## Tutorial

# AXI4-Streaming to StellarIP Interface

4DSP LLC

Email: <a href="mailto:support@4dsp.com">support@4dsp.com</a>

This document is the property of 4DSP LLC and may not be copied nor communicated to a third party without the written permission of 4DSP LLC.

© 4DSP LLC 2014



### **Revision History**

Date	Revision	Revision
2014-01-23	Initial release	1.0



1	Intr	oduc	tion	. 5
2	Crea	ate tł	ne AXI FFT star	. 6
	2.1	Crea	ite the star in StellarIP	. 6
	2.1.	1	The following steps show how to create these two worm holes	. 7
	2.1.	2	Creating a new star	. 8
	2.1.	3	Creating the skeleton	11
	2.2	Addi	ing the FFT IPcore	12
	2.2.	1	Generate the FFT core	12
	2.2.	2	Update the VHDL wrapper file	15
	2.2.	3	Adding the required files to the .LST file	15
3	Inte	erface	e data wh_in/out to axis_32b_in/out	16
	3.1	Crea	ting the wh_in2axis_32b_out star	16
	3.1.	1	Create the wh_in2axis_32b_out star in stellar IP	17
	3.1.	2	Create the wh_in2axis_32b_out skeleton	18
	3.1.	3	Generate the Xilinx FIFO	19
	3.1.	4	Update the wh_in2axis_32b_out.vhd file and implement conversion logic	23
	3.2	Crea	ting the axis_32b_in2wh_out star	25
	3.2.	1	Create the axis_32b_in2wh_out star in stellar IP	25
	3.2.	2	Create the axis_32b_in2wh_out skeleton	25
	3.2.	3	Create the axis_32b_in2wh_out conversion logic	25
4	Inte	erface	e the command wormholes to axis	27
	4.1	Cont	trol/Status Interface of FFT core	27
	4.2	Crea	ting the cmd2axis_32b star	28
	4.2.	1	Create the cmd2axis_32b star in stellar IP	29
	4.2.	2	Create the cmd2axis_32b skeleton	30
5	Dat	a forı	matter	31
	5.1.	1	Create the axis_32b_real2complex star in stellar IP	32
	5.1.	2	Create the sip_axis_32b_real2complex skeleton	32
	5.1.	3	Create the sip_axis_32b_real2complex conversion logic	32
6	Gen	erate	e a Constellation	33
7	Gen	erate	e an ISE Project and Programming File	36
8	Soft	tware	Application	37
9	Stel	larIP	Troubleshooting	42
Α.	Sip_	_cmd	2axis_xfft example source code	44



Β.	Sip_axis_32b_real2complex example source code	46
C.	Software module: xilinx_fft	48
D.	Example constellation	50

## **1** Introduction

This document briefly describes how to interface between Xilinx Axis and 4DSP StellarIP interfaces. This tutorial introduces how to create a wormhole, star and constellation as well as connecting Axis interface to 4DSP StellarIP interface.

This tutorial is performed using an FMC104 on KC705 and Xilinx FFT core.

Following are required for this tutorial

- FMC104
- KC705
- Host computer with a gigabit Ethernet available
- Xilinx ISE
- Visual C++ 2012
- 4DSP BSP

Please refer the Xilinx user guides, 4FM Getting Started Guide and other 4DSP documents to make sure all this is properly installed.

First the tutorial will show how to wrap the Xilinx IP core into a star that can be used within stellar IP and then it will show how to integrate the newly created star into a standard StellarIP project. By default 4DSP does not use AXI4-Streaming (in this document referred to as AXI or AXIS) interfaces to communicate between stars (IP cores). This tutorial will show how to connect an AXIS interface to the standard StellarIP frame work by using some helper stars. The functional block diagram of the final constellation (FPGA programming file) is shown in the following figure.



Figure 1: Block diagram of the constellation



In order to use the FFT core from StellarIP it has to be wrapped into a star. This chapter will show how to generate the FFT core and how to wrap it into a star.

First we will create the star wrapper using the StellarIP tool. Typically the creation of a star comprises of the following steps:

- Create new wormholes (group of signals to connect between stars)
- Create new star
- Generate the star skeleton

After we have created the star skeleton we will create the FFT IP core using the Xilinx Core generator tool and add it into the star.

#### 2.1 Create the star in StellarIP

The first step in creating a star is to group the signals that have a specific functionality together into a wormhole. This will simplify the integration of the star into the constellation. As shown in Figure 2 the FFT core (generation of this core is described in section 2.2.1) has 6 distinct functional interfaces.



Figure 2: FFT core interface definition

As you can see there are four AXI stream interfaces, two inputs and two outputs. However they do not all share the same signals. The DATA AXI stream ports define the TLAST signal and the DATA output also defines the TUSER port. We could create four wormholes to support all these individual interfaces or we can define the wormhole in a way that we can use it for all AXI stream interfaces. The AXI stream defines the following signals

- tvalid, tready, tdata, tstrb, tkeep, tlast, tid, tdest and tuser
  - o tdata, tstrb, tkeep, tlast, tid, tdest and tuser are optional
  - Programmable *tdata, tid, tdest* and *tuser* widths
    - tstrb, tkeep width is tdata width/8

- Per port *aclk/aresetn* inputs (supports clock domain crossing)
- Per port aclken inputs (optional)

We will define a wormhole that comprises of all these signals apart from the *tid, tdest, aclken, clk* and *aresetn*. Xilinx IP typically does not use *tid* and *tdest*, the clock enable will be tied to be always enabled and the clock and reset will be connecting to the clock and reset wormholes that are coming from a centralized clock and reset controller. The programmable width will be set to 32 bits. In this tutorial we will not make the debug signals available on the star wrapper.

We will make sure, while creating the wrapper, to tie the unused inputs of the FFT core to a fixed one or zero.. A wormhole always comes in a pair, an output and input that can connect to each other. In the next section, we will define the wormhole contents as follows:

Wormhole name = axis_32b_in					
Wormhole direction = in					
name	direction	width			
tvalid	in	1			
tdata	in	32			
tlast	in	1			
tuser	in	32			
tready	out	1			
tstrb	out	1			
tkeep	out	4			

Wormhole name = axis_32b_out Wormhole direction = out				
name	direction	width		
tvalid	out	1		
tdata	out	32		
tlast	out	1		
tuser	out	32		
tready	in	1		
tstrb	in	1		
tkeep	in	4		

Table 1 32 bits axi stream wormhole pair

#### 2.1.1 The following steps show how to create these two worm holes

- 1. Launch StellarIP
- 2. Go to Library in the menu and click wormhole editor.



3. Click 'Add'

Add Clone	Edit Delete		
Name	Value	Direction	T
:l_uni_test	cl0_xp:4:in[cl0_xn:4:in]cl0_x_cl	External	1
:lkin	clkin:32:in	Input	
:lkout	clkout:32:out	Output	
:md_in	cmdin:64:in[cmdin_val:1:in	Input	
md_out	cmdout:64:out cmdout_val:1:out	Output	
:mdclk_in	cmdclk:1:in	Input	
:mdclk_out	cmdck:1:out	Output	
ext_adc110	ck_to_fpga_p:1:in ck_to_fpga	External	
ext_adr224	ext_clock:1:inlext_tringer:1:inls	External	78

4. Create the axis\_32b\_in wormhole. In order to add ports; fill in the port properties; *name*, *width* and *direction* and click 'Add'

lame: axis_32b_in			Direction:
ort Properties			
lame:		Width:	Direction:
			inout 🔹
	Add	Save	Delete
Name	Width	Direction	
tdata	32	in	
tkeep	4	out	
tlast	1	in	
tready	1	out	
tstrb	1	out	
tuser	32	in	
tvalid	1	in	

#### 5. Create the axis\_32b\_out wormhole

lame: axis_32b_out			Direction: Output
ort Properties Jame:		Width:	Direction:
			inout
Name	Add	Save	Delete
Name	Add	Save	Delete
Name tdata	Add Width 32	Save Direction out	Delete
Name tdata tkeep Hast	Add Width 32 4	Direction out in	Delete
Name tdata tkeep tlast tready	Add Width 32 4 1	Save Direction out in out in	Delete
Name tdata tkeep tlast tready tready	Add Width 32 4 1 1 1	Save Direction out in out in in	Delete
Name tdata tkeep tlast tready tstrb tuser	Add Width 32 4 1 1 1 1 32	Save Direction out in out in in out	Delete

6. Close the wormhole editor

#### 2.1.2 Creating a new star

Now that the wormholes are defined we can create the star that will wrap the FFT core.

1. Click the 'Edit' button on the star picker window to edit the star library



r1.0

the Library Shellar	The President	Marks.	_	_	_	_	_	-
we uprary some	ar besque	rep						
🖸 😂 🖬 🛛 🖄	XUB	NO OF	22	TX	- 1	2	*	а,
Search Star:								
		_						
Show								
Of All stars	C Recently pi	cked						
Lbraries:	e	dR						
Roument								
		- 18						
		- 11						
		-1						
No. adc110		_						
nip_adc110 nip_adc224		-						
nip_adc110 nip_adc224 nip_adc250		A						
sip_adc110 sip_adc224 sip_adc250 sp_adc250 is		41						
sip_adc110 sip_adc224 sip_adc250 sip_adc250_ls sip_adc8w		41						
sp_adc110 sp_adc224 sp_adc250 sp_adc250_ls sp_adc80 sp_blact_h_Wo		41						
10p_adc110 10p_adc224 10p_adc250 10p_adc250_31 10p_adc250_31 10p_adc250_31 10p_adc250_31 10p_adc250_31		41						
102_adc210 102_adc224 102_adc2250 102_adc250_b 102_adc250		-						
102_sdc110 102_sdc224 102_sdc250 102_sdc250_1100_1100_1100_1100_1100_1100_1100_1		-						

2. In the menu click Star->New Star and insert the name for the new star, for example sip\_xfft\_wrapper, in the widow that pops-up

😽 StellarIP - F:\SYN\FW\star_lib\current		E
File Library Star Window Help		
🏠 📴 🔛 New Star 👘 😢 🧧	L + ×   = 1   _   <mark>  _   <mark>  _   =</mark>   <b>4.</b> X &amp;   <i>8</i> * *  </mark>	
Search Star: Edit Star Delete Star Star Properties		
Show	F:\SVN\FW\star_lib\current	
All stars Duplicate Star ked	sip_adc110 sip_adc110	
Libraries: Edit	-	
[⊠current	sip_adc224 sip_adc224	
	sip_adc250	

3. Draw a rectangle

📑 StellarIP - [Symbol - sip_axis_32b_in_to_v	vh_out]
📑 File Edit Library Stellar IP Desig	ner Window Help
🔁 💕 🖬 🛃 🕌 🕌 🛍 🖉	<u>a</u> 1 + ×   <sup>™</sup> ]   _2   <b>≩</b>   <b>⊒</b>   <b>X</b>   <b>3</b> * *   A ▶ C □ /
Search Star:	
Show	
All stars     Recently picked	
Libraries:	
<b>I</b> 4DSPmain	

4. Add pins to the star, each pin represents a wormhole. We will need 6 pins, 4 inputs on the left and 2 outputs on the right.

