Open Source SPECIAL FUNCTIONS UNIT – SFU

User Manual 1.0

This manual describes the interface and interconnections of a Special Functions Unit (SFU) able to perform trigonometric and transcendental operations in a floating-point format in single-precision (32 bits).

The SFU was developed to operate the functions suggested in the G80 Architecture of a (General Purpose Graphics Processing Unit) GPGPU. The SFU is integrated into the FlexGripPlus model. However, the SFU core can be easily adapted to any other processor or accelerator with minor changes.

The manual was developed by the Research Group in Robotics and Industrial Automation (GIRA) in the Electronics Engineering School.

Universidad Pedagógica y Tecnológica de Colombia (UPTC)

Colombia, July 2020

Authors:

Juan David Guerrero Balaguera	juandavid.guerrero@uptc.edu.co
Cristhian Fernando Moreno Manrique	Cristhian.Moreno@uptc.edu.co
Josie Esteban Rodriguez Condia	josie.rodriguez@polito.it

Universidad Pedagógica y Tecnológica de Colombia (UPTC)
Electronics Engineering School
Colombia

Politecnico di Torino
Department of Control and Computer Engineering (DAUIN)
Italy

This activity was partially supported by fundings by the European Commission through the Horizon 2020 **RESCUE-ETN** project under grant 722325.

For more information: http://rescue-etn.eu/

Modern devices as GPGPUs usually include special accelerators to process part of complex algorithms or complex operations. These accelerators are known as Special Functions Units (SFUs) and perform transcendental operations and special functions at the hardware level. The SFUs are used to transform, correlate, represent, and process information from the application. This document presents the architecture and operational behavior of a Special Function Unit – SFU designed to perform the following operations:

- $\sin x$
- $\cos x$
- $log_2 x$
- 2^x
- $\frac{1}{\sqrt{x}}$

This SFU performs each operation using approximate algorithms and uses the IEEE 754 floating-point representation in single precision.

The SFU hardware module is composed of 4 basic components. The first one is a *Cordic* module. This component performs the CORDIC algorithm to compute: $\sin x$ and $\cos x$ transcendental functions. This module performs 16 interactions to obtain a valid result. It means that the module needs 17 clock cycles to achieve one correct result. The other operational components perform $log_2 x$, 2^x and $\frac{1}{\sqrt{x}}$. Operations in full parallel, then those components don't require more than one clock cycle to give a valid result. Figure 1 presents the internal SFU module architecture.

