

SMIMS_VeriLink User’s Manual

Index

SMIMS_VeriLink	1
VeriLink in Simulink	1
Introduction	1
Installation	2
SMIMS VeriLink Blockset Description	7
Build your design to Simulink.....	14
Getting Start.....	21
FPGA Pin Map On Simulink	31
VeriLink In Matlab	118
Introduction	118
Data Type Conversion Function	119
VeriEnterprise Xilinx NV5 USB.....	121
VeriEnterprise Xilinx SP6 USB.....	133
SoC-S150 USB.....	145
SoCV330 USB.....	157
Macube-VEXT2 Xilinx Spartan-6 USB	169
Macube-VEXV7 Xilinx Virtex-7 USB.....	181
VeriEnterprise-VEXA7 Xilinx Artix-7 USB.....	187
VeriLite Altera Cyclone IV USB	193
VeriLite Xilinx Spartan-6 USB.....	205
Contact Information.....	217

VeriLink in Simulink

Introduction

VeriLink is a high-level tool connecting Matlab/Simulink with SMIMS FPGA board that performs reliable verification of your distinguished design. The VeriLink for Matlab brings the FPGA hardware platform to the heart of the acclaimed MATLAB technical environment.

Building upon Simulink from MathWork, along with the VeriLink from SMIMS, the overall solutions allow users to quickly perform a hardware and software co-simulation for their applications.

VeriLink shortens the design cycles by helping you to create the hardware representation of your design in the VeriLink algorithm-friendly development environment. With ready HDL design, in clicks, users can automatically feed binary to SMIMS FPGA and review the result in Simulink environment. By taking the advantages of the existing MATLAB functions and Simulink blocks to link with FPGA prototype, users can accomplish system-level design easily and quickly.

Key Features

- ✓ Link the MathWorks MATLAB/Simulink with SMIMS FPGA Platforms
- ✓ HW/SW co-simulation. Accelerate system-level co-simulation with Simulink and create an “FPGA-in-the-loop” simulation target.
- ✓ Export the HDL code to the Simulink block
- ✓ Export the FPGA download file to the Simulink block.
- ✓ Automatically produce an FPGA configuration bitstream for your HDL design
- ✓ Support all SMIMS FPGA Platforms of Xilinx/Altera

Installation

Hardware Prerequisites

Please connect SMIMS FPGA board to PC by USB. It may need some additional hardware as below.

Board	Tool
VeriEnterprise for Altera	Download cable(JTAG/AS)

Software Prerequisites

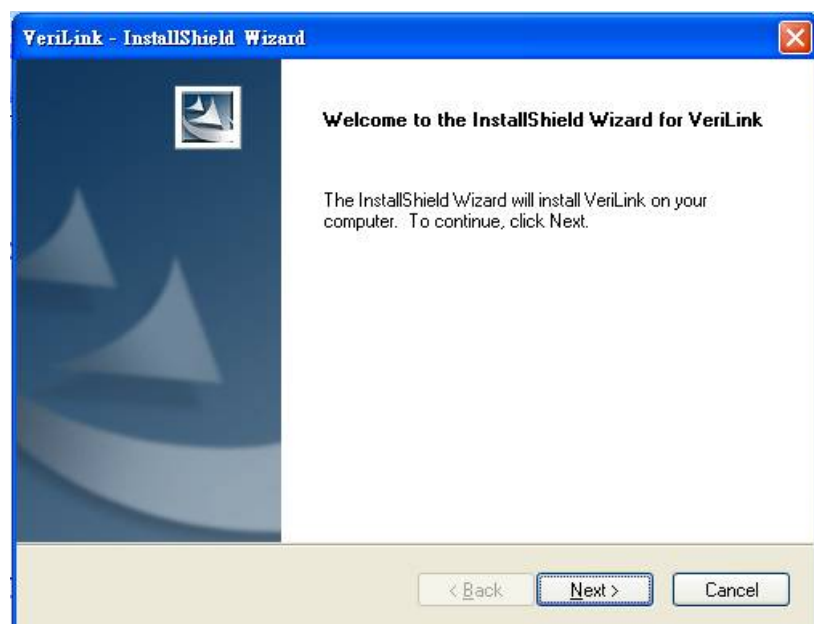
The following software must be installed before installing the VeriLink.

Board	Tool	Version
All	Matlab/Simulink	2009a or later
VeriLite for Altera	Quartus II	10.0 or later
VeriEnterprise for Altera	Quartus II	5.5 or later
VeriLite for Xilinx	Xilinx ISE	8.1i or later

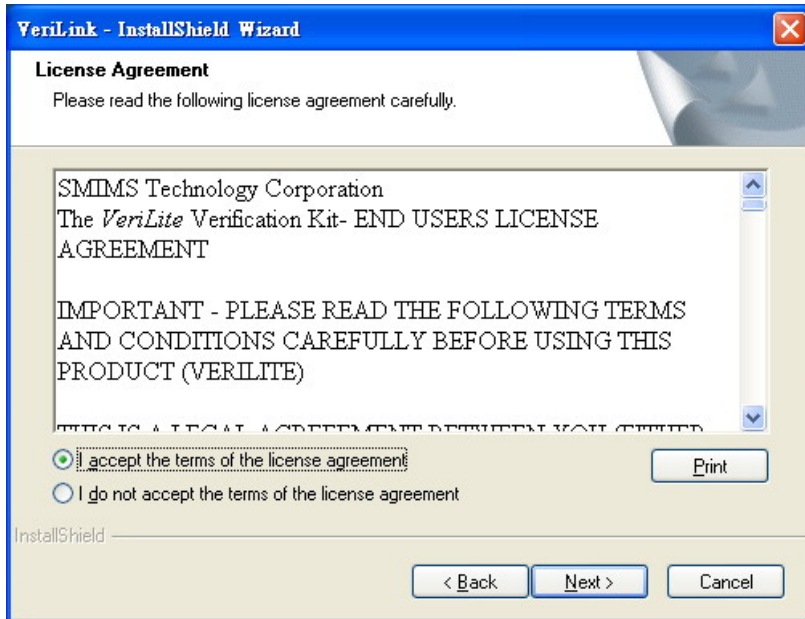
VeriEnterprise for Xilinx	Xilinx ISE	8.1i or later
VeriEnterprise for Xilinx Virtex-5	Xilinx ISE	8.2i or later
VeriEnterprise for Xilinx Spartan-6	Xilinx ISE	12.1 or later
VeriEnterprise for Xilinx Virtex-7	Xilinx ISE	14.1 or later
VeriEnterprise for Xilinx Artix-7	Xilinx ISE	14.1 or later

Installing VeriLink

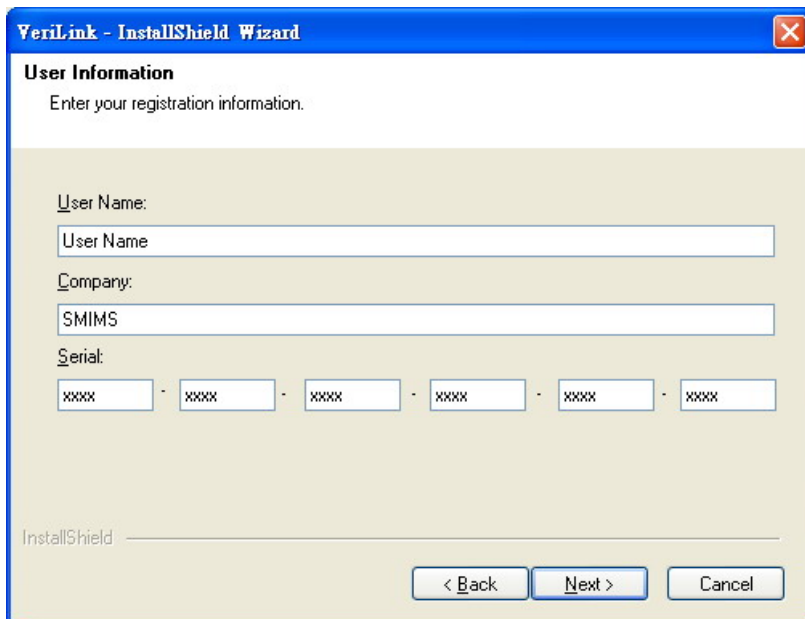
Insert SMIMS CD into the CD-ROM drive connected to the system. Double-click the ***verilink_setup.exe*** in VeriLink directory of SMIMS CD. The Welcome to the VeriLink Installer dialog box will appear.



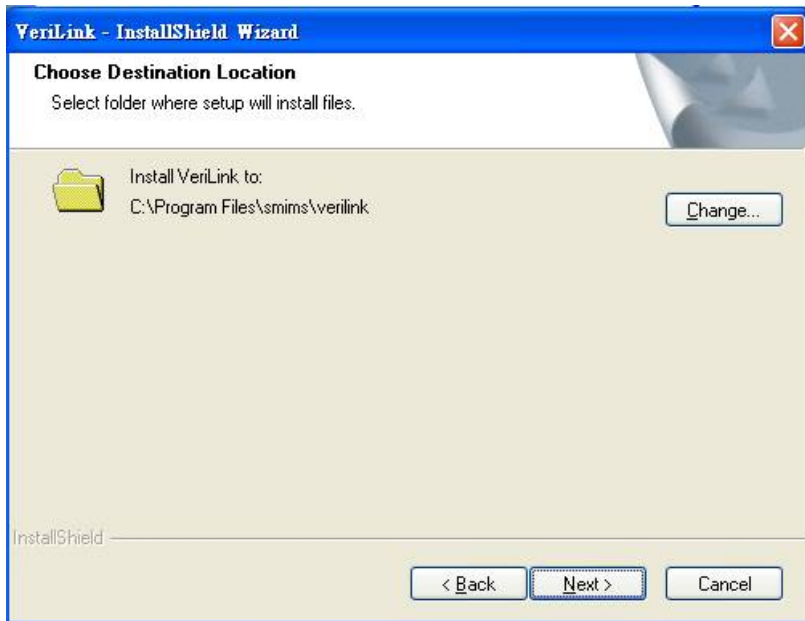
Review the software licensing agreement and Then select the “I accept the terms of the license agreement” if you agree the terms. Then click NEXT.



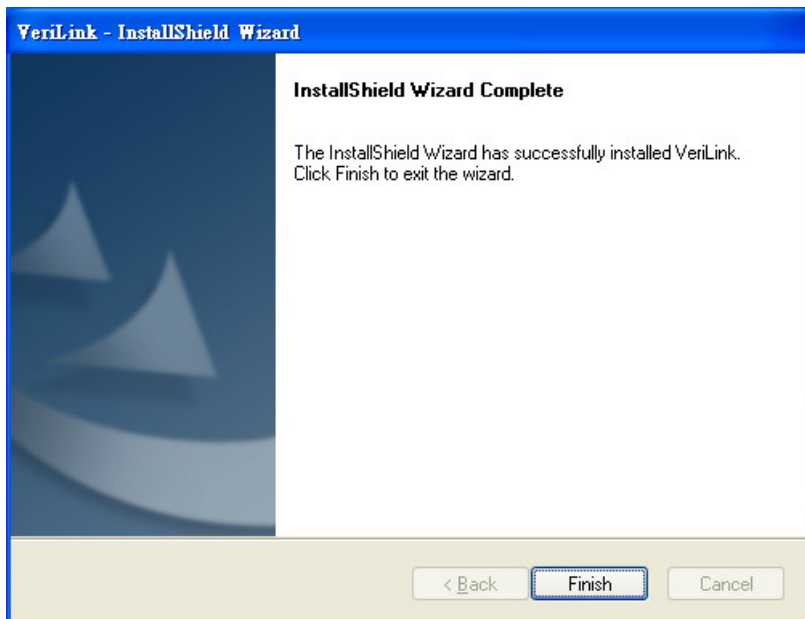
Enter user name, company name, and serial number in the warranty card and click Next.