😹 StellarIP - [Symbol - sip_new]		_ & ×
🔜 File Edit Library Stellar IP Designer	Window Help	_ <u>-</u>
🛯 🏠 📓 🛃 🖓 🐇 🖏 🙇 💌 🕫	-∭alt+× ‴il <mark>izi ☆</mark> ╤╡९ <b>×</b> ∖; #३	
Search Star:	30 40 <b>1</b> 0 70 80 90 100 110 120 130 140 150 160 170 180	190 200
		8
Show		
<ul> <li>Anstars</li> <li>Recently picked</li> </ul>		- 1 - 1

5. After placing the pin you can double click on the pin to change its properties. For the output pins we set the 'left' option and we assign the names and types: *cmdclk* = cmdclk\_in type, *rst*  = rstin type, data\_in = axis\_32b\_in type, data\_out = axis\_32b\_out type, status = axis\_32b\_out type, control = axis\_32b\_in type.

For this tutorial, all stars will run synchronously using the cmd\_clk, generated by the Ethernet MAC star (sip\_mac\_engine). Therefore we will only create stars with one single clock input. If the FFT processing needs to be accelerated, one could choose to implement another clock scheme. This requires stars or custom firmware blocks that support cross clock region interfaces.

#### The pin tool options dialog:

Pin Tool Options							
Name	cmd_out		Number	1			
Wormhole	cmd_out		Select				
© <u>∪</u> p	© <u>D</u> own ⊚ <u>L</u> eft	© <u>R</u> ight	Length:	30			
Shape	Normal 👻	Electrical	Output	•			
			Input				

The Star symbol will look like:



6. Pressing <ctrl+s> or the save button opens the star properties window. Press the 'Save' button and we have created the sip\_xfft\_wrapper star

#### 2.1.3 Creating the skeleton

Now that we have created the star in StellarIP we need to create the support files as well. The easiest way is to let StellarIP create the folder structure and vhdl wrapper files. Click the edit button in the star picker and find the sip\_xfft\_wrapper star. Right click on the sip\_xfft\_wrapper star and click "Generate HDL Skeleton (No Registers)"

StellarIP - [F:\SVN\FW\star_lib\current	ntj	_ <del>_</del> <del>_</del> <del>/</del> ×
🔜 File Library Star Window Help		_ <del>_</del> # ×
🏷 😂 🔜 🖂 🐇 🖦 📖 🕫 🌣	/ <u>≜ 1 + ×   <sup>™</sup> i   ∠   <b>☆   =   4 ×  </b>≥   <b># →</b></u>	
Search Star:	<pre>sip_fc6603_ddr2_fifo sip_fc6603_ddr2_fifo</pre>	-
All stars     C Recently picked	<pre>sip_fc6603_host_if sip_fc6603_host_if</pre>	
Libranes:	sip_ff Edt Star sip_fft Star Properties	
ab. unt_sid ex ab: endb.a. end_s.a	sip_fi New Star sip_fii Clone Star Delete Star	
sp_adc110	sip_fi Generate HDL Skeleton sip_fi1 Generate HDL Skeleton (No registers)	
sip_adc224	sip_fifo_tester	

Figure 3 FFT wrapper skeleton options

This creates the star folder structure in the library root folder. Go to "..\sip\_xfft\_wrapper\". There will be 4 folders; sip\_files, isim, simulate and vhdl.

The *sip\_files* folder has three files

- sip\_xfft\_wrapper.lst = this file lists the paths to all the source files that should be used during the creation of the Xilinx ISE/VIVADO project.
- sip\_xfft\_wrapper.nfo = this file holds the star identification number and the version number. Both are four digit hexadecimal values.
- sip\_xfft\_wrapper.vhd = this file defines the architecture of the star. This file will be modified to instantiate the FFT core and glue logic.

The *vhdl* folder should be used to store all additional design files.

The *simulate* folder can be used to hold test bench specific files, but is not used in this tutorial.

The *isim* folder can be used to hold the isim project, but is not used in this tutorial.

#### 2.2 Adding the FFT IPcore

Now that we have the wrapper files available for the FFT core we can generate the IP core itself and instantiate it in the top level wrapper vhdl file as well as adding the required files into the .LST file.

#### 2.2.1 Generate the FFT core

- 1. Open the 'CORE generator' under Start->All Programs->Xilinx Design Tools->ISE Design Suite->ISE Design tools -> 64/32-bit Tools.
- 2. New Project and save the coregen project file to "..\sip\_xfft\_wrapper\"
- 3. Select the parts for KC705 on the pop-up window as shown in the following figure and click 'Ok'.

Part								
Select the part for your project:								
Family	Kintex7	•						
Device	xc7k325t	•						
Package	ffg900	•						
Speed Grade	-2	•						

4. In the IP Catalog, find the Fast Fourier Transform 8.0 and customize it by double clicking the function.

IP Catalog				ć	7 ×
View by Function View by Name					
Name	Version	AXI4	AXI4-Stream	AXI4-Lite	*
<ul> <li>Automotive &amp; Industrial</li> <li>AXI Infrastructure</li> <li>BaseIP</li> <li>Basic Elements</li> <li>Communication &amp; Networking</li> <li>Debug &amp; Verification</li> <li>Digital Signal Processing</li> <li>Building Blocks</li> <li>Filters</li> <li>Modulation</li> <li>Transforms</li> <li>FFTs</li> <li>FFTs</li> <li>Fast Fourier Transform</li> </ul>	<b>7.1</b> 8.0		AXI4-Stream		Е
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	1.0				
📄 📂 📂 Waveform Synthesis					

5. Name the component as *xfft\_v8\_0\_1ch\_blk\_float*. In this tutorial, 1 channel N-point pipelined FFT with a block fixed point will be used.





#### Copy the settings as shown in the following figures:



- 6. Click generate
- After generation, move the *xfft\_v8\_0\_1ch\_blk\_float.vhd/.ngc/.xco* files to
   "..\sip\_fft\_wrapper\vhdl\k7\Xilinx\". Other files that are generated by the tool are not
   directly required by ISE nor StellarIP and can be deleted

#### 2.2.2 Update the VHDL wrapper file

Declare and instantiate the FFT IP core in *sip\_xfft\_wrapper.vhd*. The connection between the FFT IP core signals and the *axis\_32b\_in* and *axis\_32b\_out* wormhole signals have to be made as shown in the following table.

wormhole	signal	connects to		wormhole	signal	connects to
	tvalid	s_axis_data_tvalid			tvalid	m_axis_status_tvalid
	tdata	s_axis_data_tdata			tdata[70]	m_axis_status_tdata
data in	tlast	s_axis_data_tlast			tdata[318]	force 0x000000
	tuser	not used			tlast	force '0'
	tready	s_axis_data_tready		status	tuser	force 0x00000000
	tstrb	force '0'			tready	m_axis_status_tready
	tkeep	force 0x0			tstrb	not used
					tkeep	not used
	tvalid	m_axis_data_tvalid			tvalid	s_axis_config_tvalid
	tdata	m_axis_data_tdata			tdata[70]	s_axis_config_tdata
	tlast	m_axis_data_tlast			tdata[318]	not used
data out	tuser	m_axis_data_tuser		control	tlast	not used
	tready	m_axis_data_tready		control	tuser	not used
	tstrb	not used			tready	s_axis_config_tready
	tkeep	not used			tstrb	force '0'
					tkeep	force 0x0

#### Table 2 sip\_xfft\_wrapper wormhole connections

The clk and rst wormhole signals map to the FFT IP core as shown in Table 3.

wormhole	signal	connects to
cmdclk_in	cmdclk	Aclk
	rst_in[2]	aresetn (after inverting the signal)
rst	rst_in[10]	not used
	rst_in[313]	not used

Table 3 sip\_xfft\_wrapper clk and rst wormhole connections

#### 2.2.3 Adding the required files to the .LST file

Add the source files, located in the "..\vhdl\xilinx" folder to *sip\_fft\_wrapper.lst* using relative paths:

../vhdl/xilinx/xfft\_v8\_0\_1ch\_blk\_float.xco
../vhdl/xilinx/xfft\_v8\_0\_1ch\_blk\_float.vhd
../vhdl/xilinx/xfft\_v8\_0\_1ch\_blk\_float.ngc

## 3 Interface data wh\_in/out to axis\_32b\_in/out

4DSP has chosen to standardize communication between stars as much as possible. In most cases, data is communicated from a *wh\_out* wormhole to a *wh\_in* wormhole. These wormholes are defined as 64 bits data, 1 bits data valid and 1 bit stop. On each clock cycle where the data valid is asserted, data is transferred regardless of the status of the stop signal. However, the transmitting star is not allowed to start sending data when the stop signal is asserted and should stop sending data within 8 clock cycles after the stop signal was asserted by the receiver.

It is not possible to interface 4DSP standard wormhole to the axis\_32b wormholes directly. The main differences between the two interfaces are:

- 4DSP standard wormhole has 64-bit data width, the AXI interface is 32-bit
- 4DSP standard output wormhole asserts *data\_valid* when *data\_stop* is '0' and the *data* is valid. When the data\_stop is asserted the data\_valid should be de-asserted within 8 clock cycles. Valid data is transferred on each clock cycle where data\_valid is asserted. The AXI protocol is different, the master side asserts *tvalid* when *tdata* is valid but waits until the slave accepts the data by asserting *tready*. Only when tdata and tready are asserted at the rising edge of the clock the next data word is placed on the data bus.

We need to create two conversion stars. A *wh\_in* to *axis\_32b\_out* conversion star, this start receives data from a 4DSP standard wormhole and writes it to the AXI slave. The other star is an *axis\_32b\_in* to *wh\_out* conversion star, receiving data from the AXI master and writes the data to a connected 4DSP standard wormhole via the *wh\_out* output.

#### 3.1 Creating the wh\_in2axis\_32b\_out star

The wh\_in2axis\_32b\_out star contains a standard 4DSP wormhole input (referred to as *wh\_in*), the axis\_32b wormhole output (referred to as *axis\_32b\_out*) and the standard reset and cmdclk types.

