DSP Generator v1.0

IP User Guide (Beta Release)

February 29, 2024

Contents

IP Summary

Introduction

Digital Signal Processing is one of the most important blocks on compute in modern systems. Its use cases spans all over from clicking photographs from a camera to being used in developing high functioning neural networks such as Natural Level Processing. Digital Signal Processing is utilized to extract the needed information from signals, clean the required information and manipulate it according to the specification. Digital Signal Processing can be done by general purpose computers, or by specialized blocks of compute called Digital Signal Processors. A Digital Signal Processor is a highly focused compute block that serves only one purpose and that is of Digital Signal Processing. At its root, a Digital Signal Processor does its work by implementing a bunch of pre-designed filters, designed specifically with the use case in mind, having a bunch of pre-determined coefficients which are then multiplied with the input data producing the required output data. As it stands, all collected signals in the real world needs filtration before they can be utilized for any useful purpose and this alone gives the significance of highly customizable and scalable Digital Signal Processors.

Features

- Supports computation up to 72x72 bits.
- Supports pipelining to re-utilize resources for the computation.
- Supports signed and unsigned computations.
- Supports various sorts of equations for the computations.
- Support for choosing between multiple features to implement the computations from depending upon the resource available in a "DSP block vs FPGA logic" manner.
- Customizable and Scalable DSP solutions.
- Support Right shifting to align Accumulator data significant bits to output width.
- Support Rounding of Shifted Data.
- Support Saturation (signed and unsigned) of shifted data.
- Support Fixed Coefficients for efficient FIR filter.

Overview

DSP

Digital Signal Processors work by implementing various mathematical algorithms to achieve the computations in an efficient manner. A large number of mathematical operations need to be computed quickly and repeatedly on a series of data samples. A block diagram of the DSP Core is shown in Figure 1. As can be seen in the figure below, the DSP is made up of a various other blocks that include an Adder, a Multiplier, Shift Registers, an Accumulator, a couple of Muxes for selection of input lines and output format. It also consists of registers at both the input and output sides to support pipelining. It also has the ability to handle signed numbers because of the inclusion of a sign extender as well as the ability to round off numbers. The Accumulator gives an output of 64-bit which is then truncated to 38 bits as the DSP output. A shift register gives the ability to select which of these 64 bits are to be output from the DSP, otherwise in case of overflow, the maximum value of 38 bits is output from the DSP. A Mux is utilized at the input to select between inputs and coefficients that can then be utilized within the DSP to form an FIR filter. The inputs to the DSP block are 18 and 20 bits wide respectively but it can be used for greater numbers by utilizing DSP algorithms.