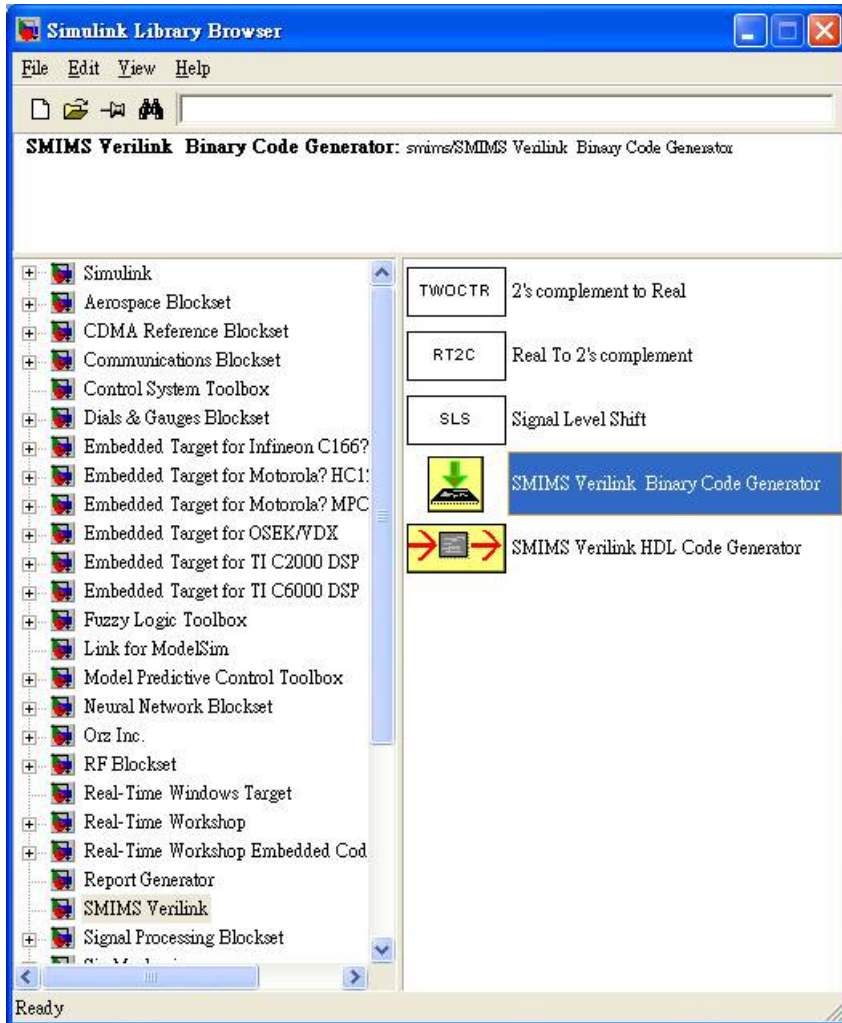
Select folder where setup will install files.



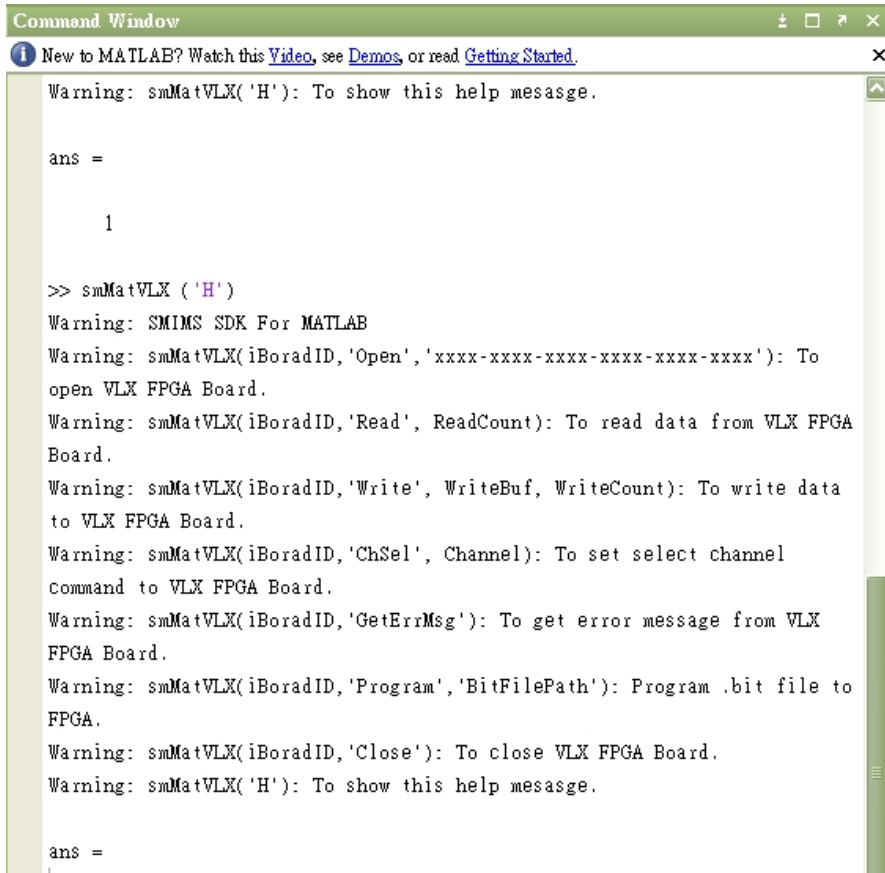
When VeriLink Installer finishes, it displays the Setup Complete dialog box.



And you will see "SMIMS Verilink" toolbox created in "Simulink Library Browser".



Or you can use VeriLink Function (smMatVLX('H') for VeriLite Xilinx,) under the Matlab command window.



```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

Warning: smMatVLX('H'): To show this help mesasge.

ans =

     1

>> smMatVLX('H')
Warning: SMIMS SDK For MATLAB
Warning: smMatVLX(iBoradID,'Open','xxxx-xxxx-xxxx-xxxx-xxxx-xxxx'): To
open VLX FPGA Board.
Warning: smMatVLX(iBoradID,'Read', ReadCount): To read data from VLX FPGA
Board.
Warning: smMatVLX(iBoradID,'Write', WriteBuf, WriteCount): To write data
to VLX FPGA Board.
Warning: smMatVLX(iBoradID,'ChSel', Channel): To set select channel
command to VLX FPGA Board.
Warning: smMatVLX(iBoradID,'GetErrMsg'): To get error message from VLX
FPGA Board.
Warning: smMatVLX(iBoradID,'Program','BitFilePath'): Program .bit file to
FPGA.
Warning: smMatVLX(iBoradID,'Close'): To close VLX FPGA Board.
Warning: smMatVLX('H'): To show this help mesasge.

ans =
```

SMIMS VeriLink Blockset Description

The VeriLink is a way of the bridge between system-level design and hardware design. With ready HDL design, in clicks, users can automatically feed binary to SMIMS FPGA and review the result in Simulink environment. It maps user's hardware circuit (either HDL design or configuration file) to Simulink blocks. By taking the advantages of the existing MATLAB functions and Simulink blocks to link with FPGA prototype, users can accomplish HDL design verification easily and quickly.

There are five blocks in SMIMS VeriLink directory of Simulink browser. The detail description for each block is shown as follow.

SMIMS VeriLink Binary Code Generator



Co-simulation with a hardware component by applying input signals to and reading output signals from a FPGA bit file model under emulation in SMIMS FPGA Platform.

Library

SMIMS VeriLink

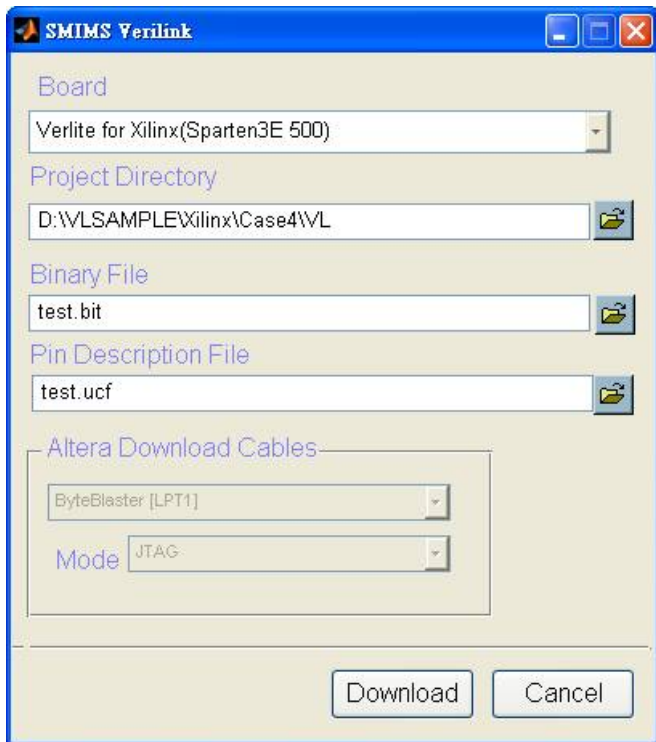
Description


The SMIMS VeriLink Binary Code Generator block co-simulates a hardware component by applying input signals to and reading output signals from a FPGA bit file model under emulation in SMIMS FPGA Platform.

Data Type

The SMIMS VeriLink Binary Code Generator block accepts the double data type.

Dialog Box



- **Board**
All available SMIMS-supported boards list. Please select SMIMS board type.
- **Project Directory**
Specify the project directory, where the binary download file and pin description file are residents.
User can also select by clicking the  icon in the right.
- **Binary File**
Specify the FPGA download/configuration file.
The .bit file is used for VeriLite/VeriEnterprise Xilinx board.
The .rbf file is used for VeriLite Altera board.

The .sof file is in the JTAG mode for VeriEnterprise Altera board.

The .pof file is in the AS mode for VeriEnterprise Altera board.

- Pin Description File

Please fill the pin description file name in this field in order to get the signal information. User can also

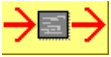
select by clicking the  icon in the right.

The pin description is specifying the pin location of hardware component's I/Os on FPGA. Please refer to the pin table of SMIMS Help.

- Altera Download Cables

As "VeriEnterprise for Altera" are selected, the user has to define the download cable and download mode in this panel.

SMIMS VeriLink HDL Code Generator



Co-simulation with a hardware component by applying input signals to and reading output signals from a HDL Code file model under emulation in SMIMS FPGA Platform.

Library

SMIMS VeriLink

Description

The SMIMS VeriLink HDL Code Generator block co-simulates a hardware component by applying input signals to and reading output signals from a FPGA bit file model under emulation in SMIMS FPGA Platform.

Data Type

The SMIMS VeriLink HDL Code Generator block accepts the double data type.

Dialog Box

SMIMS VeriLink

SMIMS Board
VeriLite for Xilinx(Spartan3E 250)

Project Directory
D:\SMIMS_SW\VeriLink\Test

Top Module File
Verilog "ImgProc.v"

Pin Description File
☒ Auto generate Pin Description File

Altera Download Cables
ByteBlasterII
Mode: JTAG

VeriLink Release V2.2 2009Q3

Generate Cancel

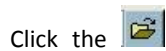
- **SMIMS Board**

All available VeriLink-supported boards list. Please select SMIMS board type.