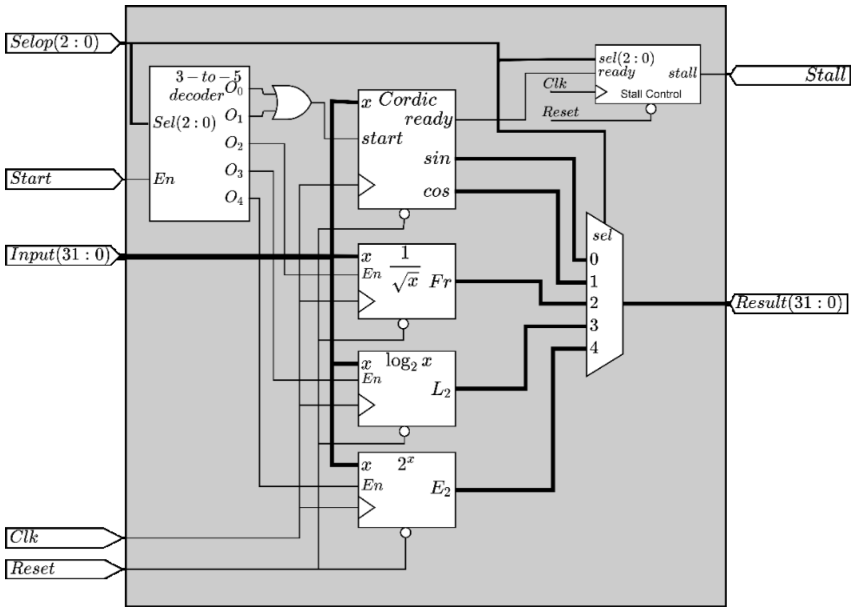


Figure 1. Internal SFU module architecture

TABLE I DESCRIPTION OF THE IN/OUT PORTS OF THE SFU MODULE

Port Name	**Direction**	**Size - Bits**	**Description**
Selop	Input	3	This port allows the user the selection of the operation that wishes to perform in the SFU Note 1. The following table shows the selection code of each SFU operation.
Start	Input	1	Initiates the SFU operation selected by ***Selop*** port, using the data in the ***Input*** port as the argument of the function selected. Note 1
Input	Input	31	Input data to be processed by the SFU. The user must provide input data using IEEE 754 single-precision representation, and the value must be within the interval shown in the table above, taking in to account the operation that the user wishes to perform in the SFU module. Note 1
Clk	Input	1	The clock signal used in the synchronous circuits, the SFU module is active in the rising edge of the clock signal.
Reset	Input	1	SFU asynchronous reset, Low-level active input.
Stall	Output	1	This port reports the state of the SFU module when $\sin x$ or $\cos x$ operation is performed. While the SFU is busy doing the operation; the logic state of stall port is HIGH. When the SFU ends the operation and there is a data valid on ***Result*** port, the stall signal changes to LOW during one clock cycle only. After that, it goes back to HIGH. If the SFU performs $log_2 x$, 2^x, or $\frac{1}{\sqrt{x}}$ operations, the stall port is always LOW. Note 2
Result	Output	31	Output data of the SFU. The user must interpret the data out of this port using IEEE 754 single-precision representation. The valid data trough ***Result*** port arrives at the same time that the stall signal goes to LOW when $\sin x$ or $\cos x$ operations are performed. With other operations, the ***Result*** port will give data valid one clock cycle after the start operation command is sent to the SFU.

Note 1: The data on ***selop, start,*** and ***Input*** ports would be to put at the same time, the start must be HIGH one clock cycle only, and the data in the ***Input*** and ***selop*** ports must be remaining unchanged until the operation finishes.

Note 2: The user must test this port signal after finishes to send the start signal.

The selection of the function is described in the following table.

TABLE II SELECTION CODE USED TO SELECT THE OPERATION IN THE SFU

Selection code (Selop)	Function selection
000	$\sin x$
001	$\cos x$
010	$\frac{1}{\sqrt{x}}$
011	$log_2 x$
100	2^x
101	Reserved to future use
110	Reserved to future use
111	Reserved to future use

Figures 2, 3, and 4 show the waveforms representing the operational behavior of each function in the SFU module.

Figure 2 shows the operation of the $\sin \mathrm{x}$ function. As can be observed, the RESET port signal must be HIGH level, and SELOP (000) port must select (see Table II). The triggering of the START signal must have the same time as the INPUT data. However, one clock cycle would be better to guarantee the correct operation between INPUT and START. The START signal remains HIGH one clock cycle only. During the operation of the module, the value in the INPUT and SELOP ports must remain unchanged. Then, after seventeen clock cycles, the SFU module determines a valid result on the RESULT port. At the same time, STALL port goes to LOW. STALL indicates to the external controller of the SFU that the operation has ended. The STALL port only will be in LOW state one clock cycle. Then, the controller must poll that port or count the clock cycles to know when the operation finishes. The module only accepts new operations after STALL signal changes from LOW to HIGH. Figure 3 has a waveform for $\cos \mathrm{x}$ operation, this operation is performed in the same way that $\sin \mathrm{x}$ operation, the only change is that the SELOP port must be equals to 001.

Figure 4 presents waveforms for $\frac{1}{\sqrt{\mathrm{x}}}$, $\log_2 \mathrm{x}$ and 2^{x} operation sequence. That operations require that data in the INPUT, SELOP, and START ports arrive at the same time. The START signal must be HIGH one clock cycle only. The data valid will appear on RESULT port at the first rising edge of the CLK signal after the start signal goes HIGH. That operation doesn't require STALL signal. Then, STALL signal will be LOW when the SFU performs these operations.