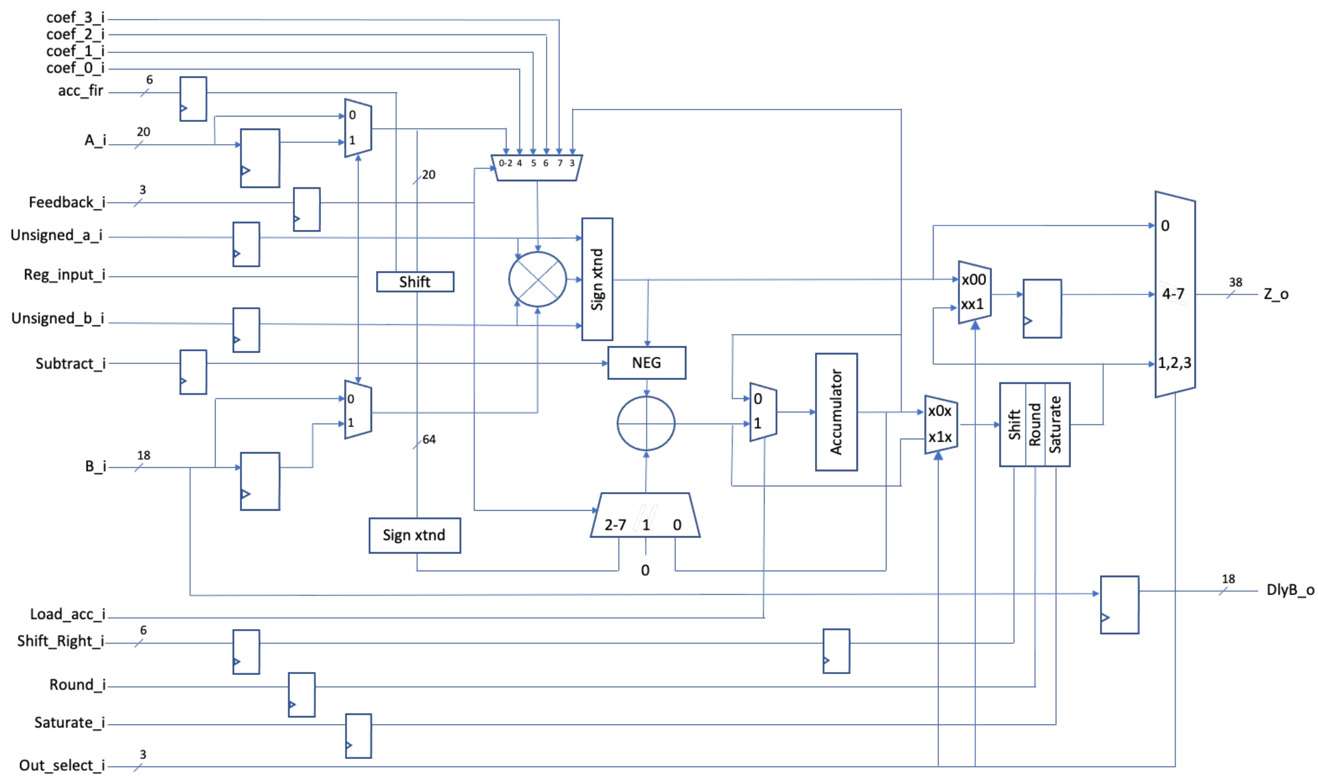


Figure 1. DSP Block Diagram

Licensing

IP Specification

The DSP core supports both signed and unsigned numbers up-to 72x72 bit wide numbers producing an output of 144 bits. The embedded DSP block offers a 20x18 built-in multiplier. This IP allows the implementation of a wider multiplier using embedded DSP block and automatically generates the required additional logic. This is achieved by utilizing the Karastuba Algorithm for DSP Decomposition. This algorithm essentially divides large numbers into smaller chunks that can then be multiplied via the DSP blocks that accept those smaller numbers. The results of these smaller chunks are then added up to produce the final answer of the actual operands. The internal accumulators of the DSPs can also be utilized for this Karatsuba Algorithm to compute the result in a pipelined manner reducing the cost of DSP blocks and FPGA logic but this does take extra clock cycles compared to the non-pipelined versions that produce the output instantly. Taking the Karatsuba Algorithm one step further, it takes the form of Karatsuba-Ofmann Algorithm that is an advanced form of the original Karatsuba Algorithm. This version mathematically simplifies the original Karatsuba's equations to lessen the number of DSP blocks utilized but at the cost of extra FPGA logic. More information about the Karatsuba-Offman Algorithm can be found at this link. All three of these algorithmic features are provided within the Raptor Design Suite so the correct implementation can be used as according to the user application. More details about all three of these algorithms can be found in the Table 1. A pictographic representation of the nested DSPs can be seen in the Figure 2.