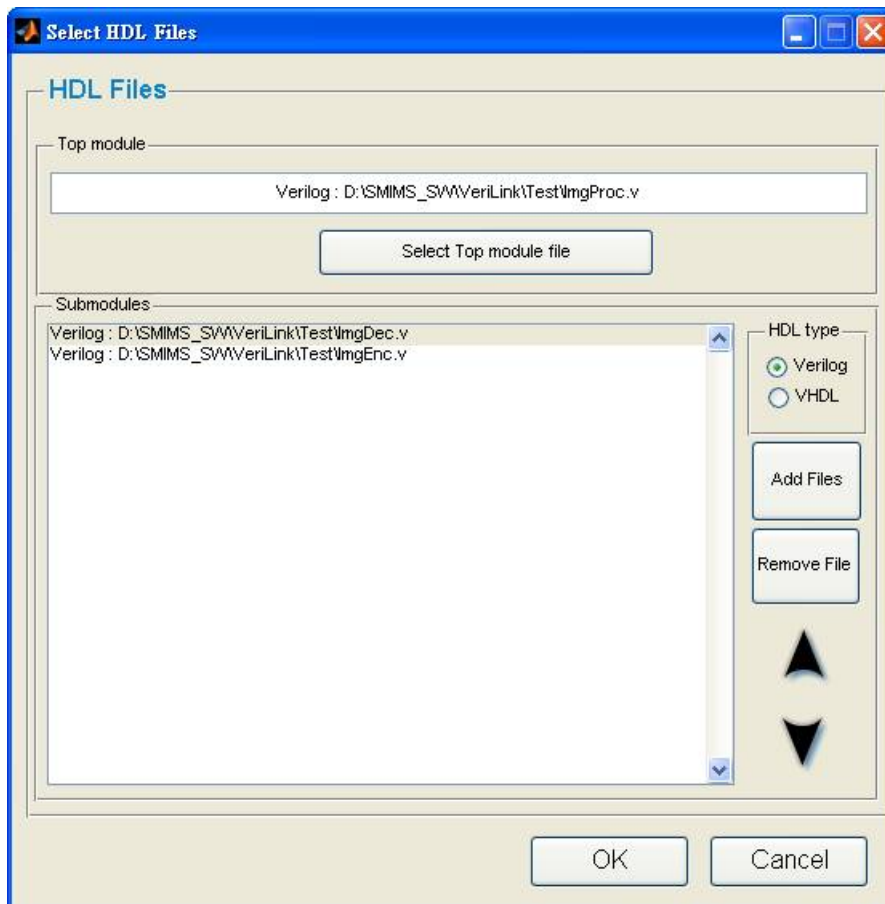
- **Project Directory**

The project directory is specifying the directory of the hardware component location.

- **Top Module File**



Click the icon in the right to open the setup dialog. Then Specific the top module and sub-module file name.



- **Pin Description File**

The pin description is specifying the pin location of hardware component's I/Os on FPGA. Please refer to the pin assignment list in appendix.

There is .ucf file in Xilinx.

There is .qsf file in Altera.

Or choose "Auto generate Pin Description File" to generate the pin description file by VeriLink automatically.

- **Altera Download Cables**

As "VeriEnterprise for Altera" are selected, the user has to define the download cable and download

mode in this panel.

2's Complement to Real



Library

SMIMS VeriLink

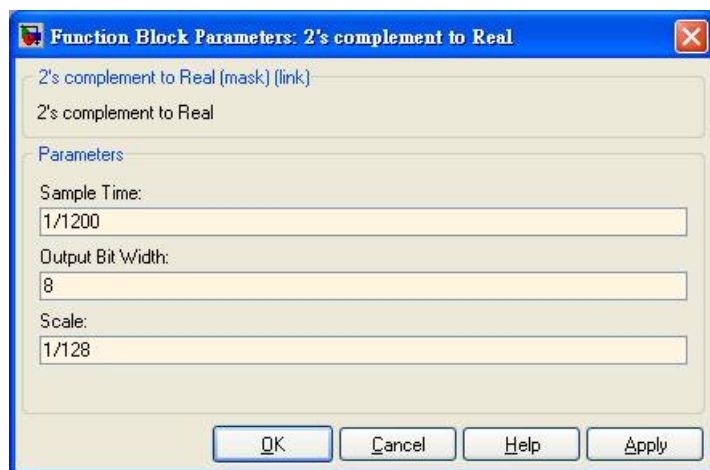
Description

This block is used to convert and scale 2's complement input to real format.

Data Type

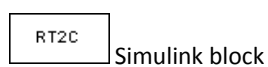
The SMIMS VeriLink 2's Complement to Real block accepts the double data type.

Dialog Box



- Sample Time
Specify the time interval between samples. To inherit the sample time, set this parameter to -1.
- Input Bit Width
Set bit-width of 2's complement data.
- Scale
To Scale the data by specified value (power of 2).

Real to 2's Complement



Library

SMIMS VeriLink

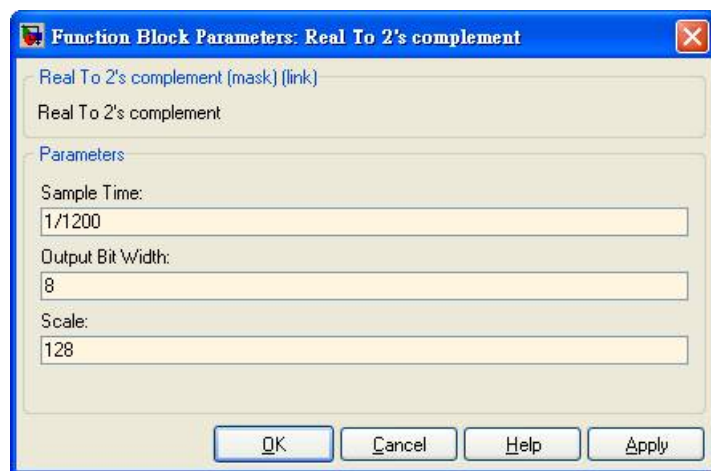
Description

This block is used to convert and scale real format input to 2's complement

Data Type

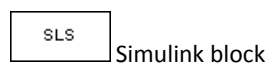
The SMIMS VeriLink Real to 2's Complement block accepts the double data type.

Dialog Box



- Sample Time
Specify the time interval between samples. To inherit the sample time, set this parameter to -1.
- Output Bit Width
Set bit-width of 2's complement data.
- Scale
To Scale the data by specified value (power of 2).

Signal Level shift



Library

SMIMS VeriLink

Description

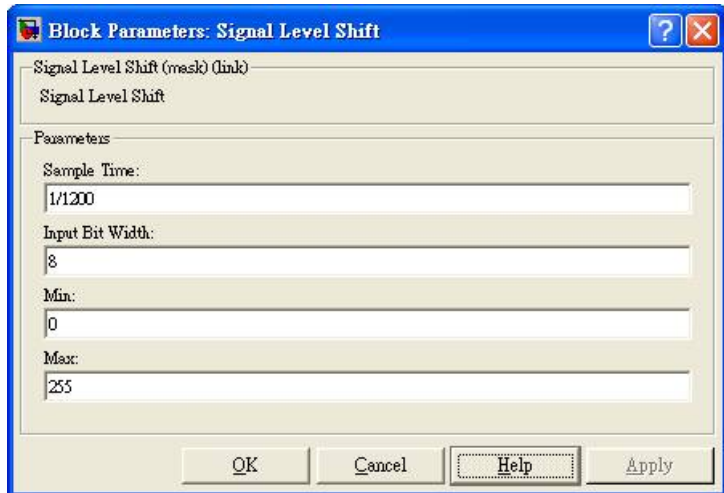
This block mapped the input signal to user-specified level.

$$\text{output value} = (\text{Max} - \text{Min}) * \text{input_value} / 2^{\text{Input_Bit_Width}}$$

Data Type

The SMIMS VeriLink Signal Level Shift block accepts the double data type.

Dialog Box

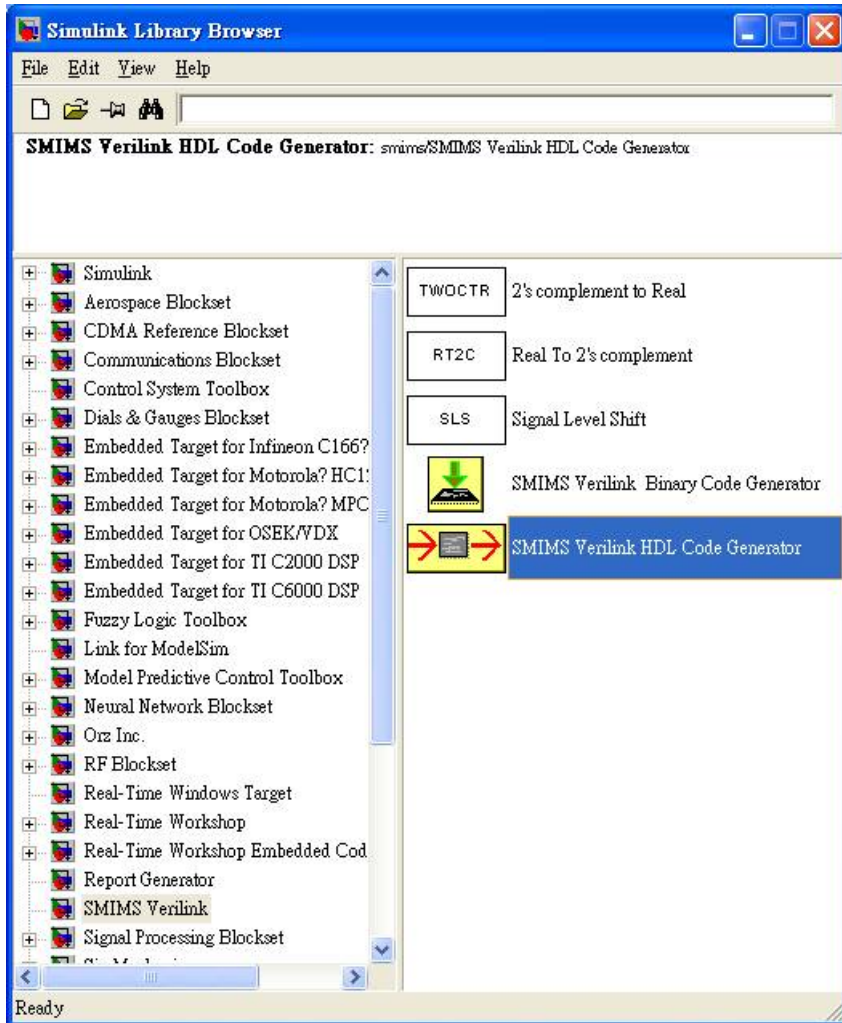


- Sample Time
Specify the time interval between samples. To inherit the sample time, set this parameter to -1.
- Input Bit Width
Specify the bit width of input signal.
- Min
Specify the minimum value of mapped signal
- Max
Specify the maximum value of mapped signal

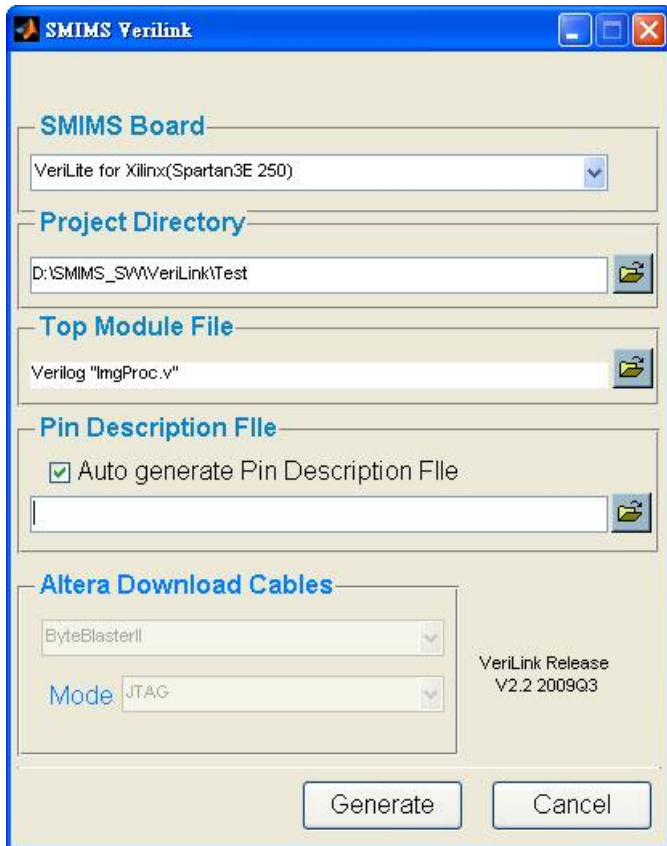
Build your design to Simulink

Build Simulink Block from HDL Code

To generate a Simulink block for HDL design, first expand the Library Browser tree to display the blocks in the SMIMS VeriLink library. Click the node to select the SMIMS VeriLink HDL Code Generator block. Library Browser shows below.



