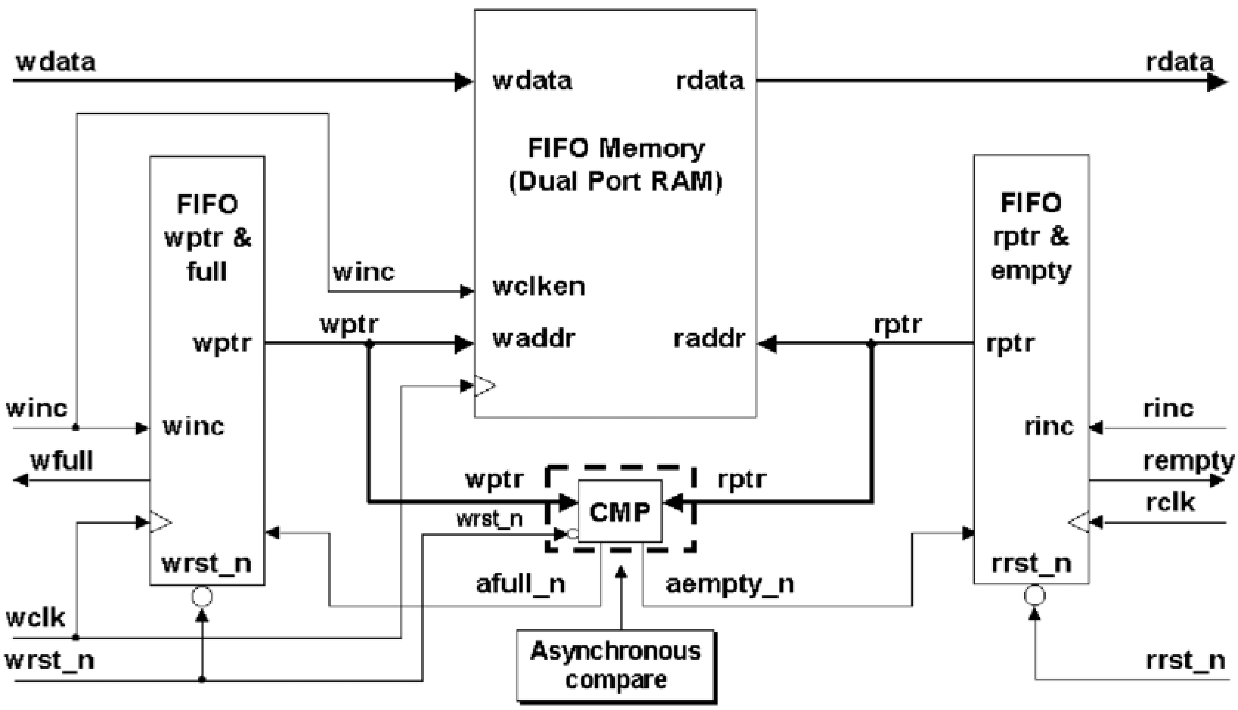

Figure 2: FIFO Internal Diagram

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.

Built-in FIFO versus Distributed Memory

Two types of memory blocks can be generated with FIFO Generator, i.e., Built-in FIFO and Distributed RAM. Built-in FIFO uses the Block RAM which 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	2 - 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$_{F}IFO$	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

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 and 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.