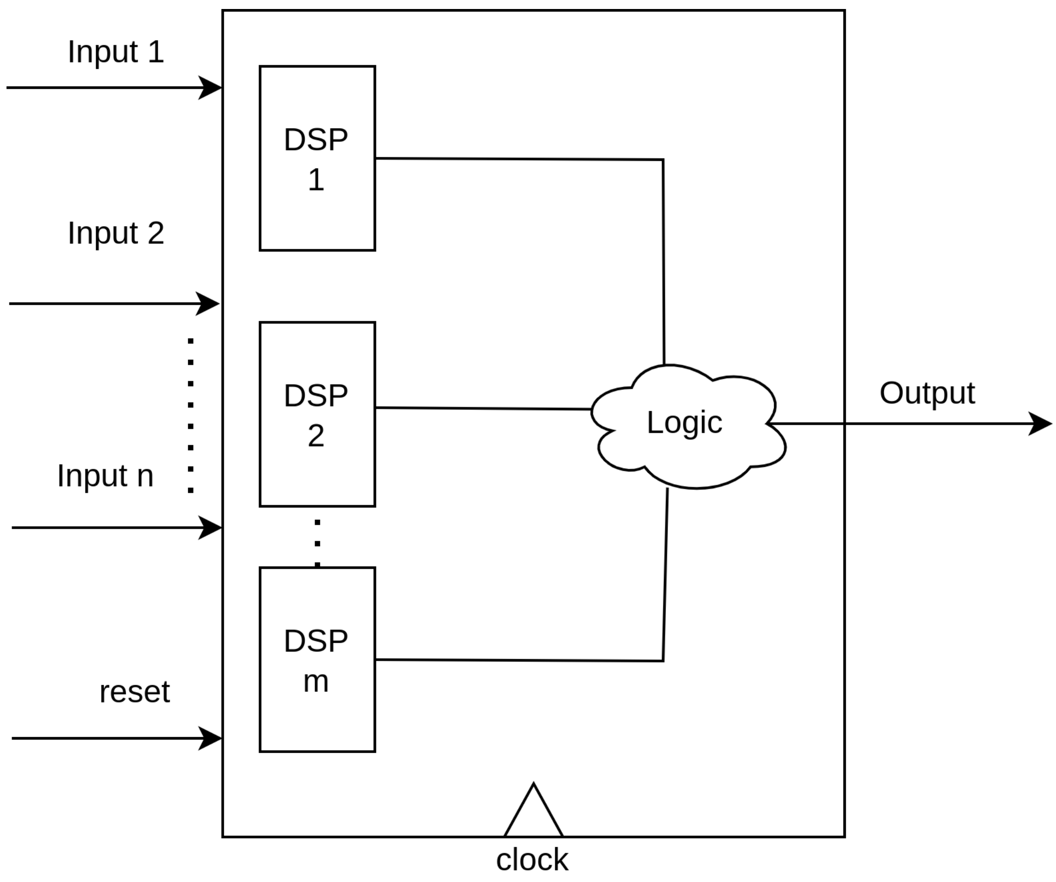


Figure 2. Top Module

Feature Comparison

There are different use cases for all the three available algorithms for the DSP Decomposition and a brief comparison of them is given below in Table 1.

Base Algorithm	Pipelined Algorithm	Enhanced Algorithm
Based on Karatsuba Algorithm	Based on Karatsuba Algorithm in Pipeline	Based on Karatsuba-Offman Algorithm
Combinational Logic with immediate output	Takes multiple clock cycles	Combinational Logic with immediate output
Upto 72x72 bit for unsigned numbers	Upto 72x72 bit for unsigned numbers	Upto 68x68 bit for unsigned numbers
Upto 72x72 bit for signed numbers	Upto 68x68 bit for signed numbers	Upto 64x64 bit for signed numbers
Favours DSP block over FPGA logic	Balances between DSP and logic over clock cycles.	Favours FPGA logic over DSP block
4 DSPs upto 36x36	3 DSPs upto 36x36 and 2 clock cycles	3 DSPs upto 34x34
9 DSPs upto 54x54	5 DSPs upto 54x54 and 3 clock cycles	6 DSPs upto 51x51
16 DSPs upto 72x72	7 DSPs upto 72x72 and 4 clock cycles	10 DSPs upto 68x68
Most DSPs utilized	Least DSPs and Logic utilized	Most FPGA logic utilized

Table 1. Algorithm Details

IP Support Details

The Table 2 gives the support details for Digital Signal Processor.

Compliance		IP Resources					Tool Flow		
Device	Interface	Source Files	Constraint File	Testbench	Simulation Model	Software Driver	Analyze and Elaboration	Simulation	Synthesis
GEMINI	Standard	Verilog	SDC	Verilog / System Verilog	Verilog	Icarus	Raptor	Raptor	Raptor

Table 2. IP Details

Resource Utilization

The parameters for computing the maximum and minimum resource utilization are given in Table 3, 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**
	Base Algorithm	20x18	DSPs	1
			LUTs	-
			Latency	1
Maximum Resource	**Options**	**Configuration**	**Resources**	**Utilized**
	Base Algorithm	72x72	DSPs	16
			LUTs	1977
			Adder Carry	78
			Latency	1
	Pipelined Algorithm	72x72	DSPs	7
			LUTs	454
			Adder Carry	486
			Registers	3
			Latency	4
	Enhanced Algorithm	68x68	DSPs	10
			LUTs	1434
			Adder Carry	1168
			Latency	1

Table 3. DSP Resource Utilization

Parameters

Table 4 lists the parameters for the Digital Signal Processor.

Parameter	Values	Default Value	Description
EQUATION	A*B A*B+C*D A*B+C*D+E*F+G*H	A*B	Equation to implement
UNSIGNED	0 / 1	1	Signed or Unsigned Inputs
REG IN	0 / 1	0	Registered Input
REG OUT	0 / 1	0	Registered Output
FEATURE	Base / Enhanced / Pipeline	Base	Algorithm to implement
A WIDTH	1 - 72 (Base and Unsigned Pipeline) 1 - 68 (Signed Pipeline and Unsigned Enhanced) 1 - 64 (Signed Enhanced)	20	Width of A input
B WIDTH	1 - 72 (Base and Unsigned Pipeline) 1 - 68 (Signed Pipeline and Unsigned Enhanced) 1 - 64 (Signed Enhanced)	18	Width of B input
C WIDTH	1 - 20	20	Width of C input
D WIDTH	1 - 18	18	Width of D input
E WIDTH	1 - 20	20	Width of E input
F WIDTH	1 - 18	18	Width of F input
G WIDTH	1 - 20	20	Width of G input
H WIDTH	1 - 18	18	Width of H input
IP TYPE	-	DSPG	Type of Peripheral
IP VERSION	-	<ip_version>	Version of Peripheral
IP ID	-	<date_and_time>	Date and Time of the generated Peripheral

Table 4. DSP Paramters

Ports

Table 5 lists the top interface ports of the Digital Signal Processor.