Double Click the SMIMS VeriLink HDL Code Generator Block to set block parameter.

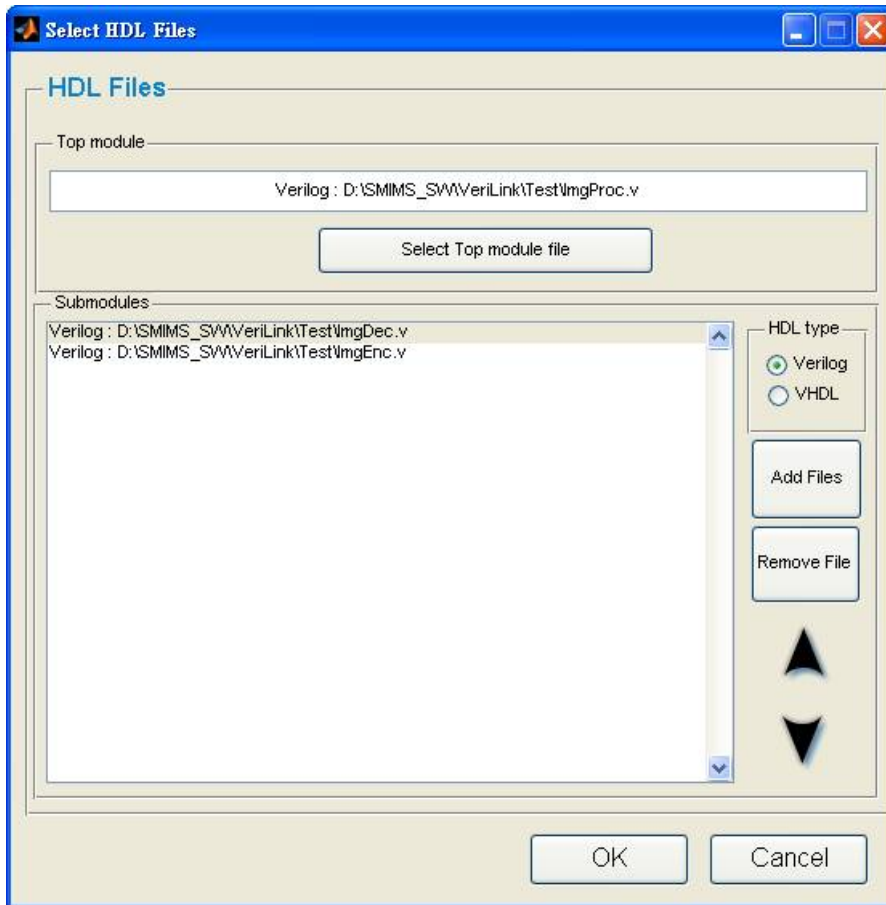


Specify the SMIMS board and project directory which may select it from  button.

The log files and compiler information of building blocks will be saved in the VL directory of this project directory.



Specify the top module and sub-module file.



The pin description is specifying the pin location of hardware component's I/Os on FPGA. Please refer to the pin table in SMIMS Help.

There is .ucf file in Xilinx.

There is .qsf file in Altea.

Or choose "Auto generate Pin Description File" to generate the pin description file by VeriLink automatically.

Note : the bi-directional pin is not supported in this version.

As "VeriEnterprise for Altera" are selected, the user has to define the download cable and download mode in this panel.

Click Generate button.



```
SMIMS VeriLink for Xilinx(Spartan3E 500)
INFO:Timing:2761 - N/A entries in the Constraints list may indicate that the
constraint does not cover any paths or that it has no requested value.
Generating Pad Report.

All signals are completely routed.

Total REAL time to PAR completion: 22 secs
Total CPU time to PAR completion: 20 secs

Peak Memory Usage: 179 MB

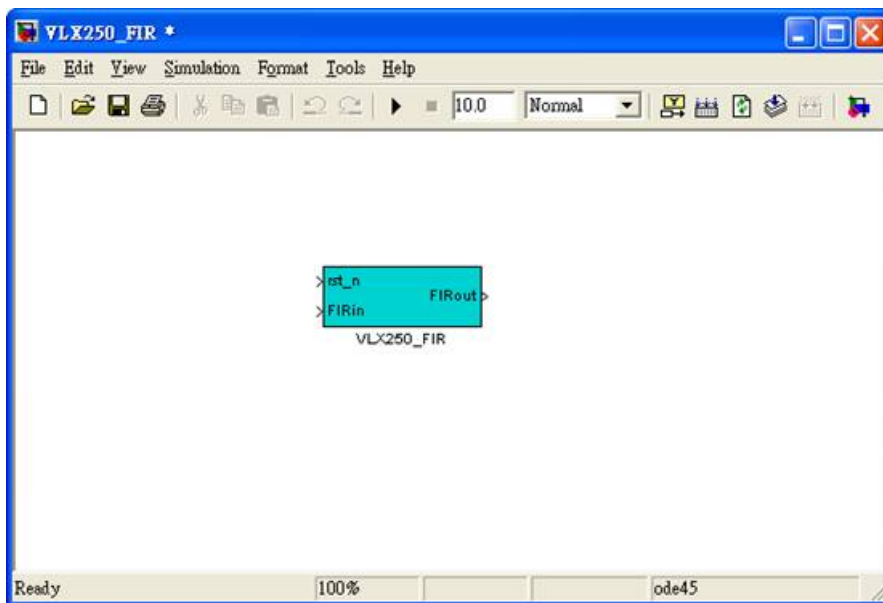
Placement: Completed - No errors found.
Routing: Completed - No errors found.

Number of error messages: 0
Number of warning messages: 0
Number of info messages: 1

Writing design to file FIR.ncd

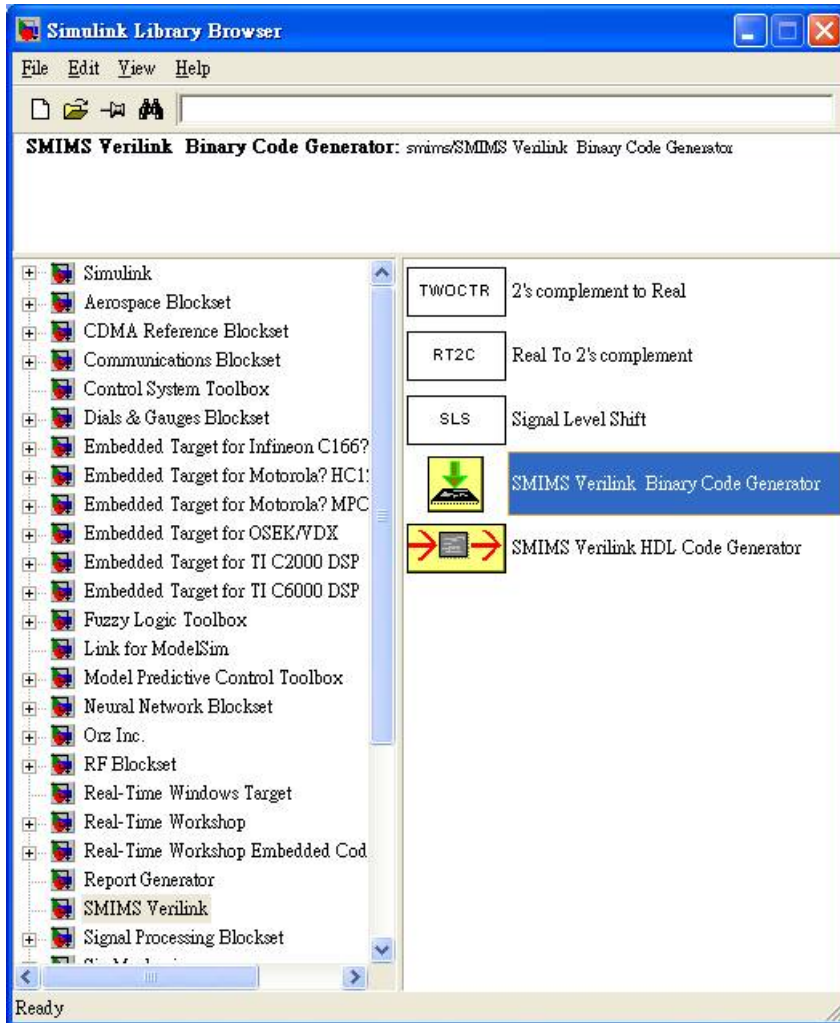
PAR done!
Downloading D:\SMIMS\Demo\FIR\VLX500\VL\FIR.bit
Downloading ..... 9%
```

After compile and download processes had done, shown as follows. Double-click to download the design to FPGA again.

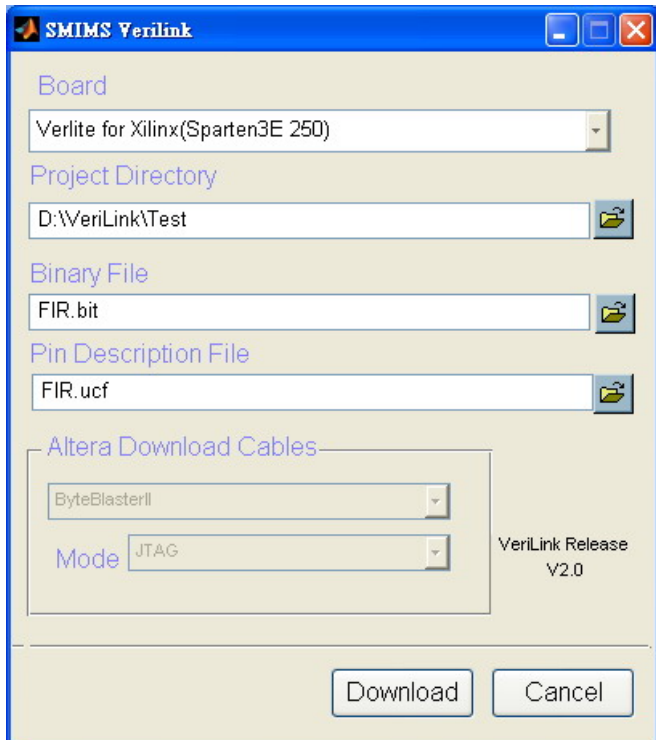


Build Simulink Block from Configuration File

To generate a Simulink block for compiled design from configuration file, first expand the Library Browser tree to display the blocks in the SMIMS VeriLink library. Click the node to select the SMIMS VeriLink Binary Code Library Browser shows below.



Double Click the SMIMS VeriLink Binary Code Generator Block to set block parameter.