<table>
<tr><th>Tool</th><td colspan="4">Raptor Design Suite</td></tr>
<tr><th>FPGA Device</th><td colspan="4">GEMINI</td></tr>
<tr><th colspan="3">Configuration</th><th colspan="2">Resource Utilization</th></tr>
<tr><td rowspan="9">Minimum Resource</td><th>Options</th><th>Configuration</th><th>Resources</th><th>Utilized</th></tr>
<tr><td>DATA WIDTH</td><td>1</td><td rowspan="8">FIFO
LUTs
Registers</td><td rowspan="8">0.5
17
11</td></tr>
<tr><td>DEPTH</td><td>3</td></tr>
<tr><td>FULL THRESHOLD</td><td>0</td></tr>
<tr><td>EMPTY THRESHOLD</td><td>0</td></tr>
<tr><td>BUILT-IN FIFO</td><td>1</td></tr>
<tr><td>SYNCHRONOUS</td><td>1</td></tr>
<tr><td>FIRST WORD FALL THROUGH</td><td>0</td></tr>
<tr><td>ASYMMETRIC</td><td>0</td></tr>
<tr><td rowspan="9">Maximum Resource</td><th>Options</th><th>Configuration</th><th>Resources</th><th>Utilized</th></tr>
<tr><td>DATA WIDTH</td><td>128</td><td rowspan="8">FIFO
LUTs
Registers
Carry Adders</td><td rowspan="8">127.5
6158
469
34</td></tr>
<tr><td>DEPTH</td><td>34816</td></tr>
<tr><td>FULL VALUE</td><td>1000</td></tr>
<tr><td>EMPTY VALUE</td><td>20</td></tr>
<tr><td>BUILT-IN FIFO</td><td>1</td></tr>
<tr><td>SYNCHRONOUS</td><td>0</td></tr>
<tr><td>FIRST WORD FALL THROUGH</td><td>1</td></tr>
<tr><td>ASYMMETRIC</td><td>0</td></tr>
</table>

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.