Name	Wormhole
data_in	wh_in
data_out	axis_32b_out
clk	cmdclk_in
rst	rst_in

#### Table 4 wh\_in2axis\_32b\_out star wormholes

The main features of the star will be:

- Convert 64-bit 4DSP wormhole data width to 32-bit AXI data width
- Change the 4DSP wormhole write protocol to the AXI protocol

The implementation of this star is based on a Xilinx FIFO generated with the Core Generator.



#### 3.1.1 Create the wh\_in2axis\_32b\_out star in stellar IP

In this subsection the steps from section 2.1.2 are repeated to create the wh\_in2axis\_32b\_out star. The wormhole types required for this star are already created in previous sections and available.

1. Click the edit button on the star picker window to edit the star library.



2. Choose the Star->New Star and insert the name for the new star, for example sip\_wh\_in2axis\_32b\_out, in the widow that pops-up

🚮 StellarIP - F:\SYN\FW\star_lib\current		- 18
File Library Star Window Help		
🎦 😂 🛃 New Star 👘 💓 🙍	L ┿ ×   Ლ 8   ∠   🙀 💐 🍕 X &   # % ↔	
Search Star: Edit Star Delete Star Star Properties		
C All stars Duplicate Star	BELSWN/FW/star_lib/current     sip_adc110     sip_adc110	
Libraries: Edk	sip_adc224 sip_adc224	
	sip_adc250	

#### 3. Draw a rectangle

🔜 StellarIP - [Symbol - sip_axis_32b_in_to_	wh_out]
🔜 File Edit Library Stellar IP Desig	ner Window Help
🎦 💕 🛃 🛃 🕌 🕷 🗠 🗠	ª l + ×   ™     _2   😭 🗐 🍳 X 🐼 / / * * / A Þ C 🗆 <
Search Star:	0 10 20 30 40 50 60 70 80 50 100
Show	
Libraries: Edit	

4. Add pins to the star, each pin represents a wormhole. We will need 5 pins, 4 inputs on the left and 1 outputs on the right.





5. After placing the pin you can double click on the pin to change its properties. For the output pins we set the 'left' option and we assign the names and types: *data\_in\_clk* = cmdclk\_in type, *rst* = rstin type, *data\_in* = wh\_in type, *data\_out* = axis\_32b\_out type.



6. Pressing <ctrl+s> or the save button opens the star properties window. Press the 'Save' button and we have created sip\_wh\_in2axis\_32b\_out star.

#### 3.1.2 Create the wh\_in2axis\_32b\_out skeleton

Click edit button in the star picker and find the sip\_wh\_in2axis\_32b\_out star star. Right click on the sip\_wh\_in2axis\_32b\_out star star and click "Generate HDL Skeleton (No Registers)".

StellarIP - [D:\Projects\FW\star_lib\4DS	Pmain]		
🔜 File Library Star Window Help			
🔁 💕 🖬 🥵 🐰 🖦 🛝 🕫 າ	🛯 🕹 🗋 🛨 ×   🏧 🖡 🔀 🙀 🗙 🗡	<b>≥   # &gt; +</b>   A   _=   □ O ≱   <i>C</i>	
Search Star:	sip_vp780_ddr3_fifox4		
sip_wh	sip vp780 host if		
Show	sip_vp780_host_if		
All stars     Recently picked			
Libraries: Edit	sip_wh_in2axis_32b_out sip_wh_in2axis_32b_out	Edit Star Symbol	45,342,688 0, 63875 407,008 0, 74 0,008 0,
<b>I</b> 4DSPmain	<pre>sip_xx720_41_host_if sip_xx720_41_host_if</pre>	Star Properties	
sip_wh_in2axis_32b_out	<pre>sip_xx720_81_host_if sip_xx720_81_host_if</pre>	Clone Star Delete Star	
	<pre>sip_xx720_ddr3_fifo sip_xx720_ddr3_fifo</pre>	Generate HDL Skeleton Generate HDL Skeleton (No registers)	
	<pre>sip_xx720_reload sip_xx720_reload</pre>		
	sip_zc702_host_if		100,0 1000,0 1,000 100,0 100
😑 Console 👍 Warnings 🗛 E	rrors All		

#### Figure 4 Generation HDL Skeleton (No Registers)



- 1. Open the 'CORE generator' under Start->All Programs->Xilinx Design Tools->ISE Design Suite->ISE Design tools -> 64/32-bit Tools.
- 2. New Project and save the coregen project file to "..\sip\_wh\_in2axis\_32b\_out\"
- 3. Select the parts for KC705 on the pop-up window as shown in the following figure and click 'Ok'.

Part elect the part for your project:						
Family Kintex7						
Device xc7k325t						
Package ffg900 💌						
Speed Grade -2						

#### 4. Double click on the FIFO Generator Function

🤤 🗴 File	ilinx CORE Generator - D:\Projects\FW\star Project View Manage IP Help	r_lib\sip_w	h_in2a	xis_32b_out\co	regen.cgp				
	) 🖻 🖥 💷 🖾 🐼 👔 😵	0							
 I	IP Catalog								₽×
14 131	View by Function View by Name								
୯-୩ ଅନ୍ୟ	Name	Version	AXI4	AXI4-Stream	AXI4-Lite	Status	License	Vendor	Library
	<ul> <li>AXI Infrastructure</li> <li>BaseIP</li> <li>Basic Elements</li> <li>Communication &amp; Networking</li> <li>Debug &amp; Verification</li> <li>Digital Signal Processing</li> <li>Embedded Processing</li> <li>FPGA Features and Design</li> <li>Math Functions</li> <li>Memories &amp; Storage Elements</li> <li>FIFOs</li> </ul>	0.2	) A VIA	AVIA Stroom		Deduction		viliau com	
	Pro Generator     Pro Gen	9.3	AX14	AXI4-Stream	AXI4-Lite	Production		xilinx.com	īρ

5. Copy the settings of page 1 to 7 as shown below. The component name shall be *wh2axis\_fifo*.



Page 1:



V FIFO Generator	Contraction a familie a	Incoments a title a lastest		
View Documents				
IP Symbol		×		
		logics FIFO Ge	nerator	xilinx.com:ip:fifo_generator:9.3
CLK		FIFO Implementation		
RST →		Choose the FIFO implementation from one of the	he following:	
SRST			Supported Features	•
		Read/Write Clock Domains Me	emory Type (1) (2) (3) (4) (5)	
WR_RST		Common Clock (CLK) Block	ock RAM X X X	
WR_CLK		Common Clock (CLK) Dis	istributed RAM X	
WR EN		Common Clock (CLK) Sh	hift Register	
PROG_FULL_THRESH[9:0]	C RD_RST	Common Clock (CLK)	uilt-in ETEO X X X X	
G_FULL_THRESH_ASSERT[9:0]	CLK			E
IG_FULL_THRESH_NEGATE[9:0]		Independent Clocks (RD_CLK, WR_CLK) Block	OCK RAM X X X X	
INJECTSBITERR	-RD_EN	<ul> <li>Independent Clocks (RD_CLK, WR_CLK) Dis</li> </ul>	stributed RAM X	
INJECTDBITERR	PROG_EMPTY_THRESH[1	Independent Clocks (RD_CLK, WR_CLK) Bu	uilt-in FIFO X X X X	
FULL←	PROG_EMPTY_THRESH_	ISER		
ALMOST_FULL ←	PROG_EMPTY_THRESH_I	(1) Non-symmetric aspect ratios (different read	d and write data widths)	
PROG_FULL	→ SBITERR	(2) First-Word Fall-Through		
	DBITERR	(3) Uses Built-in FIFO primitives		
WR DATA COUNTI9:01	ALMOST EMPTY	(4) ECC support		
	> PROG_EMPTY	(5) Dynamic Error Injection		
	> VALID	(-, -,,,,,		
		Synchronizer Option		
	RD_DATA_COUNT[10:0]	Synchronization Stages across Cross Clock Dom	main Logic 2 🔹	
	DATA_COUNT[9:0]			
				-
		Datasheet < Back Pa	Page 2 of 7 Next > Genera	te Cancel Help
•	III			



Dago 2.