Specify the SMIMS board and project directory which may select it from  button.



Specify the FPGA download/configuration file.

The .bit file is used for Xilinx board.

The .sof file is in the JTAG mode for Altera board.

The .pof file is in the AS mode for Altera board.

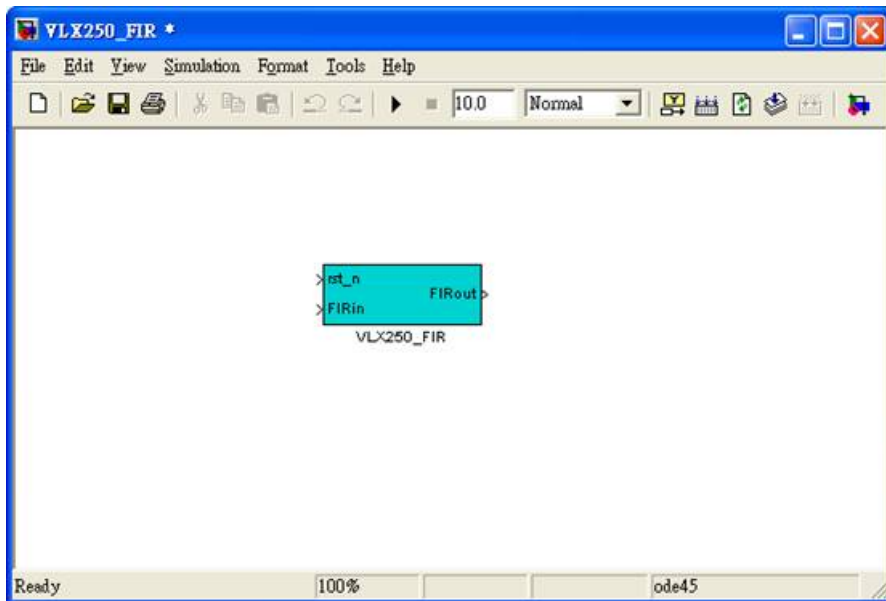
The pin description is specifying the pin location of hardware component's I/Os on FPGA. Please refer to the pin table in SMIMS Help.

As “VeriEnterprise for Altera” are selected, specify the download cable and download mode in this panel.

Click Generate button.



After download process had done, shown as follows. Double-click to download the design to FPGA again.



Getting Start


Building a Model

Creating a New Model

Starting Simulink

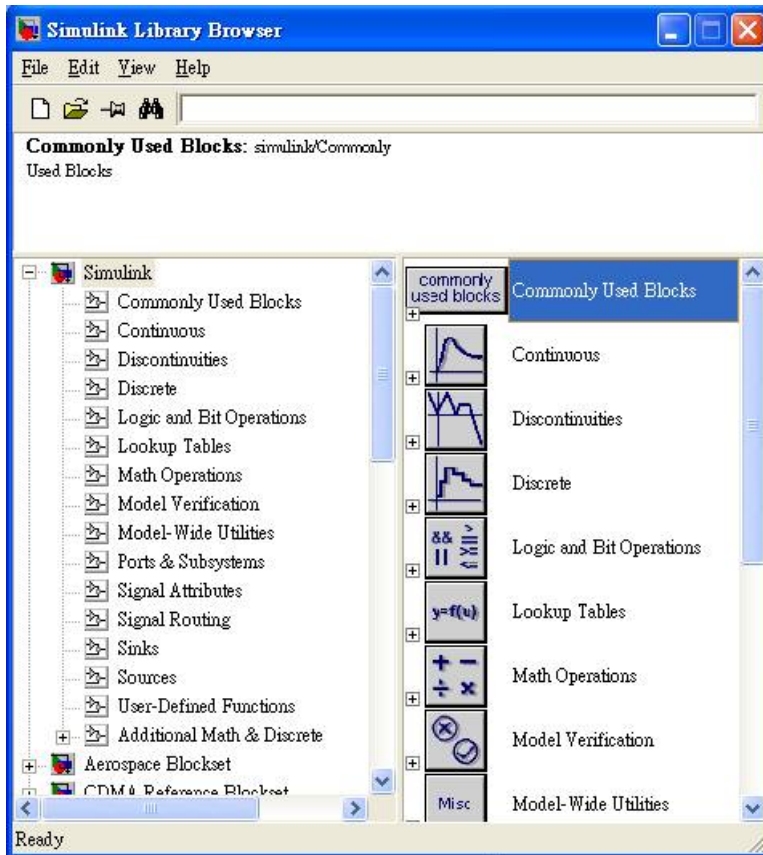
To create the model, first enter “simulink” in the MATLAB Command Window.



Or click the  button.



And, the Simulink Library Browser appears.

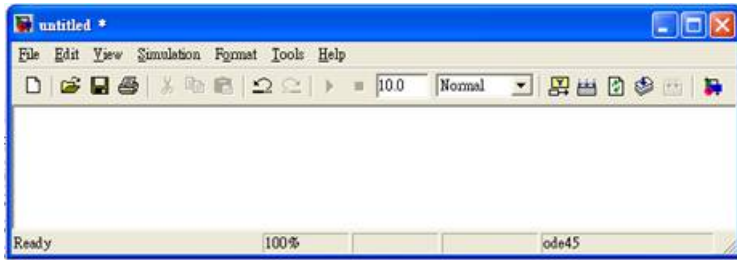


Creating a New Model

Click the New Model button on the Library Browser's toolbar to create a new model on Windows.



Simulink opens a new model window.

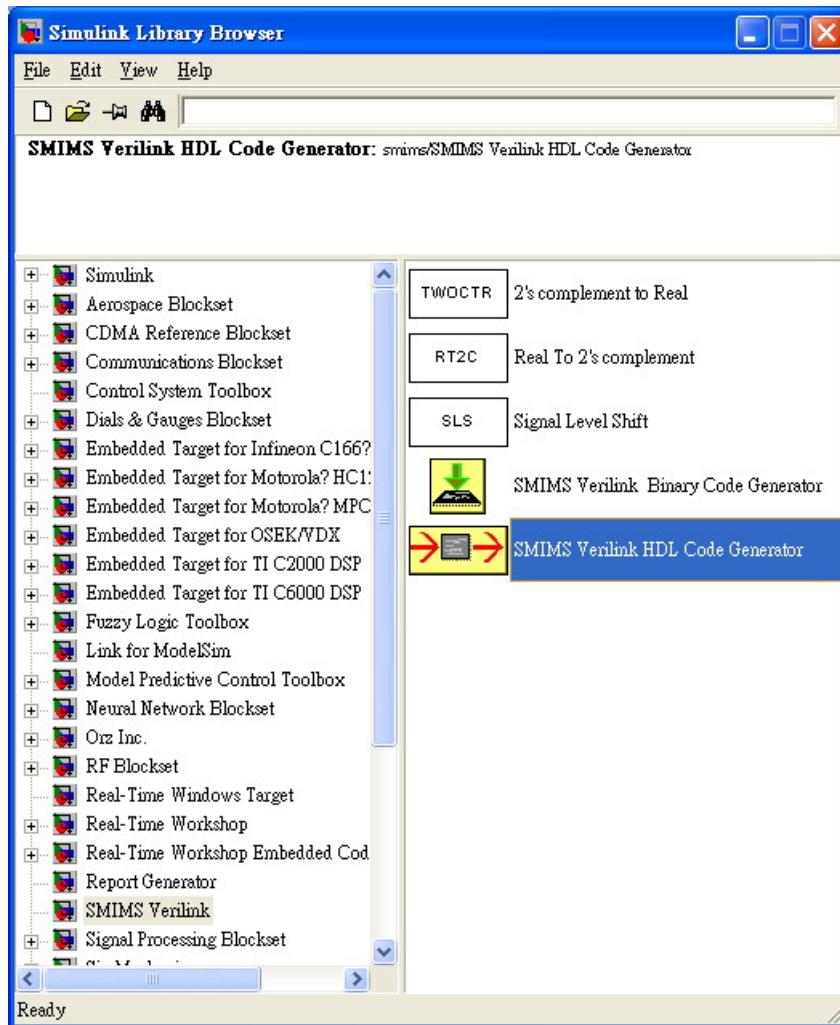


The Simulink displays a status bar at the bottom of each model and library window.

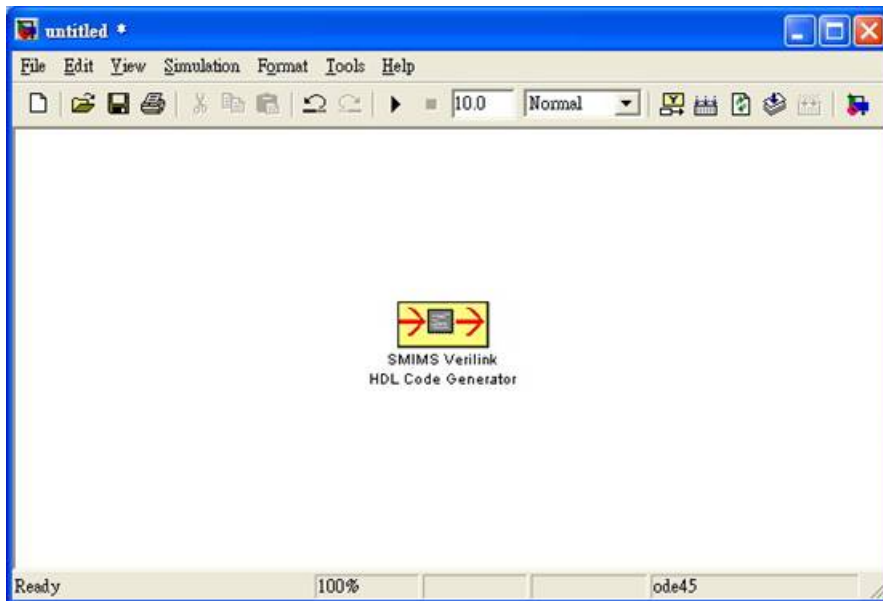
When a simulation is running, the status bar displays the status of the simulation, including the current simulation time and the name of the current solver. You can display or hide the status bar by selecting or clearing the Status Bar option on the Simulink View menu

To create your model, you need to copy blocks into the model from the Simulink block libraries. Please refer for Simulink Help for block information.

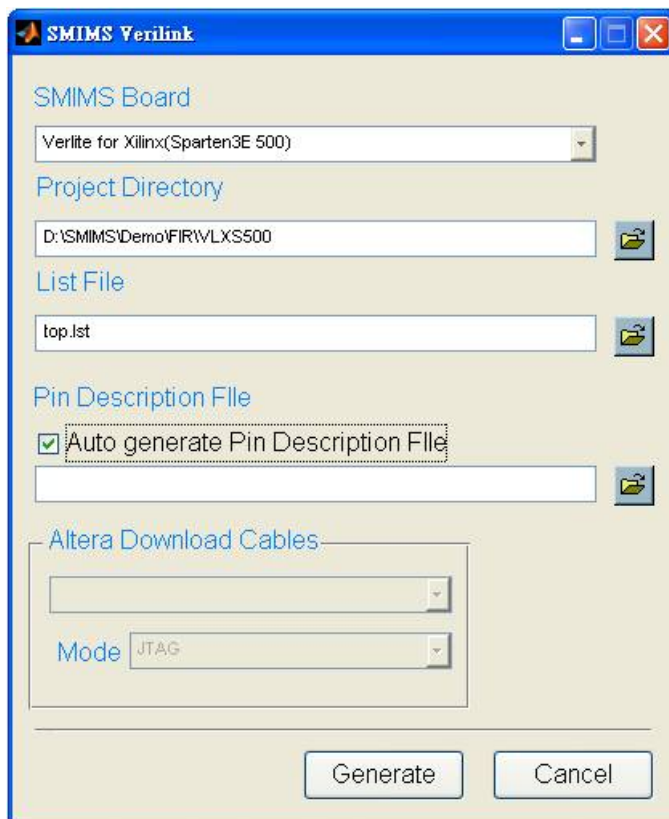
For example, you want to generate a Simulink block for your design, first expand the Library Browser tree to display the blocks in the SMIMS VeriLink library. Click the node to select the SMIMS VeriLink HDL Code Generator block. Here is how the Library Browser should look after you have done this.