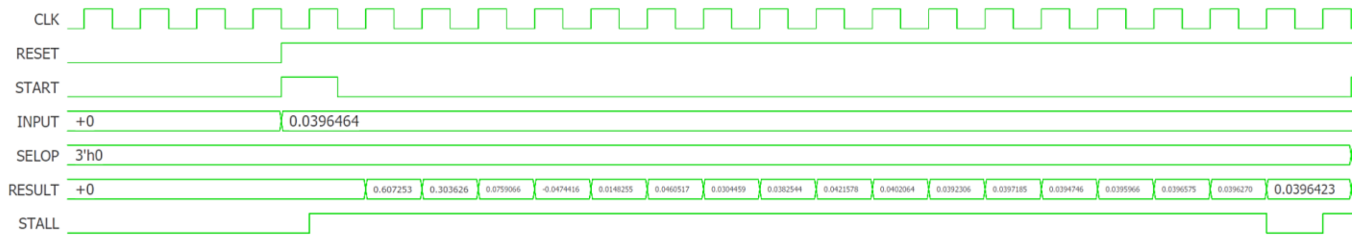


Figure 2. Waveforms for the $\sin x$ operation in the SFU.

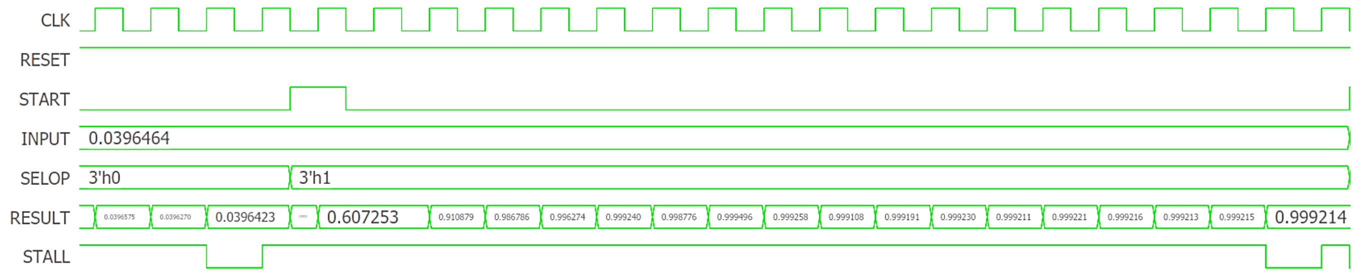


Figure 3. Waveforms for the $\cos x$ operation in the SFU.