FIFO Generator	Company a familie a finances	a titles as Sportanti	1. Au	
/iew Documents				
P Symbol	5 ×		_	
		Logic FIFO G		m:ip:fifo_generator:
CLK		Read Mode		
RST		Standard FIFO		
SRST>		First-Word Fall-Through		
WR RST-		Built-in ETEO Ontions		
WR_CLK		built in Fir O Options		
DIN[63:0]				
WR_EN		The frequency relationship of WR_CLK and R	D_CLK MUST be specified to generate the corr	rect
PROG_FULL_THRESH[9:0]	E RD_RST	implementation.	D 4 4000	
FULL_THRESH_ASSERT[9:0]	RD_CLK	Read Clock Frequency (MHZ)	Range: 11000	
INJECTSBITERR		Write Clock Frequency (MHz) 1	Range: 11000	
	PROG_EMPTY_THRESH[10:0]	Data Port Parameters		
FULL	PROG_EMPTY_THRESH_ASSER			
ALMOST_FULL -	PROG_EMPTY_THRESH_NEGAT	Range: 1,2,310	24	
PROG_FULL ←	> SBITERR	Write Depth 1024   Actual Write Dept	h: 1024	
		Read Width 32 👻		
WR DATA COUNTI9:01	ALMOST EMPTY	Read Depth 2048 Actual Read Dept	h: 2048	
/ /		······		
	> VALID	Implementation Options		
		Enable ECC		
	RD_DATA_COUNT[10:0]	Use Embedded Registers in BRAM or FIFC	(when possible)	
	DATA_COUNT[9:0]			
		Read Latency (From Rising Edge of Read Clock		
			): 0	
		Datasheet         < Back	): 0 Page 3 of 7 <u>N</u> ext > <u>G</u> enerate <u>C</u>	ancel <u>H</u> elp
( III	Þ	Datasheet         < gack	): 0 Page 3 of 7 <u>Next &gt; Generate</u> <u>C</u>	Cancel Help
age 4:		Datasheet <u>Sack</u>	: 0 Page 3 of 7 <u>Next &gt; G</u> enerate <u>C</u>	ancel <u>H</u> elp
age 4:		Datasheet Sack	:: 0 Page 3 of 7 <u>Next &gt; <u>G</u>enerate <u>G</u></u>	ancel Help
" age 4: FIFO Generator		Datasheet     < gack	: 0 Page 3 of 7 <u>Next &gt;</u> <u>Generate</u> <u>C</u>	Cancel Help
mage 4: FIFO Generator Ziew Documents	Norma, 34 (p. , a	Datasheet     < Back	: 0 Page 3 of 7 <u>Next &gt; Generate</u> <u>C</u>	ancel Help
minimized and a second and a se			: 0 Poge 3 of 7 <u>Next &gt; Generate</u> <u>C</u>	ancel Help
#IFO Generator /Iew Documents 2 Symbol			i: 0 Page 3 of 7 <u>Next &gt; Generate</u> <u>C</u> Cenerator <sub>xilinx.</sub>	com:ip:fifo_generat
mage 4: FIFO Generator //ew Documents 2 Symbol	ð x		: 0 Page 3 of 7 <u>Next &gt; Generate</u> <u>G</u> Generator xilinx.	com:ip:fifo_generat
FIFO Generator Agew Documents 2 Symbol		Logic RF FIFO C	Page 3 of 7 Next > Generate Generate Senerator xilinx.	com:lp:ffo_generat
" age 4: FIFO Generator //ew Documents 2 Symbol		LogiCRF FIFO C Optional Flags	: 0 Page 3 of 7 <u>Next &gt; Generate</u> <u>C</u> Generator xilinx. Almost Empty Flag	com:lp:fifo_generat
" age 4: FIFO Generator /jew Documents Symbol		Datasheet     Sack       LogiCRF     FIFO G       Optional Flags     Almost Full Flag       Handshaking Options     Handshaking Options	: 0 Page 3 of 7 <u>Next &gt; Generate</u> <u>Generate</u> Senerator xilinx.	com:lp:fifo_generat
m age 4: FFO Generator //ew Documents > Symbol		Datasheet     Start Control       Logi     FIFO       Optional Flags     Almost Full Flag       Handshaking Options     With Port Handshaking	: 0 Page 3 of 7 <u>Next &gt; Generate</u> <u>C</u> Senerator xilinx. Almost Empty Flag	com:ip:fifo_generat
mage 4: FIFO Generator //ew Documents > Symbol		Logi CRE FIFO G Optional Flags Almost Full Flag Handshaking Options Write Port Handshaking	Page 3 of 7 Next > Generate C	cancel Help
mage 4: FIFO Generator //ew Documents 2 Symbol CUK→ RST→ SRST→		Datasheet     < Back	Page 3 of 7 Next > Generate G Senerator xilinx. Almost Empty Flag	com:ip:fifo_generat
mage 4: FFO Generator //ew Documents 2 Symbol CLK→ RST→ SRST→ WR, RST→		Datasheet       < Back	Page 3 of 7 Next > Generate G Cenerator xilinx. Almost Empty Flag Overflow Flag Overflow (Write Error)	com:ip:fifo_generat
m age 4: FFO Generator //ew Documents ≥ symbol Cut → Rst → SRst → WR_Rst → WR_Cut →		Logi CRF FIFO C Optional Flags Almost Full Flag Handshaking Options Write Acknowledge Flag Write Acknowledge @ Active High	:: 0 Page 3 of 7 Next > Generate G Generator xilinx. Almost Empty Flag Overflow Flag Overflow Vinte Error) Overflow Vinte Error) Overflow Vinte Error)	com:lp:ffo_generat
m age 4: FFO Generator //ew Documents <sup>2</sup> Symbol CUK→ RST→ SRST→ WR_RST→ WR_CK→ DIN(RS 0)→		Datasheet     < Back	i: 0 Page 3 of 7 Next > Generate G Generator xilinx. Almost Empty Flag Overflow Flag Overflow Vinte Error) Active High Active High Active High	com:lp:fifo_generat
" age 4:  FIFO Generator  (ew Documents  > Symbol  CLK  RST→  SRT→  WR_CK→  DIN(R3.0)→  WR_EN→		Datasheet     < Back	Page 3 of 7 Next > Generate G Senerator xilinx. Almost Empty Flag Overflow Flag Overflow (Write Error) Active High Active Low	com:ip:fifo_generat
"" age 4: FIFO Generator //ew Documents > Symbol CLK→ RST→ SRST→ WR_CK→ DIN(83.0)→ WR_S0→ PRO0_FULL_THRESH(9.0)→		Logi CRF FIFO C Optional Flags Optional Flags Almost Full Flag Handshaking Options Write Port Handshaking Write Acknowledge Flag Write Acknowledge Active High Active Low Read Port Handshaking	Page 3 of 7 Next > Generate G Cenerator xiinx. Almost Empty Flag Overflow Flag Overflow Flag Overflow (Write Error) @ Active High Active Low	com:ip:fifo_generat
m age 4: FFO Generator //ew Documents 2 Symbol CLK→ RST→ SRST→ WR_CLK→ UN(RS1)→ WR_CLK→ DIN(RS1)→ WR_CLK→ DIN(RS1)→ PRO0_FULL_THRESH(S0)→ FULL_THRESH(ASSERT[S0]→	a × a × a ×	Datasheet       < Back	Page 3 of 7 Next > Generate G Generator xilinx. Almost Empty Flag Overflow Flag Overflow Flag Overflow Flag Overflow Flag Underflow Flag Overflow Flag	com:lp:fifo_generat
" age 4:  FFO Generator  //ew Documents  > Symbol  CLK  RsT  SRST  WR_RST  WR_RST  WR_CLK  DIN(83.0)→  WR_CLK  PROQ_FUL_THRESH_NSSERT[9.0)→  FUL_THRESH_NSSERT[9.0)→  FUL_THRESH_NST FUL_THRESH_NSSERT[9.0)→  FUL_THRESH_NSSERT[9.0)→  FUL_THRESH_NSSERT[9.0)→ FUL_THRESH_NSSERT[9.0)→ FUL_THRESH_NSSERT[9.0)→ FUL_THRESH_NSSERT[9.0)→ FUL_THRESH_NST FUL_THRESH_NST FUL_THRESH_NSSERT[9.0)→ FUL_THRESH_NST		Logi CRF FIFO C Optional Flags Almost Full Flag Handshaking Options Write Acknowledge Flag Write Acknowledge Attive High Active Low Read Port Handshaking Valid Flag	Page 3 of 7 Next > Generate G Generator xilinx. Almost Empty Flag Overflow Flag Overflow Virite Error) Active Low Underflow Flag	com:lp:fifo_generat
age 4: IFFO Generator //ew Documents > Symbol	BD_RST ←RD_RST ←RD_CLK → DOUT[31:0] ← RD_FN → RD_FN	Datasheet       < Back	i: 0 Page 3 of 7 Next > Generate G ienerator xilinx. Almost Empty Flag Overflow Flag Overflow Vinte Error) Active High Active Low Underflow Flag Underflow Flag	com:ip:fifo_generat
	BD_BAT ⇒ BD_CLX ⇒ DOU[11:0] ⇒ RD_CLX ⇒ DOU[11:0] ⇒ RD_EN ⇒ RD_EN	Datasheet       < Back	Page 3 of 7 Next > Generate G Senerator xilinx. Almost Empty Flag Overflow Flag Overflow (High Active High Underflow (Read Error) (a) Active High Active High	com:ip:fifo_general
"" age 4:     "      "	B × B_RST → RD_RST → RD_CK → DOUTB1:0] → RD_EN → RD_EN → RD_EN → RD_EN → RD_EN → RD_EN → RD_EN → RD_EN → RD_CK → RD_RST → RD_RST	Patasheet  Patasheet  FIFO C  Optional Flags  Amost Full Flag  Handshaking Options  Write Acknowledge Flag  Write Acknowledge  Active High Active Low  Read Port Handshaking  Valid (Read Acknowledge)  Active High Active Low  Active Low  Active High Active Low  Active High Active Low  Active Lo	Page 3 of 7 Next > Generate G Senerator xilinx. Almost Empty Flag Overflow Flag Overflow (Write Error) Active High Active Low Underflow (Read Error) Active Low	com:lp:fifo_generat
mage 4: IFO Generator //ew Documents 2 Symbol CLK→ RST→ SRST→ WR_CLK→ DIN(83.0)→ WR_CLK→ DIN(83.0)→ WR_CLK→ DIN(83.0)→ WR_CLK→ DIN(83.0)→ WR_CLK→ DIN(83.0)→ FULL_THRESH_NECATE(9.0)→	B B B B B C C C C C C C C C C C C C	Datasheet       < Back	Page 3 of 7 Next > Generate Generator xilinx.	com:ip:fifo_generat
mage 4: FIFO Generator View Documents P Symbol	BD_RST ← RD_RST ← RD_CK → DOUT[31:0] ← RD_EN ← RO_EMPT_THRESH_NEONT → SBITERR → DBITERR	Datasheet       < Back	Page 3 of 7       Next >       Generate       C         Senerator       xilinx.         Almost Empty Flag       Overflow Flag         Overflow Flag       Overflow Flag         Overflow Kligh       Active Low         Underflow Flag       Underflow Flag         Underflow Klag       Underflow Klag         Underflow Klag       Underflow Klag         Overflow Klag       Underflow Klag         Overflow Klag       Underflow Klag         Overflow Klag       Underflow Klag	com:lp:fifo_generat
CLK     FIFO Generator     View Documents     P Symbol     CLK     RST     SRST     SRST     WR_RST	AD_RST AD_RST AD_CLK → DOUT[1:0] ← RD_CM → DOUT[1:0] ← RD_EM ← RO_EMPTY_THRESH_VEGAT → SUITERR → DUTERR → DUTERR	Datasheet       < Back	Page 3 of 7 Next > Generate G Senerator xilinx. Almost Empty Flag Overflow Flag Overflow (High Active High Active Low Active Low	com:ip:fifo_generat
	B x ⇒ RD_RST ⇒ RD_CK ⇒ RD_CK ⇒ RD_CV ⇒ RD_EN ⇒ RO_EMPTy_THRESH_NEGAT ⇒ BRITER ⇒ DBITERR ⇒ DBITERR ⇒ DBITERR ⇒ ALMOST_EMPTY	Datasheet       < Back	Page 3 of 7 Next > Generate G Senerator xilinx. Almost Empty Flag Overflow Flag Overflow (Write Error) (a) Active High Active Low Underflow (Read Error) (a) Active High Active Low Double Bit Error Injection	com:ip:fifo_generat

Datasheet

→RD\_DATA\_COUNT[10:0]

<u>Back</u> Page 4 of 7 <u>Next</u> <u>Generate</u> <u>Cancel</u> <u>Help</u>



Page 5:

View Documents           IP Symbol           CLX           RST           SST           WR_RST           WR_LCX           DIN(33.0)           WR_EN           PR00_FULL_THRESH(90)           C_FUL_THRESH_VEGATE(15.0)           INUECTBRIERR           INUECTBRIERR           FR0_ENTY_THRE           PR00_EULEC           PR00_EUNTY_THRE           PR00_EUNTY_THRE           VILCOTBRIERR           PR00_EUNTY_THRE           VILC           PR00_EUNTY_THRE           VILC           PR00_EUNTY_THRE           VILLC           PR00_EUNTY_THRE           VILC           PR00_EUNTY_THRE           PR00_EUNTY_THRE           VIR_ACK           OUERTLOW           WR_ACK           OUERTLOW           DBITERR           WR_DOK           OUERTLOW           PULT           VILT			
P Symbol CUX→→ RST→→ SRST→→ SRST→→ WR_RT→→ WR_RT→→ WR_RT→→ WR_RT→→ WR_RT→→ WR_RT→→ WR_RT→→ PRO_FUL_THRESH_00→ G_FUL_THRESH_00→ RD_RT RD_RT RD_CLX G_FUL_THRESH_NSARTE00→ FRO_FUL HUECTBRTRR→→ FRO_EMPTY_THRE PRO_EMPTY_T			
CUX →→ RST →→ SRST →→ SRST →→ WR_RST →→ WR_CIX →→ DIN(93.9) →→ WR_EN →→ PROG_FUL_THRESH,90 →→ C_FUL_THRESH,90 →→ RD_CIX NUCCTSBITERR →→ FUL ←→ FROG_EMPTY_THRE FUL ←→ PROG_EMPTY_THRE PROG_EMPTY_	₽× logiC©RE	FIFO Generator	xilinx.com:ip:fifo_generator:9.
WR_DARA_COUNTIES] ← → ALMOST_EMERTY → PRO_DARA_COUNTIES] → ALMOST_EMERTY → RD_DARA_COUNTIES]	Initialization Initialization Reset Pin Reset Type Synchronous Reset Asynchronous Reset Full Flags Reset Value Ivalue Vise Dout Reset Use Dout Reset Use Dout Reset Use Dout Reset Value Programmable Flags Programmable Full Type Full Threshold Assert Value Full Threshold Assert Value Full Threshold Assert Value Encyty Threshold Assert Value Empty Threshold Negate	Enable Reset S      (Hex)      Single Programmable Full Threshold Cons     1013 Range: 111021     1012 Range: 101012      No Programmable Empty Threshold     e 4 Range: 42044     ue 5 Range: 52045	tant
	Datachort	Pack Dans 5 of 7 Mays	Concrete Cancol Hole

#### Page 6:





Page 7:



The FIFO will use 2x36K BRAMs. Reducing the FIFO depth doesn't affect the BRAM implementation.

- Generate the core and move the generated files (*wh2axis\_fifo.vhd/.ngc/.xco*) to "..\sip\_wh\_in2axis\_32b\_out\vhdl\k7\Xilinx\"
- 7. Add the required files to the .LST file as described in section 2.2.3:
  - ../vhdl/xilinx/wh2axis\_fifo.xco
  - ../vhdl/xilinx/wh2axis\_fifo.vhd

../vhdl/xilinx/wh2axis\_fifo.ngc

#### 3.1.4 Update the wh\_in2axis\_32b\_out.vhd file and implement conversion logic

Declare and instantiate the wh2axis\_fifo component in *wh\_in2axis\_32b\_out.vhd*. The connections to be made are shown in Figure 5.



#### Figure 5 wh2axis\_fifo interface connections

A summary of the wormhole connections is shown in Table 5

wormhole	signal	connects to
	in_stop	prog_full
data in	in_dval	wr_en
	in_data	din
	tvalid	valid
	tdata	dout
	tlast	force '0'
data out	tuser	force 0x00000000
	tready	rd_en
	tstrb	not used
	tkeep	not used

#### Table 5 wh\_in2axis\_32b\_out data wormholes

The clk and rst wormhole signals map as shown in this table.

wormhole	signal	connects to
clk cmd_clk		wr_clk, rd_clk
	rst_in[2]	rst
rst	rst_in[10]	not used
	rst_in[313]	not used

Table 6 wh\_in2axis\_32b\_out clk/rst wormholes



#### 3.2 Creating the axis\_32b\_in2wh\_out star

Data flowing from the AXI interface to a 4DSP standard wormhole goes through this axis\_32b\_in2wh\_out star. A star with the following wormholes is to be created in this section.

Name	Wormhole
data_in	axis_32b_in
data_out	wh_out
clk	cmd_clkin
rst	rst_in

#### Table 7 axis\_32b\_in2wh\_out wormholes

The main features of the star will be:

- Convert 32-bit AXI data width to 64-bit 4DSP wormhole data width
- Change the AXI write protocol to the 4DSP standard wormhole write protocol
- Optional: the star supports clock boundary crossing

The implementation of this star is based on a Xilinx FIFO generated with the Core Generator.

#### 3.2.1 Create the axis\_32b\_in2wh\_out star in stellar IP

Repeat the steps of section 3.1.1 to create a star with pins: *clk* = cmd\_clkin type, *rst* = rstin type, *data\_in* = axis\_32b\_in type, *data\_out* = wh\_out type.

The sip\_axis\_32b\_in2wh\_out star will look like:



Figure 6 Axis\_32b\_in2wh\_out star symbol

#### 3.2.2 Create the axis\_32b\_in2wh\_out skeleton

Repeat the steps of section 3.1.2 to create the skeleton.

#### 3.2.3 Create the axis\_32b\_in2wh\_out conversion logic

Similar to the conversion logic of the wh\_in2axis\_32b\_out , a FIFO is used to implement the conversion logic. The differences between the AXIS and 4DSP data wormhole protocols require a dedicated FIFO with some small configuration differences with respect to the wh2axis\_fifo. Repeat the steps described in section 0 with the following changes:

- Page 3: 'Read Mode' = Standard FIFO, 'Write Width' = 32, 'Read Width' = 64

Copy the *axis2wh\_fifo.vhd/.ngc/.xco* files to "..\sip\_axis\_32b\_in2wh\_out\vhdl\k7\Xilinx\" and add the files to the .LST file:

../vhdl/k7/xilinx/ axis2wh\_fifo.vhd

../vhdl/ k7/xilinx/ axis2wh\_fifo.xco

../vhdl/ k7/xilinx/ axis2wh\_fifo.ngc

Declare and instantiate the axis2wh\_fifo in wh\_axis\_32b\_in2wh\_out.vhd. The connections to be made are shown in Figure 7.



#### Figure 7 axis2wh\_fifo interface connections and logic

A summary of the wormhole connections is shown in Table 8

wormhole	signal	connects to
	tvalid	and with inverse of prog_full and feed into wr_en
tdata		din
	tlast	not used
data in	tuser	not used
udla ili	tready	connect to inverse of prog_full
	tstrb	force '0'
	tkeep	force 0x0
	out_stop	inverse and feed into rd_en
data out	out_dval	valid
	out_data	dout

#### Table 8 axis\_32b\_in2wh\_out data wormhole connections

The clk and rst wormhole signals map as shown in this table.

wormhole	signal	connects to
clk	cmd_clkin	wr_clk, rd_clk
	rst_in[2]	rst
rst	rst_in[10]	not used
	rst_in[313]	not used

Table 9 axis\_32b\_in2wh\_out clk/rst wormhole connections

AN001



r1.0

## **4** Interface the command wormholes to axis

4DSP has defined a command distribution scheme based on a command wormhole that comprises of 64 bits vector and a valid signal. The 64 bits vector is used to send command packets. A command packet is made from 32 bits data, 24 bits address and 4 bits command. These fields are mapped to the 64 bits vector. Within a constellation there is always one star that will be the command master and the other stars can be command slaves. The command master has a *cmd\_out* wormhole that connects directly to multiple cmd\_in wormholes in parallel. The cmd\_out wormhole of each star is multiplexed to one *cmd\_out* wormhole before it is input to the *cmd\_in* wormhole of the command master star. Each star is assigned a range within the 24 bits address space which allows individual addressing of status and control registers within each star.

In order to retrieve status from the FFT core star and to send control data to the FFT core star we need to create a conversion star between axis\_32b\_in/out and cmd\_in/out wormholes.

#### 4.1 Control/Status Interface of FFT core

The FFT core has one status register (address 0x0) and one control register (address 0x1). The registers are based on the "Implementation Details" of the FFT IP Core generated with COREGen and explained below.

Bit nr.	31 downto 23	22 downto 9	8	7 downto 5	4 downto 0
Name	Reserved	Scale Schedule	FWD_INVn	Reserved	NFFT

Table 10: Register FFT core configuration register

Field	Description
Scale Schedule	These are the scaling bits, setting of this field depends on input power level and transform size. Refer to the XFFT core datasheet for more information
FWD_INVn	Fordward or inverse Fourier Transform. When performing FFT of ADC samples this field must be set to forward = '1'
NFFT	Transform size. This field must be configured according to the number of points that the FFT transform will be. The settings of this field are:



Field	Description	
	NFFT[4:0]	Transform size (N)
	00011	8
	00100	16
	00101	32
	00110	64
	00111	128
	01000	256
	01001	512
	01010	1024
	01011	2048
	01100	4096
	01101	8192
	01110	16384
	01111	32768
	10000	65536

#### Table 11: FFT core configuration field description

This status register implements a single status bit:

Bit nr.	31 downto 1	0
Name	Reserved	OVFLO

#### Table 12 FFT core status field

This is the overflow indicator and is set to '1' in case an overflow has occurred in a single FFT block of size NFFT. An overflow might occur when the scaling schedule is set incorrectly or the input power is too high.

The control register is accessed by writing a value on the control AXIS input port. Status is written to the status AXIS output port.

#### 4.2 Creating the cmd2axis\_32b star

The cmd2axis\_32b star must be created with the wormholes defined in Table 13.

Name	Туре
cmd_in	cmd_in
cmd_out	cmd_out
cmdclk_in	cmdclk_in
rst	rst_in
axis_32b_out	axis_32b_out
axis_32b_in	axis_32b_in

#### Table 13 cmd2axis\_32b wormholes

This star implements two registers on the StellarIP register map. A write to register at offset 0 will translate into a write on the axis\_32b\_out wormhole. The data from the command bus is mapped directly on the 32 bits axis data bus. In our design example the FFT core will only receive the lower 24 bits as it does not connect the upper 8 bits.

For the status we can choose two implementations

- We implement a FIFO between the axis\_32b\_in wormhole to store status words that are received. Each read from address offset 1 will then return a word from the FIFO. This is only useful if the status will be read constantly and if it is important to keep track the status for each frame.
- 2) We implement a counter that increments each time the overflow bit was set. We will clear the counter on each read from register offset 1.