Signal Name	I/O	Description
Registered Input / Output OR Pipelined Feature		
clk	I	System Clock
reset	I	Active High System Reset
Equation: AxB		
a	I	First Input
b	I	Second Input
z	O	DSP Output
Equation: AxB + CxD		
a	I	First Input
b	I	Second Input
c	I	Third Input
d	I	Fourth Input
z	O	DSP Output
Equation: AxB + CxD + ExF + GxH		
a	I	First Input
b	I	Second Input
c	I	Third Input
d	I	Fourth Input
e	I	Fifth Input
f	I	Sixth Input
g	I	Seventh Input
h	I	Eighth Input
z	O	DSP Output

Table 5. DSP Interface

Equations Description

1. **AxB**

   AxB is one of the equations supported in DSP Generator which is used to multiply two numbers. This implementation is shown in Figure 3.

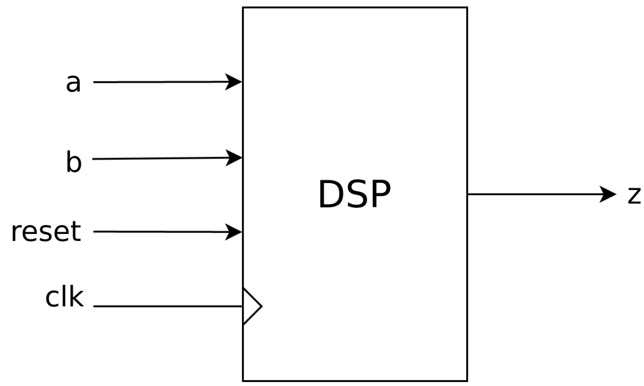


Figure 3. AxB

2. **AxB+CxD**

   AxB+CxD is the second available equation of DSP Generator which is used to accumulate the result of two products. This implementation is shown in Figure 4.

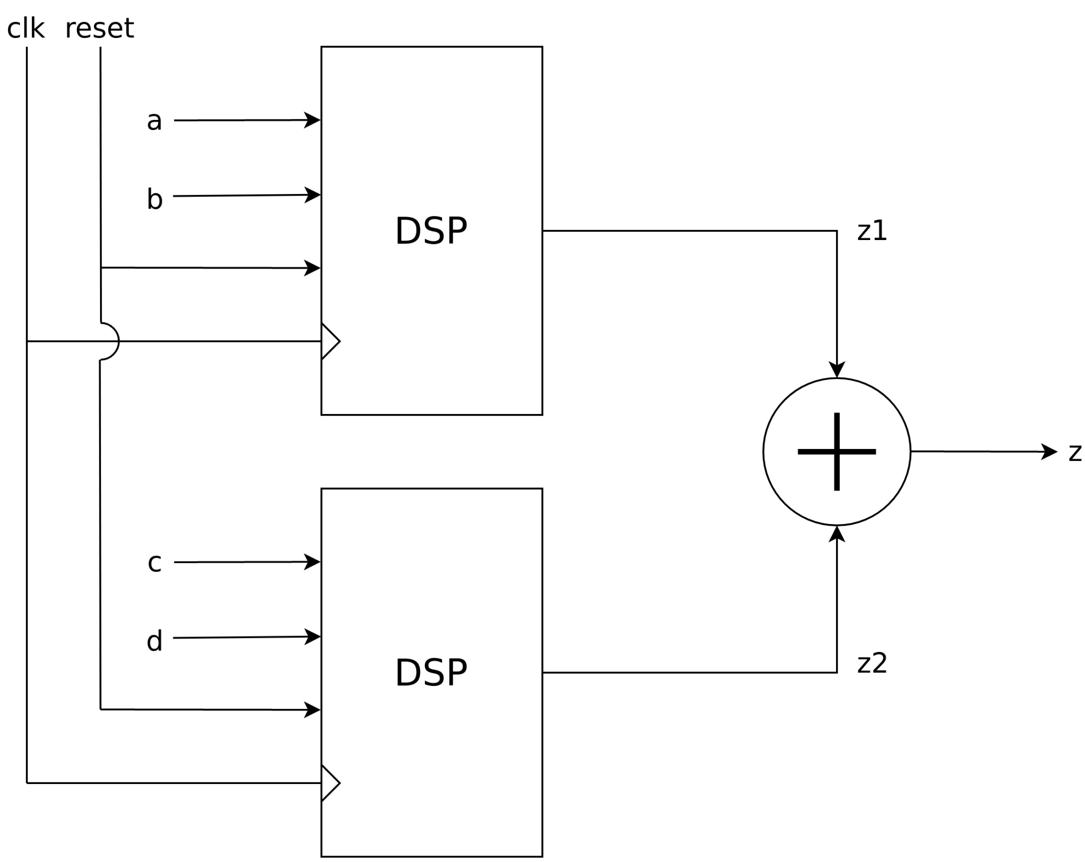


Figure 4. AxB+CxD

3. **AxB+CxD+ExF+GxH**

AxB+CxD+ExF+GxH is the third equation of DSP Generator which is used to accumulate the result of four products. This feature is shown in Figure 5.