Now drag a copy of the SMIMS VeriLink HDL Code Generator block from the browser and drop it in the model window.



Every block that has block-specific parameters has a dialog box that you can use to view and set the parameters. Double Click to set the block parameters.



Click Generate button. If there is the VL directory which saved the previous log of generating your hardware

design to Simulink block, you will see the follows



Click Yes, to continue the process. A Dos window will show up to indicate the compile process information. All the compilation information will save in the VL directory of your project directory.

```
SMIMS VeriLite for Xilinx(Sparten3E 500)
INFO:Timing:2761 - N/A entries in the Constraints list may indicate that the
constraint does not cover any paths or that it has no requested value.
Generating Pad Report.

All signals are completely routed.

Total REAL time to PAR completion: 22 secs
Total CPU time to PAR completion: 20 secs

Peak Memory Usage: 179 MB

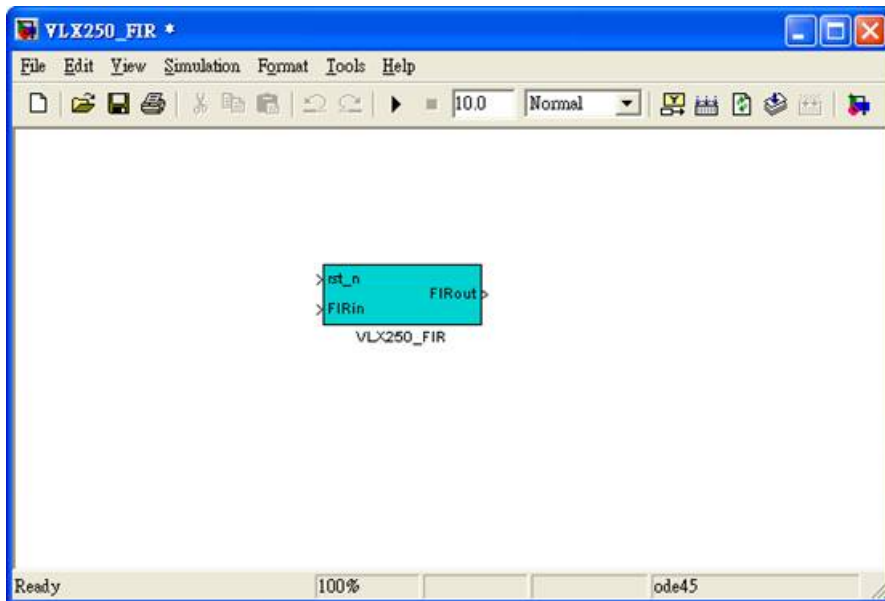
Placement: Completed - No errors found.
Routing: Completed - No errors found.

Number of error messages: 0
Number of warning messages: 0
Number of info messages: 1

Writing design to file FIR.ncd

PAR done!
Downloading D:\SMIMS\Demo\FIR\VLX500\VL\FIR.bit
Downloading ..... 9%
```

Then it finished, the DOS window will disappear. And you will see the generated block of your design named as VLX500_FIR (or VLX250_FIR/VEX_FIR/VEA_FIR/VLA_FIR etc)

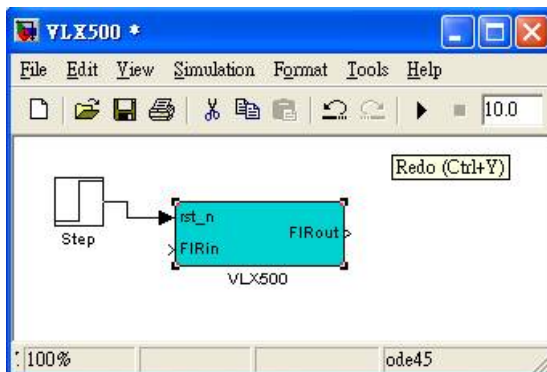


Double-click to download the design to FPGA again.

Auto Connecting Two Blocks

1. Select the source block.
2. Hold down Ctrl and left-click the destination block.

Simulink connects the source block to the destination block, routing the line around intervening blocks if necessary. When connecting two blocks, Simulink draws as many connections as possible between the two blocks as illustrated in the following example.



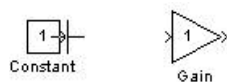
Manually Connecting Blocks

Simulink allows you to draw lines manually between blocks or between lines and blocks. You might

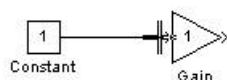
want to do this if you need to control the path of the line or to create a branch line.

Drawing a Line Between Blocks

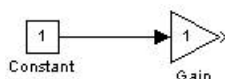
1. To connect the output port of one block to the input port of another block:



2. Press and hold down the mouse button.
3. Drag the pointer to the second block's input port. You can position the cursor on or near the port or in the block. If you position the cursor in the block, the line is connected to the closest input port. The cursor shape changes to double crosshairs.



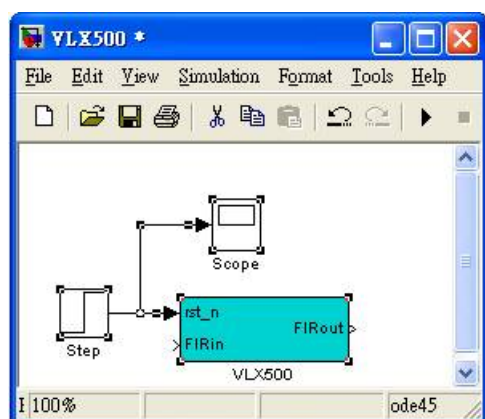
4. Release the mouse button. Simulink replaces the port symbols by a connecting line with an arrow showing the direction of the signal flow. You can create lines either from output to input, or from input to output. The arrow is drawn at the appropriate input port, and the signal is the same.



Simulink draws connecting lines using horizontal and vertical line segments. To draw a diagonal line, hold down the Shift key while drawing the line.

Drawing a Branch Line

A branch line is a line that starts from an existing line and carries its signal to the input port of a block. Both the existing line and the branch line carry the same signal. Using branch lines enables you to cause one signal to be carried to more than one block.



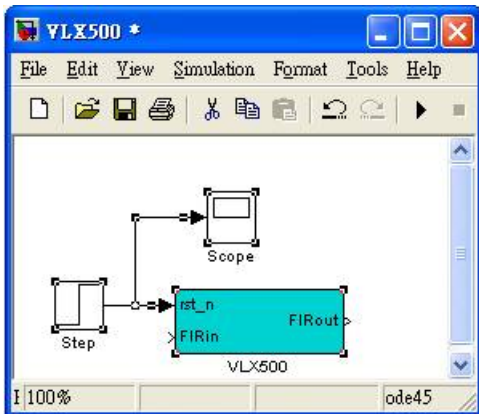
To add a branch line, follow these steps:

1. Position the pointer on the line where you want the branch line to start.
2. While holding down the Ctrl key, press and hold down the left mouse button.

3. Drag the pointer to the input port of the target block, then release the mouse button and the Ctrl key. You can also use the right mouse button instead of holding down the left mouse button and the Ctrl key

Drawing a Branch Line

A branch line is a line that starts from an existing line and carries its signal to the input port of a block. Both the existing line and the branch line carry the same signal. Using branch lines enables you to cause one signal to be carried to more than one block.



To add a branch line, follow these steps:

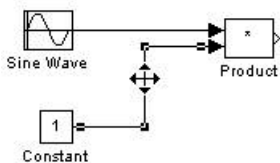
1. Position the pointer on the line where you want the branch line to start.
2. While holding down the Ctrl key, press and hold down the left mouse button.
3. Drag the pointer to the input port of the target block, then release the mouse button and the Ctrl key.

You can also use the right mouse button instead of holding down the left mouse button and the Ctrl key

Moving a Line Segment

To move a line segment, follow these steps:

1. Position the pointer on the segment you want to move.
2. Press and hold down the left mouse button.

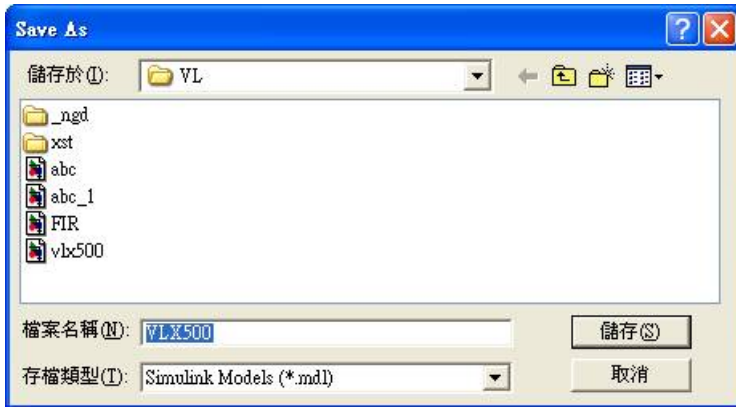


3. Drag the pointer to the desired location.
4. Release the mouse button.

To move the segment connected to an input port, position the pointer over the port and drag the end of the segment to the new location. You cannot move the segment connected to an output port.

Saving a Model

You can save a model by choosing either the Save or Save As command from the File menu. Simulink saves the model by generating a specially formatted file called the model file (with the .mdl extension) that contains the block diagram and block properties. If you are saving a model for the first time, use the Save command to provide a name and location for the model file.

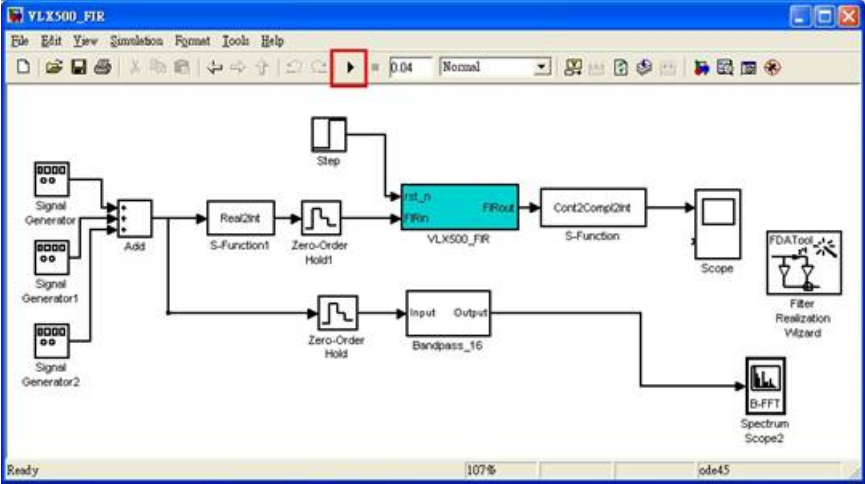


Model file names must start with a letter and can contain no more than 63 letters, numbers, and underscores. The file name must not be the same as that of a MATLAB command. If you are saving a model whose model file was previously saved, use the Save command to replace the file's contents or the Save As command to save the model with a new name or location. You can also use the Save As command to save the model in a format compatible with previous releases of Simulink

Running a model

An interesting demo program provided with SMIMS VeriLink examples. To run those demos, follow these steps:

1. Start MATLAB. See your MATLAB documentation if you're not sure how to do this.
2. Run the demo model by typing specify example file name in the MATLAB Command Window. This command starts up Simulink and creates a model window that contains this model.
3. Delete the blue block and re-create it. Please follow up instructions in the "Build your design to Simulink Block" section.
4. To start the simulation, pull down the Simulation menu and choose the Start command (or click the Start button on the Simulink toolbar). When you're finished running the simulation, close the model by choosing Close from the File menu.