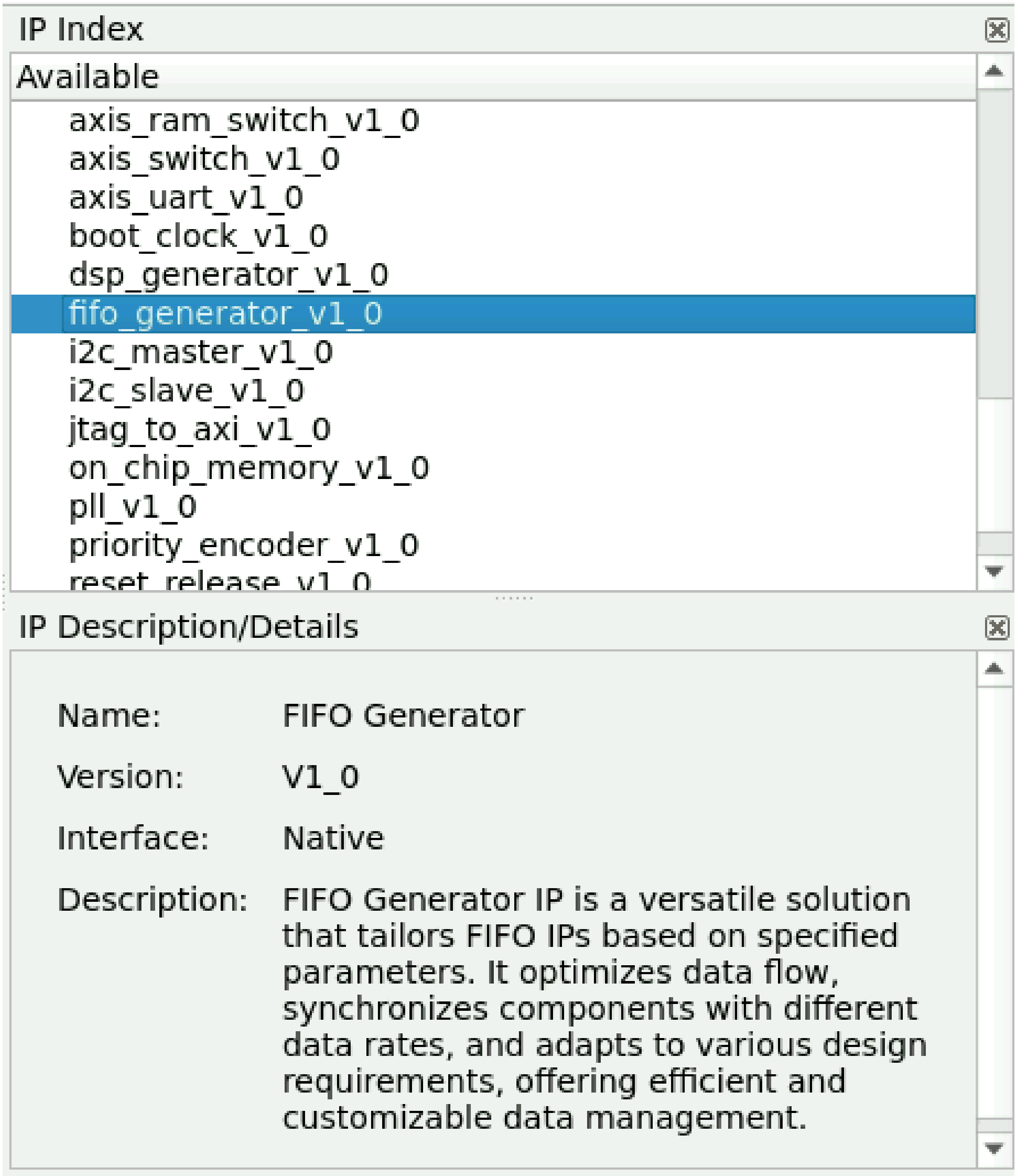

Figure 3: IP list

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.