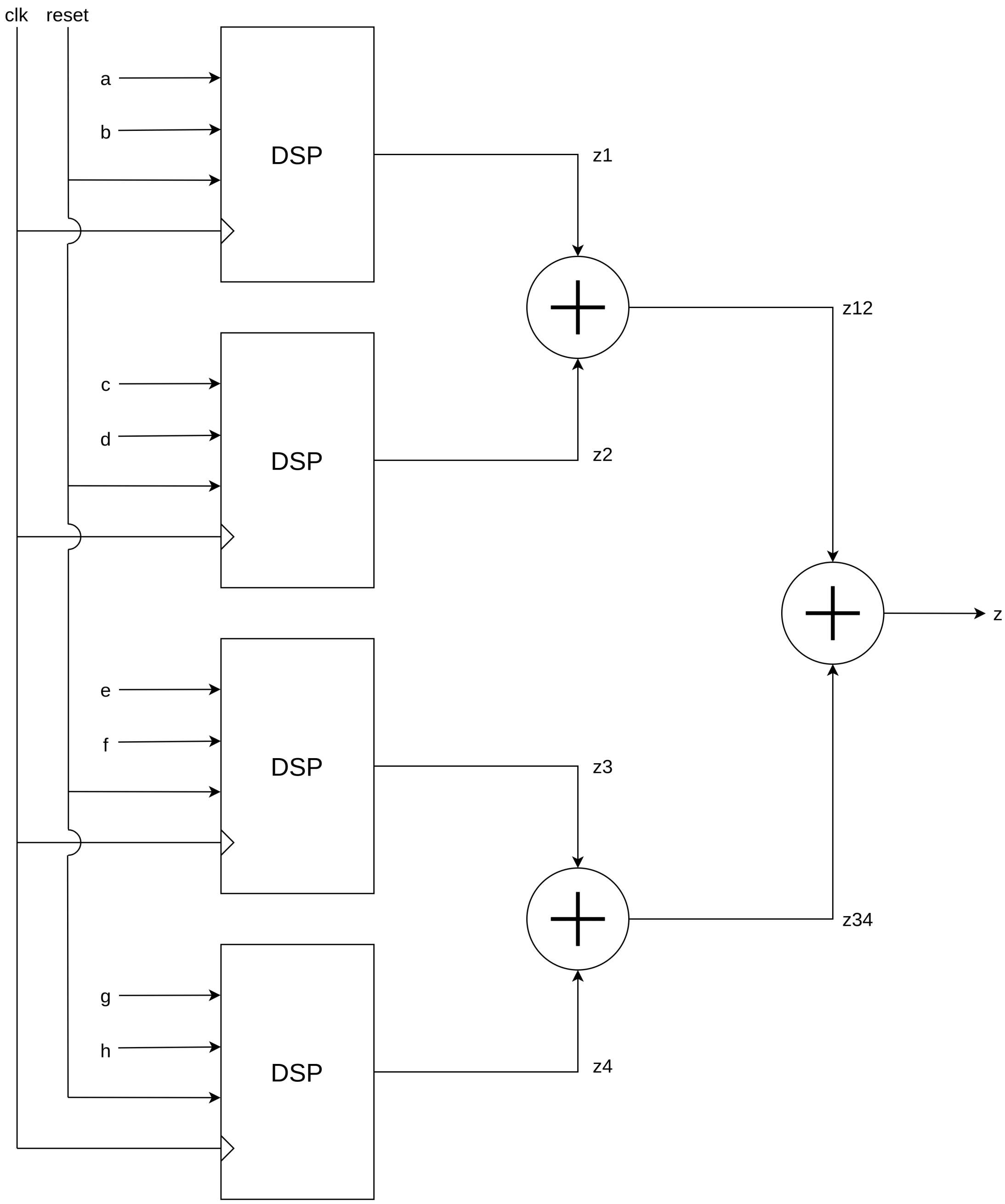


Figure 5. AxB+CxD+ExF+GxH

Design Flow

IP Customization and Generation

Digital Signal Processor IP core is a part of the Raptor Design Suite Software. A customized DSP block can be generated from the Raptor's IP configurator window as shown in Figure 6.