For the detailed Simulink usage and information, please refers to Matlab Help.

FPGA Pin Map On Simulink

VeriEnterprise Xilinx NV5 USB

SMIMS FPGA Pin Map

In this chapter, the FPGA pin, which connects to other devices such as I/O connectors, clocks etc, will be shown in the following tables.

Pin Tables

In this chapter, the FPGA pin, which connects to other devices such as I/O connectors, clocks etc, will be shown in the following tables.

XC5VLX110FF1760

	Pin Name	Comment
clk	K15	User circuit clock pin connect to this pin
input0	E7	Circuit input signal
input1	E8	
input2	K8	
input3	K9	

input4	H8	
input5	J8	
input6	F9	
input7	G9	
input8	E10	
input9	E9	
input10	H9	
input11	H10	
input12	F10	
input13	F11	
input14	F12	
input15	G11	
input16	E12	
input17	E13	
input18	H11	
input19	G12	
input20	H13	
input21	G13	
input22	J10	
input23	J11	
input24	K12	
input25	J12	
input26	L11	
input27	L10	
input28	N11	
input29	M11	
input30	M12	
input31	L12	
input32	P10	
input33	N10	
input34	P12	
input35	P11	
input36	M13	
input37	N13	
input38	N14	
input39	P13	
input40	N15	
input41	P15	
input42	AV3	
input43	AU3	

input44	AT5	
input45	AT6	
input46	AU4	
input47	AT4	
input48	AV5	
input49	AV4	
input50	AP7	
input51	AP6	
input52	AR5	
input53	AP5	
input54	AN8	
input55	AM8	
Input56	AM7	
Input57	AN6	
Input58	AL9	
Input59	AK9	
Input60	AK8	
Input61	AL7	
Input62	AH11	
Input63	AG11	
Output0	AL11	Circuit output signal
Output1	AL12	
Output2	AK10	
Output3	AL10	
Output4	AJ10	
Output5	AJ11	
Output6	AJ12	
Output7	AK12	
Output8	AJ13	
Output9	AK13	
Output10	AK14	
Output11	AK15	
Output12	AL15	
Output13	AL14	
Output14	AH16	
Output15	AJ15	
Output16	AJ16	
Output17	AJ17	

Output18	AJ26	
Output19	AJ27	
Output20	AJ28	
Output21	AK29	
Output22	G6	
Output23	F6	
Output24	F5	
Output25	E5	
Output26	G7	
Output27	H6	
Output28	J5	
Output29	H5	
Output30	J7	
Output31	K7	
Output32	K5	
Output33	J6	
Output34	M7	
Output35	M6	
Output36	L7	
Output37	L6	
Output38	P8	
Output39	N9	
Output40	M8	
Output41	M9	
Output42	P7	
Output43	N8	
Output44	N6	
Output45	P6	
Output46	T10	
Output47	R10	
Output48	R9	
Output49	T9	
Output50	R7	
Output51	R8	
Output52	U7	
Output53	T7	
Output54	U8	
Output55	U9	
Output56	U11	
Output57	T11	

Output58	V9	
Output59	V8	
Output60	V11	
Output61	V10	
Output62	F7	
Output63	G8	

XC5VLX220FF1760

	Pin Name	Comment
clk	K15	User circuit clock pin connect to this pin
input0	E7	Circuit input signal
input1	E8	
input2	K8	
input3	K9	
input4	H8	
input5	J8	
input6	F9	
input7	G9	
input8	E10	
input9	E9	
input10	H9	
input11	H10	
input12	F10	
input13	F11	
input14	F12	
input15	G11	
input16	E12	
input17	E13	
input18	H11	
input19	G12	
input20	H13	
input21	G13	
input22	J10	
input23	J11	

input24	K12	
input25	J12	
input26	L11	
input27	L10	
input28	N11	
input29	M11	
input30	M12	
input31	L12	
input32	P10	
input33	N10	
input34	P12	
input35	P11	
input36	M13	
input37	N13	
input38	N14	
input39	P13	
input40	N15	
input41	P15	
input42	AV3	
input43	AU3	
input44	AT5	
input45	AT6	
input46	AU4	
input47	AT4	
input48	AV5	
input49	AV4	
input50	AP7	
input51	AP6	
input52	AR5	
input53	AP5	
input54	AN8	
input55	AM8	
Input56	AM7	
Input57	AN6	
Input58	AL9	
Input59	AK9	
Input60	AK8	
Input61	AL7	
Input62	AH11	
Input63	AG11	

Output0	AL11	Circuit output signal
Output1	AL12	
Output2	AK10	
Output3	AL10	
Output4	AJ10	
Output5	AJ11	
Output6	AJ12	
Output7	AK12	
Output8	AJ13	
Output9	AK13	
Output10	AK14	
Output11	AK15	
Output12	AL15	
Output13	AL14	
Output14	AH16	
Output15	AJ15	
Output16	AJ16	
Output17	AJ17	
Output18	AJ26	
Output19	AJ27	
Output20	AJ28	
Output21	AK29	
Output22	G6	
Output23	F6	
Output24	F5	
Output25	E5	
Output26	G7	
Output27	H6	
Output28	J5	
Output29	H5	
Output30	J7	
Output31	K7	
Output32	K5	
Output33	J6	
Output34	M7	
Output35	M6	
Output36	L7	
Output37	L6	

Output38	P8	
Output39	N9	
Output40	M8	
Output41	M9	
Output42	P7	
Output43	N8	
Output44	N6	
Output45	P6	
Output46	T10	
Output47	R10	
Output48	R9	
Output49	T9	
Output50	R7	
Output51	R8	
Output52	U7	
Output53	T7	
Output54	U8	
Output55	U9	
Output56	U11	
Output57	T11	
Output58	V9	
Output59	V8	
Output60	V11	
Output61	V10	
Output62	F7	
Output63	G8	

XC5VLX330FF1760

	Pin Name	Comment
clk	K15	User circuit clock pin connect to this pin
input0	E7	Circuit input signal
input1	E8	
input2	K8	
input3	K9	

input4	H8	
input5	J8	
input6	F9	
input7	G9	
input8	E10	
input9	E9	
input10	H9	
input11	H10	
input12	F10	
input13	F11	
input14	F12	
input15	G11	
input16	E12	
input17	E13	
input18	H11	
input19	G12	
input20	H13	
input21	G13	
input22	J10	
input23	J11	
input24	K12	
input25	J12	
input26	L11	
input27	L10	
input28	N11	
input29	M11	
input30	M12	
input31	L12	
input32	P10	
input33	N10	
input34	P12	
input35	P11	
input36	M13	
input37	N13	
input38	N14	
input39	P13	
input40	N15	
input41	P15	
input42	AV3	
input43	AU3	

input44	AT5	
input45	AT6	
input46	AU4	
input47	AT4	
input48	AV5	
input49	AV4	
input50	AP7	
input51	AP6	
input52	AR5	
input53	AP5	
input54	AN8	
input55	AM8	
Input56	AM7	
Input57	AN6	
Input58	AL9	
Input59	AK9	
Input60	AK8	
Input61	AL7	
Input62	AH11	
Input63	AG11	
Output0	AL11	Circuit output signal
Output1	AL12	
Output2	AK10	
Output3	AL10	
Output4	AJ10	
Output5	AJ11	
Output6	AJ12	
Output7	AK12	
Output8	AJ13	
Output9	AK13	
Output10	AK14	
Output11	AK15	
Output12	AL15	
Output13	AL14	
Output14	AH16	
Output15	AJ15	
Output16	AJ16	
Output17	AJ17	

Output18	AJ26	
Output19	AJ27	
Output20	AJ28	
Output21	AK29	
Output22	G6	
Output23	F6	
Output24	F5	
Output25	E5	
Output26	G7	
Output27	H6	
Output28	J5	
Output29	H5	
Output30	J7	
Output31	K7	
Output32	K5	
Output33	J6	
Output34	M7	
Output35	M6	
Output36	L7	
Output37	L6	
Output38	P8	
Output39	N9	
Output40	M8	
Output41	M9	
Output42	P7	
Output43	N8	
Output44	N6	
Output45	P6	
Output46	T10	
Output47	R10	
Output48	R9	
Output49	T9	
Output50	R7	
Output51	R8	
Output52	U7	
Output53	T7	
Output54	U8	
Output55	U9	
Output56	U11	
Output57	T11	

Output58	V9	
Output59	V8	
Output60	V11	
Output61	V10	
Output62	F7	
Output63	G8	

FPGA Dedicated IO

Dedicated IO

Pin table of FPGA U1 dedicated IO (J1)

J1 (Dedicated IO)					
Top/Bottom Connector	FPGA IO Pin	IO Description	Top/Bottom Connector	FPGA IO Pin	IO Description
J1.001		VDD5V	J1.061		GND
J1.002		VDD5V	J1.062		GND
J1.003	D1	IO_L19P_28	J1.063	C21	IO_L9P_CC_32
J1.004	A2	IO_L18P_28	J1.064	C23	IO_L8P_CC_32
J1.005	D2	IO_L19N_28	J1.065	C20	IO_L9N_CC_32
J1.006	B2	IO_L18N_28	J1.066	B22	IO_L8N_CC_32
J1.007	C1	IO_L17P_28	J1.067	C26	IO_L7P_32
J1.008	B3	IO_L16P_28	J1.068	B23	IO_L6P_32
J1.009	B1	IO_L17N_28	J1.069	B26	IO_L7N_32
J1.010	C3	IO_L16N_28	J1.070	A24	IO_L6N_32
J1.011	D3	IO_L15P_28	J1.071	C24	IO_L5P_32
J1.012	C5	IO_L14P_28	J1.072	B24	IO_L4P_32
J1.013	C4	IO_L15N_28	J1.073	C25	IO_L5N_32
J1.014	D5	IO_L14N_VREF_28	J1.074	A25	IO_L4N_32
J1.015	B4	IO_L13P_28	J1.075	B28	IO_L3P_32
J1.016	A7	IO_L11P_CC_28	J1.076	C29	IO_L2P_32
J1.017	A4	IO_L13N_28	J1.077	A27	IO_L3N_32
J1.018	B7	IO_L11N_CC_28	J1.078	C28	IO_L2N_32
J1.019	D7	IO_L10P_CC_28	J1.079	B27	IO_L1P_32
J1.020	B6	IO_L9P_CC_28	J1.080	A29	IO_L0P_32