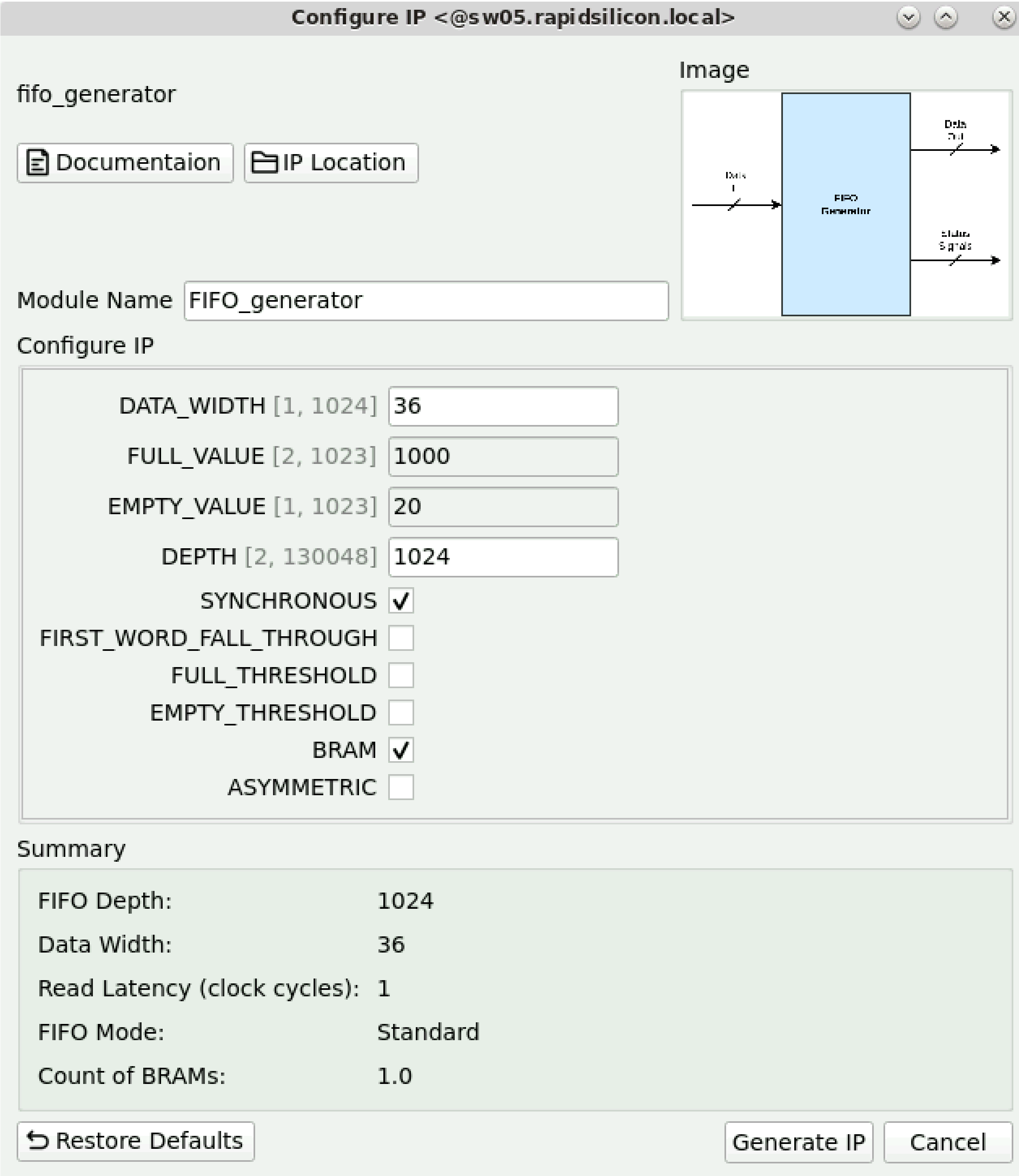

Figure 4: IP Configuration

Synthesis and PR

Raptor Suite is armed with tools for Synthesis 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, based on Verilog HDL, can be stimulated by any number of industry standard means. These may include simple Verilog test benches or stimulating the FIFO via an OS or via bare-metal firmware. The bundled test-bench for this IP is a Verilog based testbench that can be 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 by clicking on the "Simulate IP" button as shown in figure 5.

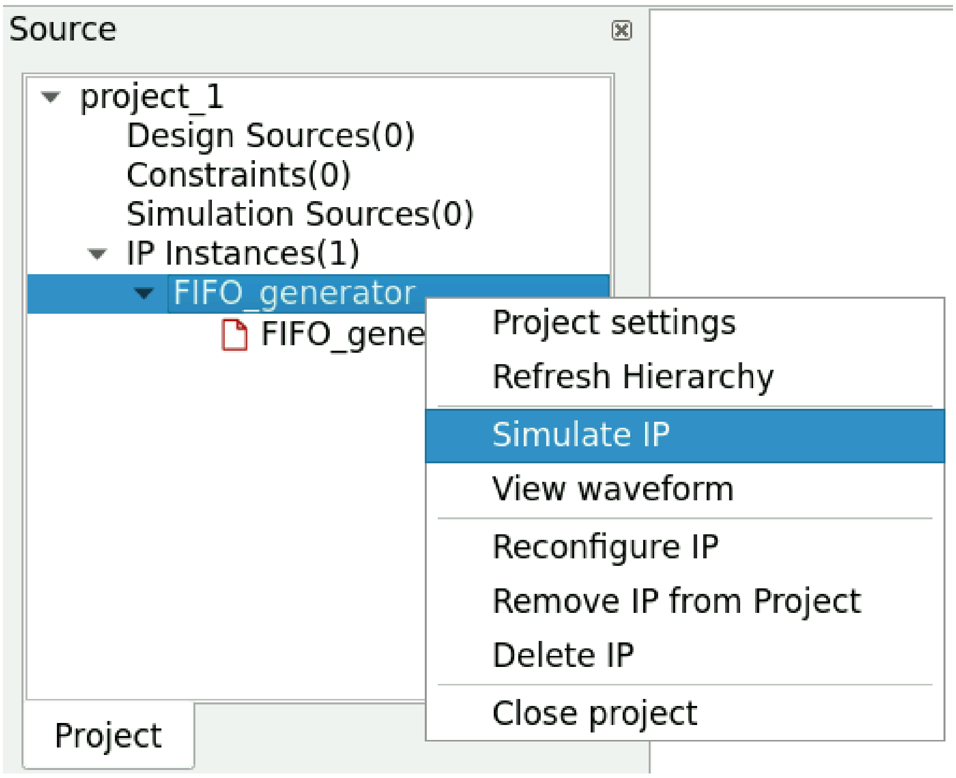

Figure 5: Simulate IP Window

To make sure that the generated FIFO IP works as intended, the testbench writes the same data in a memory array created inside the testbench Then when the read happens from the FIFO, the output data is compared with the data that was written earlier in the memory array. The waveform can then be viewed in the integrated wave-viewer by clicking the "View waveform" as shown in the figure 6.