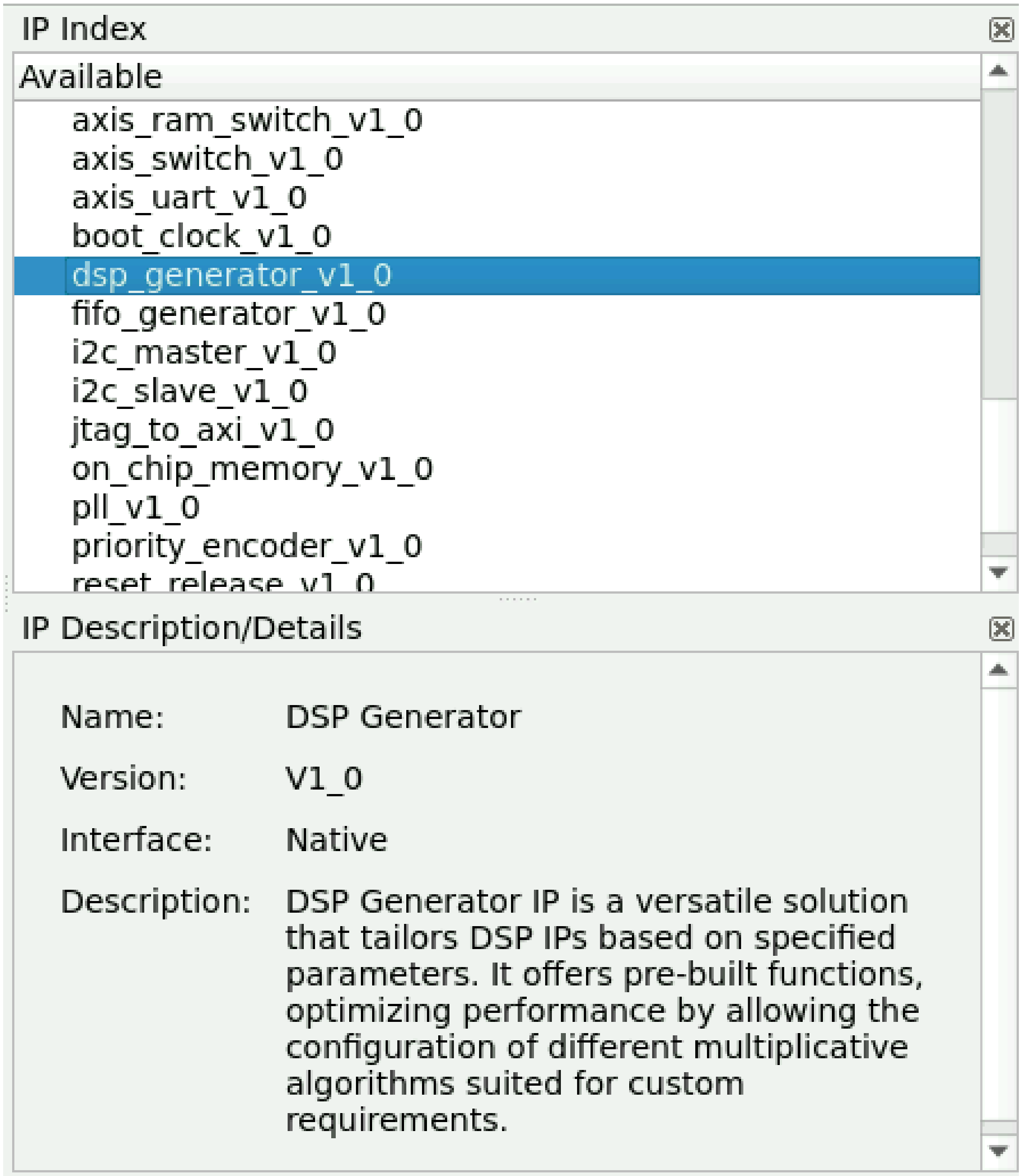


Figure 6. IP list

Parameters Customization: From the IP configuration window, the parameters of the DSP can be configured and DSP options can be enabled or disabled for generating a customized DSP core that suits the user application requirement as shown in Figure 7. After IP Customization, all the source files are made available to the user.