For this tutorial the second option will be used. We understand there is little value in this application but it serves well as an example.

#### 4.2.1 Create the cmd2axis\_32b star in stellar IP

The star will be connected to the 4DSP CMD wormholes and requires a generic start and stop address. By enabling this option, StellarIP defines the start and stop address automatically as such that it will not conflict with any other star's CMD wormhole address ranges. To enable these generics, the star (required ports shown below) can be created in the similar way as previous ones, but when saving the star, one field must be edited in the "Star Properties" window:

	Juli Flopen	lies -> Gene	
🔜 Library Star Properties			×
sip_cmd2axis_xfft	Attributes Externals Generics	•	
	Wormhole		
	Add Delete		Save
Library Hie D: (Projects (Fr_lib) (	USPmain		

1. Go to "Library Star Properties -> Generic" and click "Add".

2. Select to top one and confirm with 'Ok':

×		Wormhole Listing
c .	es are displayed in this list	te that only generic wormh
	Direction	lame
	Generic	eneric
	Generic	jeneric_def
Cancel	OK	Editor
	ОК	Editor



r1.0

Then save the star.

The sip\_cmd2axis\_32b star will look like:



Figure 8 Cmd2axis\_32b star symbol

#### 4.2.2 Create the cmd2axis\_32b skeleton

Now create the star skeleton with the setting: "Generate Skeleton" (in this case the one <u>with</u> registers). The .lst/.nfo and .vhd files are generated. In addition two more VHDL files are generated: *sip\_cmd2axis\_xfft\_stellar\_cmd.vhd* and *sip\_cmd2axis\_xfft\_stellar\_regs.vhd*. In this tutorial, the *sip\_cmd2axis\_xfft\_stellar\_regs.vhd* file is not used. The *sip\_cmd2axis\_xfft\_stellar\_cmd.vhd* features an important function, it converts the 4DSP CMD wormhole protocol in a simple discrete register interface with the required address, data, request and valid signals. The in- and outputs of the command block connect to a synchronous process that implements the interface to the AXI-Streaming interface of the FFT core. The example source code of this block is provided in Appendix A.

## **5** Data formatter

The FFT core expects the input data to be 16 bits, 2s complement complex data mapped to 32 bits data bus. In case only real data is input the imaginary part of the data should be all zeros. The FMC104 star maps the data from each ADC on a 64 bits *wh\_out* wormhole. The 14 bits data, is 2's complement, sign extended to 16 bits and then de-multiplexed to have 4 samples on 64 bits before it is transmitted to other stars. We have two options:

1) We create a data formatter that takes a *wh\_in* and and for each 64 bits sample it will output two 64 bits samples. The data on the output will be as follows

1st clock cycle		2nd cloc	k cycle
data_out[150]	data_in[150]	data_out[150]	data_in[4732]
data_out[3116]	0x0000	data_out[3116]	0x0000
data_out[4732]	data_in[3116]	data_out[4732]	data_in[6348]
data_out[6348]	0x0000	data_out[6348]	0x0000

To cope with the data rate difference it is required to use a FIFO inside the star.

2) The other option is to create a star with an *axis\_32b\_in* and an *axis\_32b\_out* wormhole. For each 32 bits input sample it will output two 32 bit output samples. The data on the output will be as follows

1st clock	cycle	2nd cloc	k cycle
data_out[150]	data_in[150]	data_out[150]	data_in[3116]
data_out[3116]	0x0000	data_out[3116]	0x0000

Because of the nature of the AXI stream protocol this could be implemented without FIFO.

Within this tutorial we have chosen to implement the second option. The star will have the interfaces shown in the table below.

Name	Wormhole
clk	cmdclk_in
rst	rst_in
axis_32b_out	axis_32b_out
axis_32b_in	axis_32b_in

Table 14 axis\_32b\_real2complex star wormholes



r1.0

#### 5.1.1 Create the axis\_32b\_real2complex star in stellar IP

Create the data formatter star with symbol:



Figure 9 axis\_32b\_real2complex star symbol

#### 5.1.2 Create the sip\_axis\_32b\_real2complex skeleton

After saving the star, create the skeleton without registers:

sip_axis_32b_real2complex sip_axis_32b_real2complex		
	Edit Star Symbol	data_in data_out
Errors All	Star Properties	
te.	New Star	
y. map.	Clone Star	
the entity.	Delete Star	
into the file. (1)	Generate HDL Skeleton	
into the file. (3)	Generate HDL Skeleton (No registers)	

Figure 10 Generation axis\_32b\_real2complex skeleton

#### 5.1.3 Create the sip\_axis\_32b\_real2complex conversion logic

An example without the use of a FIFO is provided in Appendix B.



## **6** Generate a Constellation

This section describes the steps that one could follow to generate a new constellation from scratch. It is also possible to clone an existing constellation; this is not part of this tutorial.

- 1. Open the schematic tool
- 2. Create new design. Please refer the following settings. Make sure that KC705 is selected and ISE supported is checked.

🔜 New Design Det	tails 📃 🔀
Document Number	419
Name	kc705_fmc104_xfft
Туре	Reference Firmware for Xilinx FFT Tutorial
Revision	1.0
Description	Training Material - StellarIP and AXI4-Streaming interface connection with FFT core
Notes	
Last Change Date	Jan. 09, 2014
Author	iklink
Organization	4DSP
Output Path	D:\Projects\FW
Final Path	D:\Projects\FW\419_kc705_fmc104_xfft\419_kc705_fmc104_xfft.dsn
Board Type	KC705 -
FPGA Name	A 🔹
FPGA Type	XC7K325T-2FFG900
Memory Start	0x000000
Memory Stop	0x0001FFF
Firmware Flow	Vivado Supported ISE Supported

3. This windows is recalled if you go to "File->Design Details", the design settings can be reedited.

4. Go to Stellar IP in the menu and open settings. Make sure that Xilinx ISE is enabled and ISE binary path is correct.

Enabled				
New ISE binary path:			•	
Active ISE binary path:	C:\Xilinx\14.7\ISE_DS\I	SE\bin\nt64		
Xilinx Vivado				
Enabled				
New Vivado binary path:				]
Active Vivado binary path:	C:\Xilinx\Vivado\2013.3	bin		
Launch:	✓ Launch Vivado GUI af	ter generation		
StellarIP Log path: Library source files:	C:\Users\4DSP-61\Doc	uments∖og.txt ource files (Xilinx uses absolute o	ath)	
Simulation:	Add 'Run Simulation' t	no the tril script		
Load local library:	No	•		
Load local library: Verbosity	No	<ul> <li>Recent Files History</li> </ul>		
Load local library: Verbosity Global system verbosity:	No 3 - Normal	<ul> <li>Recent Files History</li> <li>Max nr. of entries:</li> </ul>	6	
Load local library: Verbosity Global system verbosity: File Extensions	No 3 - Normal	<ul> <li>Recent Files History</li> <li>Max nr. of entries:</li> </ul>	6 🔹	

5. From the star navigator on the left, stars can be added to the design. Add stars required for the design and connect the pins. Instead of drawing the data path, labels can be used.

tiedto0 label is a reserved label for StellarIP to connect an input to ground.

Right clicking on the star symbol shows the possible actions in the context menu. The example of the constellation is shown in Appendix D.

6. Open the completed tutorial design.

7. Open Star Order under Edit menu. StellarIP assigns the register addresses by the star order if the star requires a register allocation. To maintain compatibility with the existing '159\_kc705\_fmc104' constellation, the star ordering shown below is recommended. Using this order allows one to easily reuse parts of the software and simulation scripts.

	sition. Pre	ess OK to continue.		
Name	ID	LST File	*	Move Up
sip_cid	0	Αυτογαυτο		Move Down
sip_mac_engine	0	sip_mac_engine\sip_files\sip_mac_engine_k7.lst		
sip_router_s5d1	0	Αυτογουτο		
sip_fifo64k	0	Αυτογοτυα	-	
sip_fmc104	0	sip_fmc104\sip_files\sip_fmc104_k7.lst	-	
sip_i2c_master	0	Αυτογοτυα		
sip_cmd12_mux	0	Αυτογοτυα		
sip_fmc_ct_gen	0	<pre>sip_fmc_ct_gen\sip_files\sip_fmc_ct_gen_2gbp</pre>		
sip_cmd2axis_xfft	0	οτυς/οτυς		
sip_axis_32b_in2wh_out	0	Αυτογαυτο		
sip_wh_in2axis_32b_out	0	Αυτο\αυτο		
sip_axis_32b_real2complex	0	Αυτο\Αυτο		
sip xfft wrapper	0	Αυτολαυτο	-	OK

8. Some stars can be used on multiple boards, for these stars it is required to point StellarIP to the correct .lst file. You can change the LST path and LST name of stars by double clicking on the star. This will show a window where you can edit these properties For example:

www.4dsp.com

sip_tmc104 I	ool Optio	ons	
Attribute	Show	Value	
Name	Yes	sip_fmc104	
Package	No	Stellar IP Component	
ID	No	0	
LST Path	No	sip_fmc104\sip_files	
LST Name	No	sip_fmc104_k7.lst	
Nbr Register	s No	131072	
Add	De	elete	
sip mac enq	ine Tool C	Options	
			_
Attribute	Show	Value	
Name	Yes	sip_mac_engine	
Dackage			
Fackage	No	Stellar IP Component	
ID	No No	Stellar IP Component 0	
ID LST Path	No No No	Stellar IP Component 0 sip_mac_engine\sip_files	
ID LST Path LST Name	No No No	Stellar IP Component 0 sip_mac_engine_sip_files sip_mac_engine_k7.lst	
ID LST Path LST Name Nbr Register	No No No No s No	Stellar IP Component 0 sip_mac_engine\sip_files sip_mac_engine_k7.lst 0	
ID LST Path LST Name Nbr Register	No No No S No	Stellar IP Component 0 sip_mac_engine\sip_files sip_mac_engine_k7.lst 0	
ID LST Path LST Name Nbr Register	No No No s No	Stellar IP Component 0 sip_mac_engine\sip_files sip_mac_engine_k7.lst 0 elete	

9. Save the changes

AN001



This section briefly describes the steps to generate the ISE project and compile the design.

- In StellarIP, open the completed tutorial design. StellarIP automatically gathers the information in the design file and can generate the ISE Project with all required files. Go to StellarIP in the menu and click "Generate" (or use ctrl+g).
- The console window reports the StellarIP process. If there's an error, it reports possible issues/errors. By changing the Global System Verbosity (in "StellarIP -> Settings", or F11), more detailed reports can be generated. Some of the errors can be found in the trouble shooting section at the end of this document.