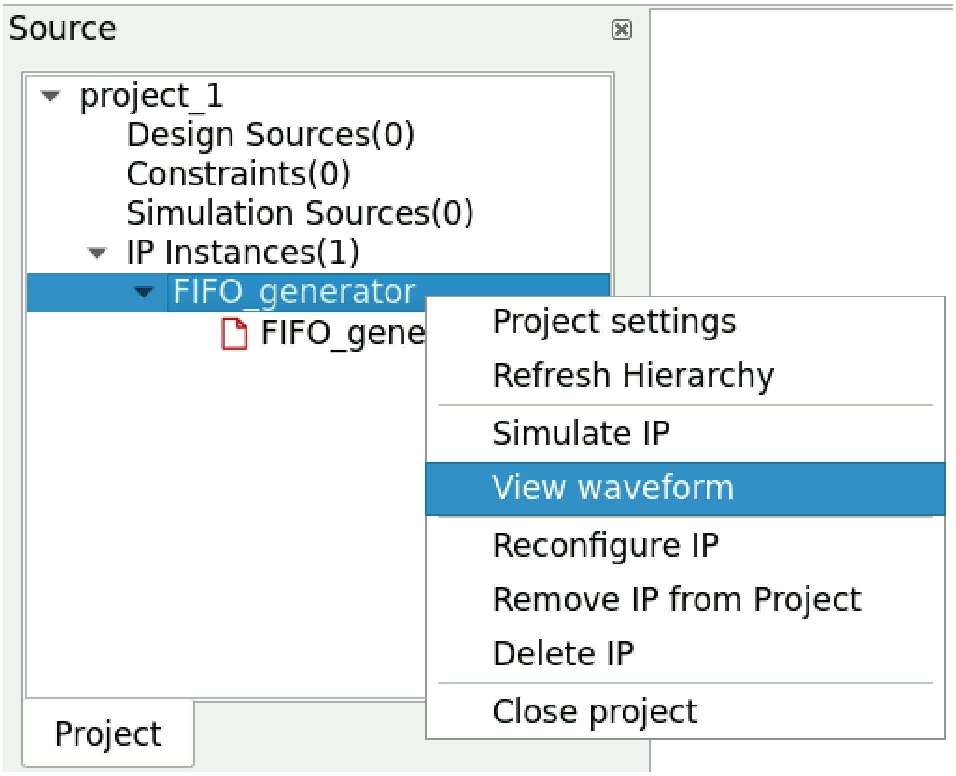

Figure 6: View Waveform Window

A sample waveform can be seen in the Figure 7.

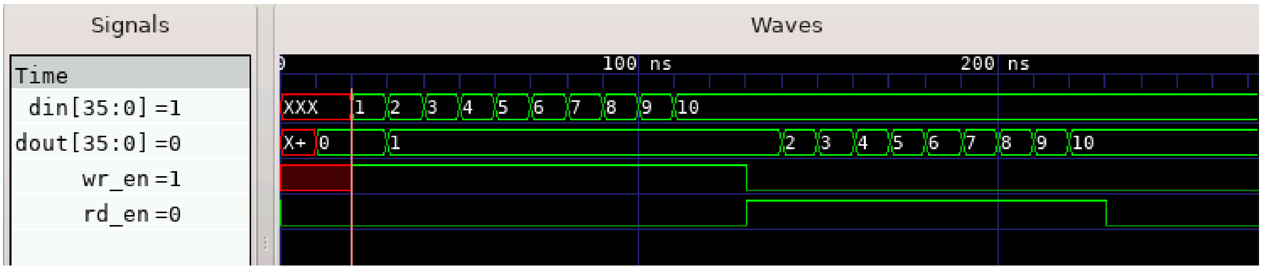

Figure 7: Test Results

The full and empty conditions can be seen in Figure 8.