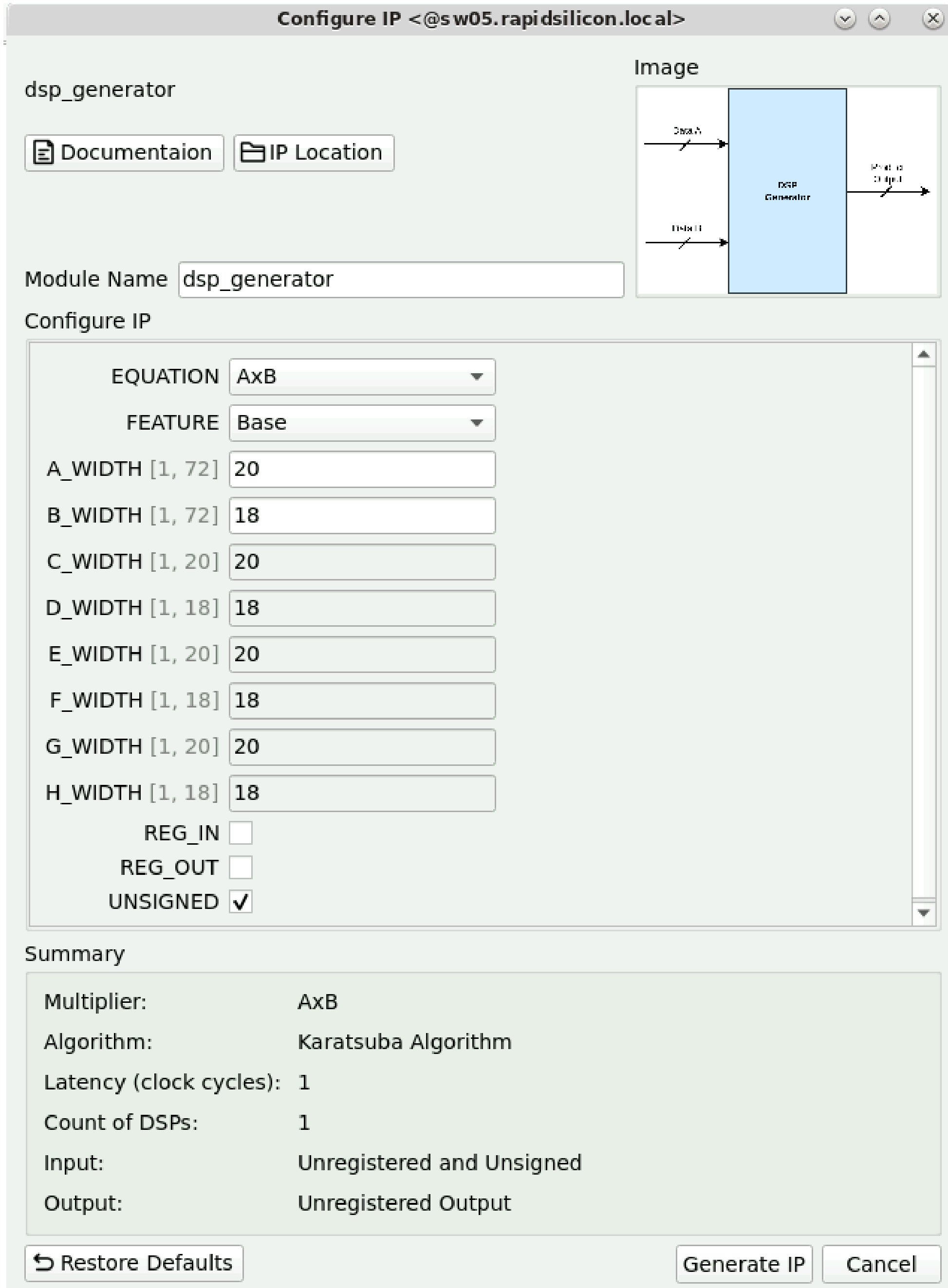


Figure 7. IP Configuration

Example Design

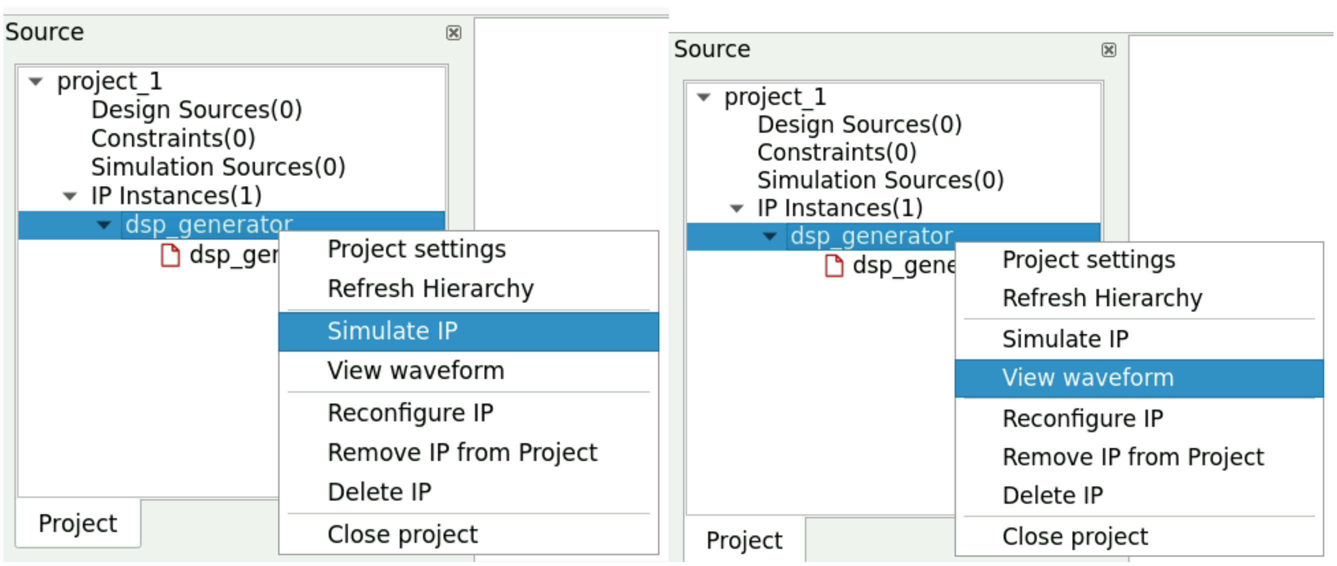
Overview

Simulating the Example Design

Synthesis and PnR

Test Bench

The DSP 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 some 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 DSP 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 DSP IP works as intended, the testbench compares the output of the generated DSP IP to that of a basic Verilog multiplier and hence makes sure that the generated IP works as intended. The simulation can be easily run by clicking the "Simulate IP" button as shown in figure 8a. The waveforms are also