3. Go to the output folder under the constellation folder:

"../419\_kc705\_fmc104\_xfft/output/kc705\_fmc104\_xfft/". The ISE project folder is generated there. Open the "kc705\_fmc104\_xfft.xise". Click the top vhdl module and run "Generate Programming File".



4. After the complete compilation, verify that there are no errors or timing violations.

## 8 Software Application

To go through this section, it is required to have Visual Studio 2012 installed. If not executed yet, follow the steps that are required to use this software, described in the 4FM Getting Started Guide.

- In Visual Studio 2012, open the project "..\4dsp\FMC Board Support Package\Refs\Software\FMC10x\Fmc10xAPP.vcxproj"
- 2. Open "fmc10xids.h" All constellations that are supported by this application are defined here. The Constellation for this tutorial was set to 419 in Section 0. To support this ID add the following line.

#define CONSTELLATION\_ID\_KC705\_FMC104\_XFFT 0x1A3

3. Open "Libs/FMC10x/Impls/fmc10x.cpp" and go to the switch case that configures the IO delay controllers. The configuration of the delay controllers should be the same as per *CONSTELLATION\_ID\_KC705\_FMC104*. Add

case CONSTELLATION\_ID\_KC705\_FMC104\_XFFT: To the CONSTELLATION\_ID\_KC705\_FMC104 case.

- Open "Libs/FMC10x/Impls/fmc10x\_clocktree.cpp" and add case CONSTELLATION\_ID\_KC705\_FMC104\_XFFT: To the CONSTELLATION\_ID\_KC705\_FMC104 case of the switch.
- Open "main.cpp" and add a CMD2AXI\_XFFT\_ID. Assign to this the value of the star ID. In order to find the star ID, go to "sip\_cmd2axis\_xfft/sip\_files" and open "sip\_cmd2axis\_xfft.nfo" file.

#define CMD2AXI\_XFFT\_ID 0xE2

 Within "main.cpp", find the variable set up for given constellation IDs. FMC10xAPP supports 4DSP FMC products such that FMC103, FMC104, FMC107 and FMC108. Add the following case.

```
case CONSTELLATION_ID_KC705_FMC104_XFFT :
deviceFW = "FMC104 Xilinx FFT on KC705\n";
modeFMC104 = 1;
FMCConstID = FMC104_ID;
FMCnbrch = 4;
routerID = ROUTER_S5D1_ID;
modeKC705 = 1;
break;
```

 cid\_getstaroffset() reads the star offset which represent the star register address. The star offset read functions are followed by the constellation ID case statements. Add the following function call. (define AddrSipCmd2AxiXfft as uint32\_t)

```
if(cid_getstaroffset(CMD2AXI_XFFT_ID, &AddrSipCmd2AxiXfft, &size)!=SIP_CID_ERR_OK) {
    printf("Could not obtain address for star type %d, exiting\n",
    CMD2AXI_XFFT_ID);
    sipif_free();
```



r1.0

return -7;

}

8. The FFT core processes real samples and outputs complex samples. In "main.cpp", the BurstSize defines the number of samples captured by the ADC interface and transmitted to the FFT core. The *pInData* memory is allocated to store the data samples. This memory is allocated in bytes. For this constellation it is necessary to allocate 4 times the BurstSize. Change the memory allocation to:

```
unsigned char *pInData = (unsigned char *)_aligned_malloc(4*BurstSize, 4096);
```

9. The FFT core must be configured before starting a data capture. Because the XFFT control interface is a dedicated star within the design, it recommended to create a new module for this star. Within "..\Fmc10xAPP\Libs\", create to following folders and files:

"\XILINX FFT\Incs\xilinx fft.h"

"\XILINX\_FFT\Impls\xilinx\_fft.cpp"

In Visual Studio->Project->FMC10xAPP Properties, add the new include directory ("\XILINX\_FFT\Incs\xilinx\_fft.h") with a relative path. The picture below shows the dialog.

Fmc10xAPP Property Pages		8 ×
Configuration: Active(Debug)	Platform: Active(Win32)	Configuration Manager
> Common Properties	Additional Include Directories	Libs\SIPIF\Incs;Libs\FMC10x\Incs;Libs\CT_GEN\Incs;Libs\SXD) 🔻
Configuration Properties	Additional #using Directories	
General	Debug Information Format	Program Database for Edit And Continue (/ZI)
Debugging	Common Language RunTime Support	
VC++ Directories	Consume Windows Runtime Extension	
⊿ C/C++	Suppress Startup Banner	Yes (/nologo)
General	Warning Level	Level3 (/W3)
Optimization	Treat Warnings As Errors	No (/WX-)
Preprocessor	SDL checks	
Code Generation	Multi-processor Compilation	
Language		
Precomplied Headers		
Browse Information		
Advanced		
All Ontions		
Command Line		
⊳ Linker		
Manifest Tool		
XML Document Generator		
Browse Information		
Build Events		
Custom Build Step		
Code Analysis	Additional Include Directories	
	Specifies one or more directories to add to the	e include path; separate with semi-colons if more than one.
4 III >	(/I[path])	
		OK Cancel Apply

As defined in the sip cmd2axi xfft.vhd, the address of the configuration and status are

- Configuration: AddrSipCmd2Axi+0

AddrSipCmd2Axi+1 - Status:

Implement the source codes provided in Appendix C to be able to configure the Xilinx FFT core and retrieve status.

AN001 Tutorial: Axis to StellarIP Interface

10. In "main.cpp", include the header file created in previous step #include "xilinx\_fft.h"

Further down the file, before the data capture for the four channels is initiated, configure the FFT core by adding the following source code (calling the function in the xilinx fft module) :

```
// Configure Xilinx FFT Core
if (cid getconstellationid()==CONSTELLATION ID KC705 FMC104 XFFT) {
     int scaling = 0x7777;
     if(xfft configure(AddrSipCmd2AxiXfft, true, scaling,
     BurstSize)!=SIP XFFT ERR OK){
          printf("Could not configure Xilinx FFT Core\n");
          printf("Xilinx FFT is set to illegal amount of points:
          %d\n",BurstSize);
          printf("BurstSize must be set with a power of 2, range 64 to
          8192\n");
          sipif free();
     }else{
          printf("-----\n");
          printf("Xilinx FFT set to %d points\n",BurstSize);
          printf("-----\n");
     }
}
```

11. In "main.cpp", go to the part where data is read from the pipe, this is done within the for loop:

```
// loop here as many times we have channels to grab samples from
for(int32_t i = 0; i < FMCnbrch; i++) {</pre>
```

Double the amount of bytes have to be read from the pipe for the FFT constellation. Change:

```
printf("Retrieve %d samples from ADC%d\n", BurstSize,i);
if(sipif_readdata (pInData, 2*BurstSize)!=SIPIF_ERR_OK) {
    printf("Could not communicate with device %d\n", devIdx);
    sipif_free();
    _aligned_free(pInData);
    return -24;
```

To:

```
if (cid_getconstellationid()!=CONSTELLATION_ID_KC705_FMC104_XFFT) {
    printf("Retrieve %d samples from ADC%d\n", BurstSize,i);
    // Real samples only
    if(sipif_readdata (pInData, 2*BurstSize)!=SIPIF_ERR_OK) {
        printf("Could not communicate with device %d\n", devIdx);
        sipif_free();
        _aligned_free(pInData);
        return -24;
    }
} else {
    printf("Retrieve %d point FFT from ADC%d\n", BurstSize,i);
    // Complex samples from FFT core
    if(sipif_readdata (pInData, 4*BurstSize)!=SIPIF_ERR_OK) {
        printf("Could not communicate with device %d\n", devIdx);
    }
}
```

}

}



```
sipif_free();
_aligned_free(pInData);
return -24;
```

12. Received *BurstSize* number of samples contain the complex values. Use the following codes to separate the real and imaginary values and to compute the magnitude of the complex values. Then, write each buffer to file. (this is part of the same for loop as per previous step)

```
if (cid getconstellationid()!=CONSTELLATION ID KC705 FMC104 XFFT) {
       // compute the file names
       char filenamebin[1024];
       char filenameascii[1024];
       sprintf(filenameascii, "adc%d.txt", i);
       sprintf(filenamebin, "adc%d.bin", i);
       // clean the files previously saved
       DeleteFile(filenamebin);
       DeleteFile(filenameascii);
       // write to file
       Save16BitArrayToFile(pInData, BurstSize, filenameascii, ASCII);
       Save16BitArrayToFile(pInData, BurstSize, filenamebin, BINARY);
} else {
       short *pInData_re = new short[BurstSize];
       short *pInData_im = new short[BurstSize];
       short *pInData_mag = new short[BurstSize];
       short *data = (short *)pInData;
// Map the complex and real samples into the correct files and calc magnitude
       int count = 0;
       for(int j=0; j<(BurstSize*2); j=j+2)</pre>
       {
              short temp[2];
              double mag, re, im;
              temp[0] = data[j];
              temp[1] = data[j+1];
              re = (double)temp[0];
              im = (double)temp[1];
              mag = (re*re) + (im*im);
              mag = sqrt(mag);
              pInData_re[count] = temp[0];
              pInData_im[count] = temp[1];
              pInData_mag[count++] = (short)mag;
       }
       // compute the file names for complex samples
       char filenameascii_re[1024];
       char filenameascii_im[1024];
       char filenameascii_mag[1024];
       char filenamebin_re[1024];
       char filenamebin_im[1024];
       char filenamebin_mag[1024];
       sprintf(filenameascii_re, "adc_re%d.txt", i);
sprintf(filenameascii_im, "adc_im%d.txt", i);
```

AN001 Tutorial: Axis to StellarIP Interface

sprintf(filenameascii\_mag, "adc\_mag%d.txt", i); sprintf(filenamebin\_re, "adc\_re%d.bin", i);
sprintf(filenamebin\_im, "adc\_im%d.bin", i); sprintf(filenamebin\_mag, "adc\_mag%d.bin", i); // clean the files previously saved DeleteFile(filenameascii re); DeleteFile(filenameascii im); DeleteFile(filenameascii mag); DeleteFile(filenamebin\_re); DeleteFile(filenamebin\_im); DeleteFile(filenamebin\_mag); // Save output to files Save16BitArrayToFile(pInData\_re, BurstSize, filenameascii\_re, ASCII); Save16BitArrayToFile(pInData\_im, BurstSize, filenameascii\_im, ASCII); Save16BitArrayToFile(pInData\_mag, BurstSize, filenameascii\_mag, ASCII); Save16BitArrayToFile(pInData\_re, BurstSize, filenamebin\_re, BINARY); Save16BitArrayToFile(pInData\_im, BurstSize, filenamebin\_im, BINARY); Save16BitArrayToFile(pInData\_mag, BurstSize, filenamebin\_mag, BINARY);

13. Within the same for loop, after the data is read from the pipe, read the status of the XFFT overflow flag to see whether the input power was too high or the scaling was set incorrectly. Add the following source code (this source code calls the *xfft\_getstatus* function inside the xilinx\_fft module):

```
// Retrieve overflow status of XFFT core
if (cid_getconstellationid()==CONSTELLATION_ID_KC705_FMC104_XFFT) {
    uint32_t ovflow_cnt = 0;
    if(xfft_getstatus(AddrSipCmd2AxiXfft,
        &ovflow_cnt)!=SIP_XFFT_ERR_OK){
        printf("Could not retrieve overflow status from Xilinx FFT
        Core\n ");
        sipif_free();
    }else{
        printf("FFT Scaling overflow count of ADC%d:%d \n\n", i,
        ovflow_cnt);
    }
}
```

- 14. Text files for real, imaginary and magnitude values can be found in the software folder. You can further implement the software to generate the FFT plot or use other application.
- 15. Compile the project. The executable is generated in "..\4dsp\FMC Board Support Package\Refs\Software\FMC10x\Debug". Follow the steps described in the 4FM Getting Started Guide to test the application.