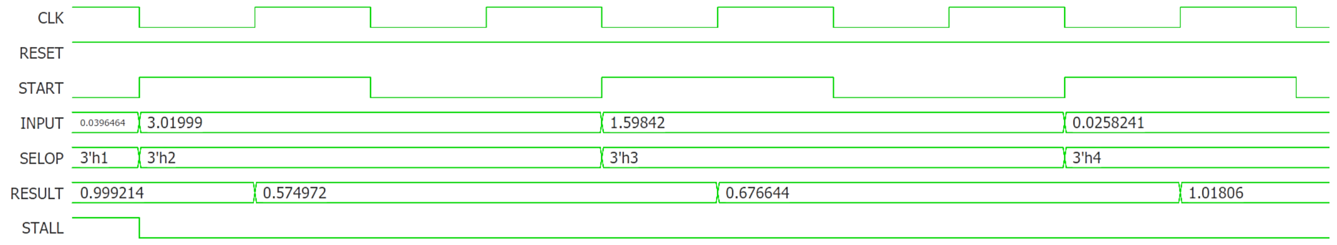


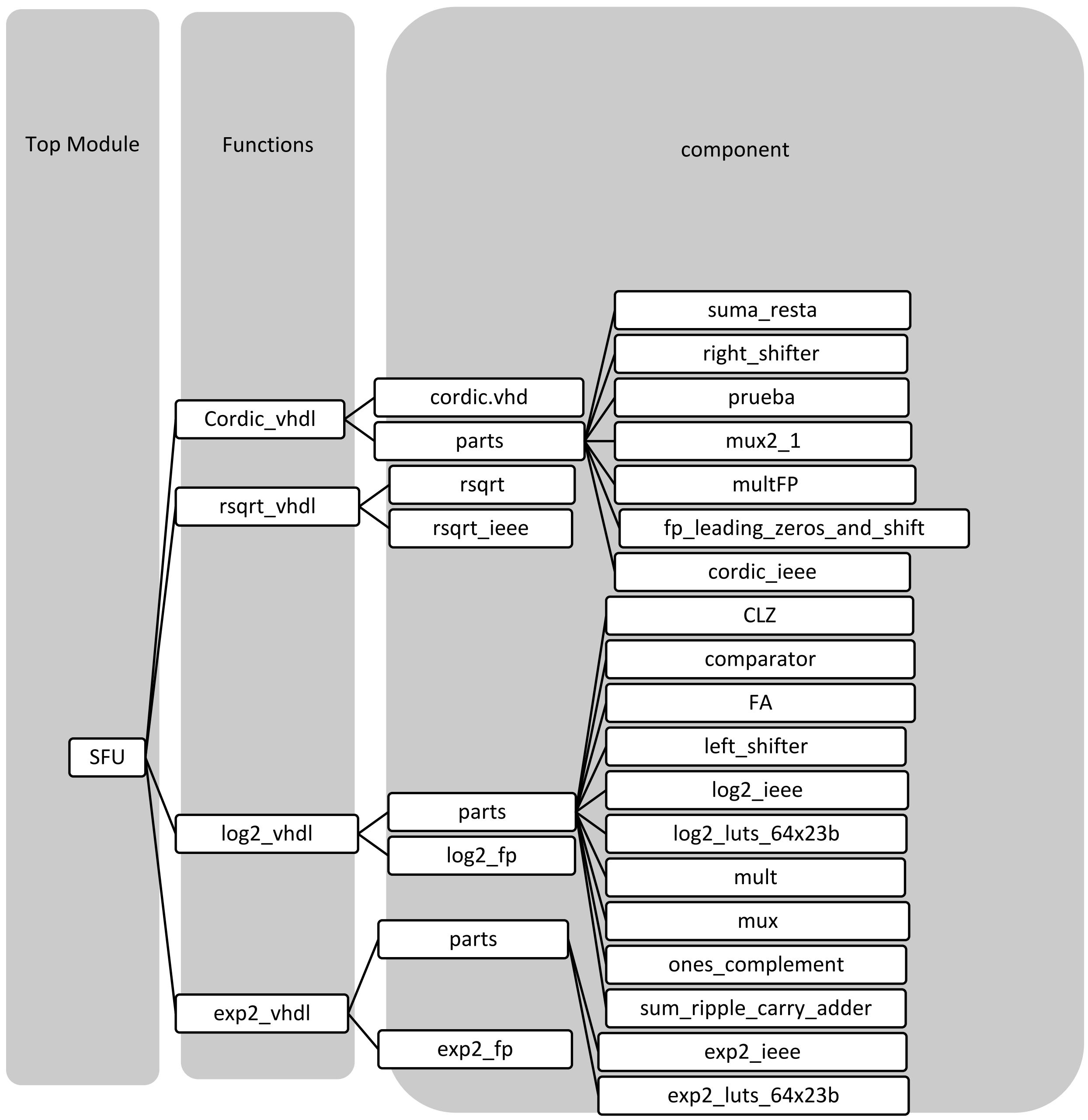
Figure 4. Waveforms for the $\frac{1}{\sqrt{x}}$, $\log_2 x$, and 2^x in the SFU.

The SFU was verified with 20,000 random data by operation, within the input interval presented in the next table.

TABLE III AVERAGE ERROR AND STANDARD DEVIATION FOR SFU USING RANDOM SET SAMPLES

Function	Input Interval
$\sin x$	$[0, \pi/2]$
$\cos x$	$[0, \pi/2]$
$log_2 x$	$[1,2)$
2^x	$[0,1)$
$\frac{1}{\sqrt{x}}$	$[1,4)$

In the folder of the project, you will find the next hierarchy of files:



Top Module
Functions
component
SFU
Cordic_vhdl
rsqrt_vhdl
log2_vhdl
exp2_vhdl
cordic.vhd
parts
rsqrt
rsqrt_ieee
parts
log2_fp
parts
exp2_fp
suma_resta
right_shifter
prueba
mux2_1
multFP
fp_leading_zeros_and_shift
cordic_ieee
CLZ
comparator
FA
left_shifter
log2_ieee
log2_luts_64x23b
mult
mux
ones_complement
sum_ripple_carry_adder
exp2_ieee
exp2_luts_64x23b