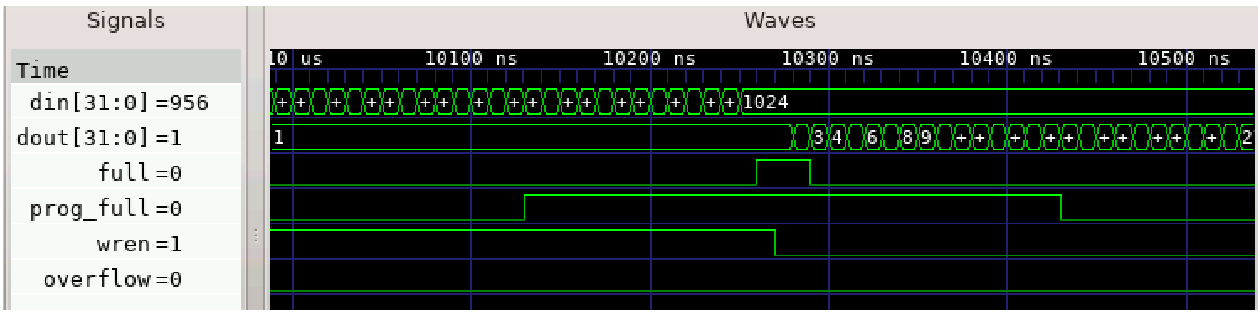

Figure 8: Full conditions

The empty and programmable full conditions can be seen in Figure 9.

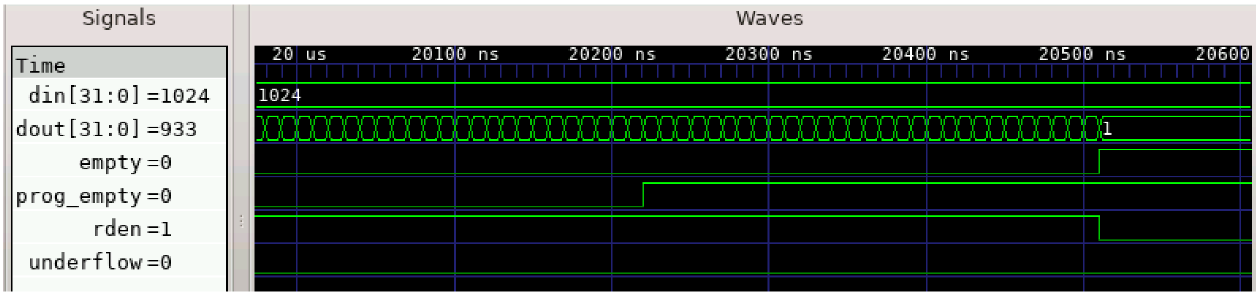

Figure 9: Empty conditions

The simulation results are then shown on the console window as shown in the figure 10.

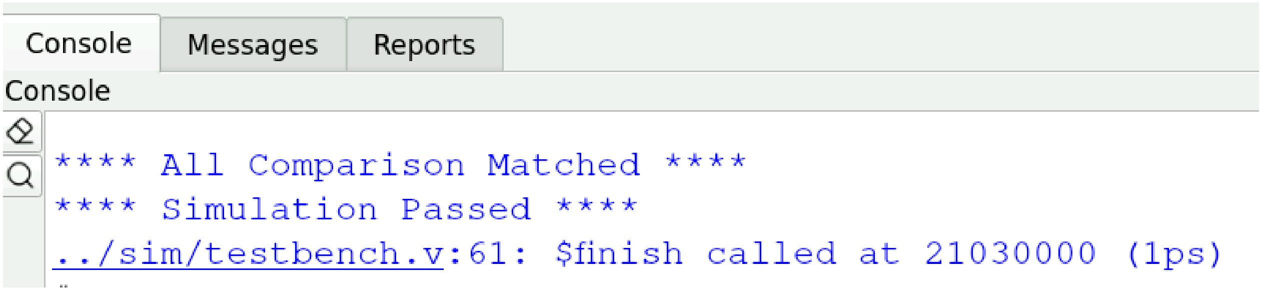

Figure 10: Simulation Results

Release

Release History

Date	Version	Revisions
November 21, 2023	0.1	Initial version FIFO Generator User Guide Document