AN001

}





## **9** StellarIP Troubleshooting

A list of possible errors which may occur due to errors in the design is provided in the table below. Sometimes more information can be gathered from the log window. The verbosity of the log can be changed in "StellarIP -> Settings" (shortkey F11). The list also provides suggestions for solutions.

Error message	Solution
Unable to open the following sip_core info file: D:\\\sip_mac_engine\sip_files\sip_mac_engine _k7.nfo	The path or name of the .LST file is incorrect.
SDF Error -18: No such port name	One of the stars has a Pin with an incorrect configuration: A Pin with Electrical interface set to <u>Output</u> is not allowed to have ' <u>in</u> ' in its Pin name, for example: data_out_clk being an input is not allowed This also applies for the <u>Input</u> with ' <u>out</u> ' in its name. Also check if the Electrical interface in Pin Tool Options is set correctly (input/output) <u>Name data_dckin Select</u> <u>yb Down Left Bight Length:</u> <u>30</u> <u>shape Normal Blectrical input</u>



Error message	Solution
The star 'xxxxx.0' has unconnected ports, please check your .sdf file The unconnected wormhole might be named 'xxxxxxx'	Check in the constellation schematic if all signals are properly connected. This wormhole input is connected to a star with a wormhole output, check if Electrical interface of the Pin of this star is correctly set to Output Pin Tool Options Vormhole wh_out Select Up Down @Left Right Length: 30 Shape Normal Electrical Output
HDLCompiler:806 - "D://vhd" Line XX: Syntax error near ":"	There is a syntax error in the VHDL programming file
Input Error: Missing value for required design parameter 'Target FPGA'	Make sure that the correct target FPGA is selected in the Design Details dialog. When migrating an old .sdf to the new .dsn format some FPGA related details can get corrupted:



## A. Sip\_cmd2axis\_xfft example source code

component instantiations
i_cmd2axis_xfft_stellar_cmd: cmd2axis_xfft_stellar_cmd
generic·map
(
<pre>&gt;&gt;start_addr</pre>
···stop_addr············=>private_stop_addr_gen
)
port-map
(
<pre>resetreset</pre>
···command if
<pre>clk_emde&gt;emdelk_in_emdelk,</pre>
<pre>out_emd</pre>
···out_cmd_val·················=>cmd_out_cmdout_val,
···in_cmd····················=>cmd_in_cmdin,
···in_cmd_val····································
···register ·interface
<pre>clk_reg=&gt;axi_clk_cmdclk,</pre>
<pre>out_reg</pre>
···out_reg_val···············->out_reg_val,
···out_reg_addr···········=>out_reg_addr,
···in reg···········=>in reg,
···in reg val··············=>in reg val,
···in reg reg.
···in reg addr=>in reg addr.
···mbx out reg····································
···mbx out val····································
···mbx in reg····································
···mbx in val····································
Static-signal-assignments
axis 32b in tkeep ······<=·(others -=>·'0'); ·
axis 32b in tready
axis 32b in tstrb
axis 32b out tlast ·····<=·'0';
axis 32b out tdata
antduty



```
--synchronous processes
process(rst_rstin, axi_clk_cmdclk)
..variable overflow_count_val :: std_logic_vector(31 downto 0);
· · begin
····if·(rst_rstin·=·'1') ·then
....axis_32b_out_tdata_reg <= (others -> '0');
....axis_32b_out_tvalid....<=.'0';</pre>
·····in_reg······<=·(others·=>·'0');··
....overflow_count_val....:=.(others.=>.'0');
....elsif (rising_edge(axi_clk_cmdclk)) then
....if (out_reg_val = '1') and (conv_integer(out_reg_addr) = 0) then
....axis_32b_out_tdata_reg <= out_reg;</pre>
....axis_32b_out_tvalid....<=.'1';</pre>
·····elsif (axis_32b_out_tready = '1') then
....axis_32b_out_tvalid....<=.'0';</pre>
·····end·if;
....if (in_reg_req -= '1') then
....in_reg_val <= .'1';</pre>
....if (conv_integer(in_reg_addr) = 0) then
.....in reg <= axis 32b out tdata reg;
······elsif (conv_integer(in_reg_addr) ·= ·1) ·then
.....in_reg <= overflow_count_val;</pre>
·····end·if;
....overflow_count_val := (others -> '0');
····else
.....in_reg_val.<=.'0';</pre>
·····end·if;
·····--·Overflow·counter
....if (axis_32b_in_tvalid = '1') and (axis_32b_in_tdata(0) = '1') then
·····if · (overflow_count_val ·< ·x"FFFFFFFF") · then
....overflow_count_val.:=.overflow_count_val.+.'1';
·····end·if;
·····end·if;
····end·if;
end process;
```



## B. Sip\_axis\_32b\_real2complex example source code

```
--Sync process
data_map_proc : process(rst_rstin,clk_clkin)
 begin
   if (rst_rstin = '1') then
     data_reg <=(others=>'0');
     dout_sel <= '0';</pre>
     dout_val <= '0';</pre>
    elsif (clk_clkin'event and clk_clkin = '1') then
      --we should sample the data as soon as we ack to the input we received it correctly
     -- we should make sure never to ack the data unless we are sure we do not need it anymore
     if(din_trdy= '1' and data_in_tvalid='1') then
       data_reg <= data_in_tdata;</pre>
     end if;
     --as soon as we sample valid data we tell the output the data is valid
      --as soon as we sent out our last data we tell its not valid anymore
     if (din_trdy= '1' and data_in_tvalid='1') then
       dout_val <= '1';</pre>
     elsif(dout_val= '1' and data_out_tready='1' and dout_sel='1' ) then
       dout_val <= '0';</pre>
     end if;
     --each valid output word is toggeling the output mux
     if(dout_val= '1' and data_out_tready='1') then
       dout_sel <= not dout_sel;</pre>
     end if;
    end if;
  end process;
din_trdy_proc : process(dout_val,dout_sel, data_out_tready)
begin
   --if we have no data valid on the output pending we can ack data
   --or if we are about to ack our last word on the output
  if ((dout val='0')or (dout val='1' and dout sel='1'and data out tready ='1')) then
    din trdy <= '1';</pre>
   else
    din_trdy <= '0';</pre>
   end if;
end process;
```



Signal assignmen	ts
data_out_tlast	<= '0';
data_out_tuser	<= (others => '0');
data_out_tvalid	<= dout_val;
data_in_tkeep	<= (others => '0');
data_in_tready	<= din_trdy;
data_in_tstrb	<= '0';

--output mux

data\_out\_tdata <= x"0000" & data\_reg(15 downto 0) when dout\_sel = '0' else x"0000" & data\_reg(31 downto 16);</pre>



## C. Software module: xilinx\_fft

#### Xilinx\_fft.h

```
///@file xilinx fft.h
#ifndef _SIPXFFT_H_
#define _SIPXFFT_H_
#include <stdint.h>
/* error codes */
#define SIP_XFFT_ERR_OK
                                                     Ø
                 /*!< No error encountered during execution. */</pre>
#define SIP_XFFT_ERR_LOW_LEVEL_IO
                                         -1
      /*!< Unable to communicate with the hardware/firmware over Ethernet. */
#define SIP_XFFT_ILLEGAL_CONFIGURATION
                                         -2
      /*!< Illegal configuration. The BurstSize is not supported by the XFFT core. */
// C++ "helper"
#ifdef __cplusplus
extern "C" {
#endif
int32_t xfft_configure(unsigned int bar, bool forwardFFT, int scaling, int BurstSize);
int32_t xfft_getstatus(unsigned int bar, uint32_t *nbrscalingovf);
// C++ "helper"
#ifdef __cplusplus
}
#endif
```

#endif



#### Xilinx\_fft.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include "xilinx fft.h"
#include "sipif.h"
int32_t xfft_configure(unsigned int bar, bool forwardFFT, int scaling, int BurstSize)
{
       uint16 t NFFT = 0;
       switch(BurstSize) {
               case 64 :
                      NFFT = 0x06;
                      break;
               case 128 :
                      NFFT = 0 \times 07;
                      break;
               case 256 :
                      NFFT = 0x08;
                      break;
               case 512 :
                      NFFT = 0 \times 09;
                      break;
               case 1024 :
                      NFFT = 0 \times 0A;
                      break;
               case 2048 :
                      NFFT = 0 \times 0B;
                      break;
               case 4096 :
                      NFFT = 0 \times 0C;
                      break;
               case 8192 :
                      NFFT = 0 \times 0D;
                      break;
               default :
                      return SIP_XFFT_ILLEGAL_CONFIGURATION;
       }
       // Default the scaling is set to max (0x7776) to avoid overflows
       // Bit 0x100 is the Forward FFT Enable setting
       uint32_t xfft_value =(scaling<<9) + (forwardFFT<<8) + NFFT;</pre>
       sipif_writesipreg(bar+0x00, xfft_value);
       return SIP_XFFT_ERR_OK;
}
int32_t xfft_getstatus(unsigned int bar, uint32_t *nbrscalingovf)
{
       if(sipif_readsipreg(bar+0x1, nbrscalingovf)!=SIPIF_ERR_OK) {
               return SIP_XFFT_ERR_LOW_LEVEL_IO;
       }else{
              return SIP_XFFT_ERR_OK;
       }
}
```



## **D.** Example constellation



page 50 of 50