dumped for in-depth analysis of the whole operation which can be seen by clicking the "View Waveform" button as shown in 8b.



(a) Simulate IP Window **(b)** View Waveform Window

Figure 8. IP Source Window

A sample waveform for when A is 20 bits and B being 18 bits with registered inputs is shown in figure 9.

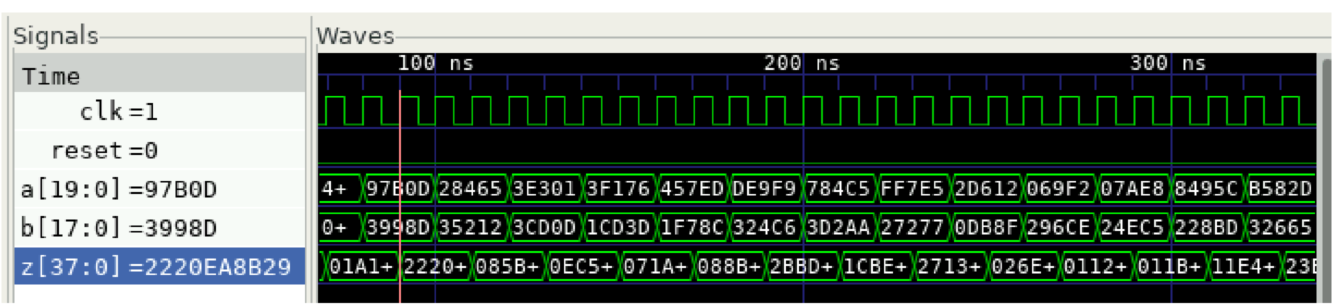


Figure 9. Waveform

The simulation results are also displayed in the console window a glimpse of which can be seen in figure 10.

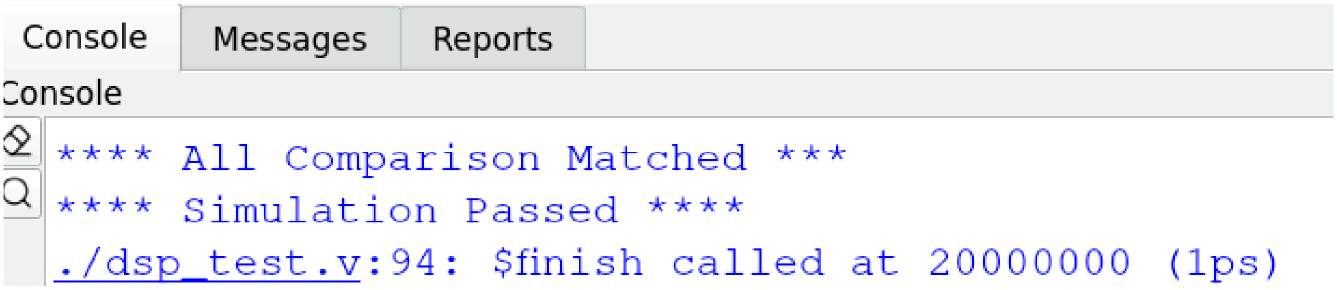


Figure 10. Simulation Results

Revision History

Date	Version	Revisions
February 29, 2024	0.01	Initial version DSP Generator User Guide Document