J1.021	D6	IO_L10N_CC_28	J1.081	A26	IO_L1N_32
J1.022	C6	IO_L9N_CC_28	J1.082	B29	IO_L0N_32
J1.023	C8	IO_L8P_CC_28	J1.083	E14	IO_L0P_31
J1.024	A9	IO_L7P_28	J1.084	E15	IO_L1P_31
J1.025	B8	IO_L8N_CC_28	J1.085	D15	IO_L0N_31
J1.026	B9	IO_L7N_28	J1.086	F14	IO_L1N_31
J1.027	C10	IO_L6P_28	J1.087	D16	IO_L2P_31
J1.028	A11	IO_L5P_28	J1.088	F16	IO_L3P_31
J1.029	D10	IO_L6N_28	J1.089	D17	IO_L2N_31
J1.030	A10	IO_L5N_28	J1.090	F15	IO_L3N_31
J1.031		GND	J1.091		GND
J1.032		GND	J1.092		GND
J1.033	C9	IO_L4P_28	J1.093	F17	IO_L4P_31
J1.034	D11	IO_L3P_28	J1.094	E20	IO_L5P_31
J1.035	D8	IO_L4N_VREF_28	J1.095	E17	IO_L4N_VREF_31
J1.036	D12	IO_L3N_28	J1.096	F20	IO_L5P_31
J1.037	B12	IO_L2P_28	J1.097	D18	IO_L6P_31
J1.038	B11	IO_L1P_28	J1.098	E19	IO_L7P_31
J1.039	A12	IO_L2N_28	J1.099	E18	IO_L6N_31
J1.040	C11	IO_L1N_28	J1.100	F19	IO_L7N_31
J1.041	C13	IO_L0P_28	J1.101	D22	IO_L8P_CC_31
J1.042	B13	IO_L19P_32	J1.102	D21	IO_L9P_CC_31
J1.043	D13	IO_L0N_28	J1.103	D23	IO_L8N_CC_31
J1.044	B14	IO_L19N_32	J1.104	D20	IO_L9N_CC_31
J1.045	C14	IO_L18P_32	J1.105	F21	IO_L10P_CC_31
J1.046	B16	IO_L17P_32	J1.106	E23	IO_L11P_CC_31
J1.047	C15	IO_L18N_32	J1.107	E22	IO_L10N_CC_31
J1.048	C16	IO_L17N_32	J1.108	F24	IO_L11N_CC_31
J1.049	A14	IO_L16P_32	J1.109	E24	IO_L13P_31
J1.050	B17	IO_L15P_32	J1.110	F27	IO_L14P_31
J1.051	A15	IO_L16N_32	J1.111	E25	IO_L13N_31
J1.052	B18	IO_L15N_32	J1.112	E27	IO_L14N_VREF_31
J1.053	A16	IO_L14P_32	J1.113	F26	IO_L15P_31
J1.054	C18	IO_L13P_32	J1.114	E29	IO_L16P_31
J1.055	A17	IO_L14N_VREF_32	J1.115	F25	IO_L15N_31
J1.056	C19	IO_L13N_32	J1.116	E28	IO_L16N_31
J1.057	A21	IO_L11P_CC_32	J1.117	D27	IO_L17P_31
J1.058	A20	IO_L10P_CC_32	J1.118	F30	IO_L18P_31
J1.059	A22	IO_L11N_CC_32	J1.119	D28	IO_L17N_31
J1.060	B21	IO_L10N_CC_32	J1.120	F29	IO_L18N_31

Pin table of FPGA dedicated IO (J4)

J4 (Dedicated IO)					
Top/Bottom Connector	FPGA IO Pin	IO Description	Top/Bottom Connector	FPGA IO Pin	IO Description
J4.001		VDD5V	J4.061		GND
J4.002		VDD5V	J4.062		GND
J4.003	AV1	IO_L0P_30	J4.063	BB21	IO_L9P_CC_34
J4.004	AW2	IO_L1P_30	J4.064	BA20	IO_L10P_CC_34
J4.005	AW1	IO_L0N_30	J4.065	BB22	IO_L9N_CC_34
J4.006	AY2	IO_L1N_30	J4.066	AY20	IO_L10N_CC_34
J4.007	BA1	IO_L2P_30	J4.067	AY22	IO_L11P_34
J4.008	AY3	IO_L3P_30	J4.068	BB23	IO_L13P_34
J4.009	BA2	IO_L2N_30	J4.069	AY23	IO_L11N_34
J4.010	AW3	IO_L3N_30	J4.070	BA24	IO_L13N_34
J4.011	BB2	IO_L4P_30	J4.071	BB26	IO_L14P_34
J4.012	AY5	IO_L5P_30	J4.072	BB24	IO_L13P_34
J4.013	BB3	IO_L4N_VREF_30	J4.073	BA26	IO_L14N_VREF_34
J4.014	AW5	IO_L5N_30	J4.074	BA25	IO_L15N_34
J4.015	AY4	IO_L13P_30	J4.075	BA27	IO_L16P_34
J4.016	BA5	IO_L6P_30	J4.076	AY27	IO_L17P_34
J4.017	BA4	IO_L6N_30	J4.077	BB27	IO_L16N_34
J4.018	BB4	IO_L7N_30	J4.078	AY28	IO_L17N_34
J4.019	BB7	IO_L8P_CC_30	J4.079	BB29	IO_L18P_34
J4.020	AW7	IO_L9P_CC_30	J4.080	BA29	IO_L19P_34
J4.021	BA7	IO_L8N_CC_30	J4.081	BB28	IO_L18N_34
J4.022	AW6	IO_L9N_CC_30	J4.082	AY29	IO_L19N_34
J4.023	BB6	IO_L10P_CC_30	J4.083	AW15	IO_L19P_33
J4.024	AY7	IO_L11P_CC_30	J4.084	AU14	IO_L18P_33
J4.025	BA6	IO_L10N_CC_30	J4.085	AV15	IO_L19N_33
J4.026	AW8	IO_L11N_CC_30	J4.086	AV14	IO_L18N_33
J4.027	AY10	IO_L13P_30	J4.087	AW16	IO_L17P_33
J4.028	BB9	IO_L14P_30	J4.088	AV13	IO_L16P_33
J4.029	AW10	IO_L13N_30	J4.089	AV16	IO_L17N_33
J4.030	BA10	IO_L14N_VREF_30	J4.090	AW13	IO_L16N_33
J4.031		GND	J4.091		GND
J4.032		GND	J4.092		GND
J4.033	AY8	IO_L15P_30	J4.093	AW17	IO_L15P_33
J4.034	AW11	IO_L16P_30	J4.094	AU17	IO_L14P_33
J4.035	AY9	IO_L15N_30	J4.095	AV18	IO_L15N_33

J4.036	AW12	IO_L16N_30	J4.096	AU16	IO_L14N_VREF_33
J4.037	BA12	IO_L17P_30	J4.097	AW18	IO_L13N_33
J4.038	BB11	IO_L18P_30	J4.098	AW21	IO_L11P_CC_33
J4.039	BB12	IO_L17N_30	J4.099	AV19	IO_L13N_33
J4.040	BA11	IO_L18N_30	J4.100	AW20	IO_L11N_CC_33
J4.041	AY13	IO_L19P_30	J4.101	AV21	IO_L10P_CC_33
J4.042	BB13	IO_L0P_34	J4.102	AU23	IO_L9P_CC_33
J4.043	AY12	IO_L19N_30	J4.103	AV20	IO_L10N_CC_33
J4.044	BA14	IO_L0N_34	J4.104	AU22	IO_L9N_CC_33
J4.045	AY14	IO_L1P_34	J4.105	AW26	IO_L8P_CC_33
J4.046	BB16	IO_L2P_34	J4.106	AV24	IO_L7P_33
J4.047	AY15	IO_L1N_34	J4.107	AW25	IO_L8N_CC_33
J4.048	BA16	IO_L2N_34	J4.108	AV23	IO_L7N_33
J4.049	BA15	IO_L3P_34	J4.109	AW27	IO_L6P_33
J4.050	BB18	IO_L4P_34	J4.110	AW23	IO_L5P_33
J4.051	BB14	IO_L3N_34	J4.111	AV28	IO_L6N_33
J4.052	BB19	IO_L4N_VREF_34	J4.112	AW22	IO_L5N_33
J4.053	BB17	IO_L5P_34	J4.113	AU28	IO_L4P_33
J4.054	AY19	IO_L6P_34	J4.114	AV25	IO_L3P_33
J4.055	BA17	IO_L5N_34	J4.115	AU27	IO_L4N_REF_33
J4.056	BA19	IO_L6N_34	J4.116	AU24	IO_L3N_33
J4.057	AY17	IO_L7P_CC_34	J4.117	AV30	IO_L2P_33
J4.058	BA21	IO_L8P_CC_34	J4.118	AV26	IO_L1P_33
J4.059	AY18	IO_L7N_34	J4.119	AW30	IO_L2N_33
J4.060	BA22	IO_L8N_34	J4.120	AU26	IO_L1N_33

Pin table of FPGA dedicated IO (J5)

J5 (Dedicated IO)					
Top/Bottom Connector	FPGA IO Pin	IO Description	Top/Bottom Connector	FPGA IO Pin	IO Description
J5.001		VDD5V	J5.061		GND
J5.002		VDD5V	J5.062		GND
J5.003	AU29	IO_L13P_6	J5.063	AY40	IO_L2P_29
J5.004	AT30	IO_L11P_CC_6	J5.064	AY38	IO_L6P_29
J5.005	AT29	IO_L13N_6	J5.065	BA41	IO_L2N_29
J5.006	AR30	IO_L11N_CC_6	J5.066	AW37	IO_L6N_29
J5.007	AT14	IO_L14P_6	J5.067	AY37	IO_L9P_CC_29
J5.008	AR28	IO_L18P_CC_6	J5.068	AY35	IO_L11P_CC_29
J5.009	AT15	IO_L14N_VREF_6	J5.069	AW36	IO_L9N_CC_29
J5.010	AR29	IO_L13N_6	J5.070	AW35	IO_L11N_CC_29

J5.011	AR15	IO_L9P_CC_6	J5.071	AY33	IO_L14P_29
J5.012	AR13	IO_L10P_CC_6	J5.072	AY32	IO_L16P_29
J5.013	AP15	IO_L9N_CC_6	J5.073	AY34	IO_L14N_VREF_29
J5.014	AR14	IO_L10N_CC_6	J5.074	BA32	IO_L16N_29
J5.015	AP28	IO_L2P_6	J5.075	AY30	IO_L19P_29
J5.016	AP26	IO_L6P_6	J5.076	AW41	IO_L1P_29
J5.017	AP27	IO_L2N_6	J5.077	AW31	IO_L19N_29
J5.018	AR27	IO_L6N_6	J5.078	AW40	IO_L1N_29
J5.019	AN26	IO_L4P_6	J5.079	AW38	IO_L7P_29
J5.020	AN19	IO_L7P_6	J5.080	AW33	IO_L15P_29
J5.021	AN25	IO_L4N_VREF_6	J5.081	AY39	IO_L7N_29
J5.022	AP18	IO_L7N_6	J5.082	AW32	IO_L15N_29
J5.023	AN18	IO_L5P_6	J5.083	AU21	IO_L18P_8
J5.024	AM18	IO_L3P_6	J5.084	AT27	IO_L6P_8
J5.025	AP17	IO_L5N_6	J5.085	AT22	IO_L18N_8
J5.026	AM17	IO_L3N_6	J5.086	AT26	IO_L6N_8
J5.027	AL26	IO_L0P_6	J5.087	AT25	IO_L8P_CC_8
J5.028	AL25	IO_L15P_6	J5.088	AT20	IO_L16P_8
J5.029	AM26	IO_L0N_6	J5.089	AT24	IO_L8N_CC_8
J5.030	AM24	IO_L15N_6	J5.090	AT21	IO_L16N_8
J5.031		GND	J5.091		GND
J5.032		GND	J5.092		GND
J5.033	AL24	IO_L19P_6	J5.093	AR23	IO_L11P_CC_8
J5.034	AL19	IO_L18P_6	J5.094	AR22	IO_L19P_8
J5.035	AK25	IO_L19N_6	J5.095	AR24	IO_L11N_CC_8
J5.036	AM19	IO_L18N_6	J5.096	AP22	IO_L19N_8
J5.037	AK19	IO_L16P_6	J5.097	AR20	IO_L14P_8
J5.038	AK18	IO_L1P_6	J5.098	AR18	IO_L10P_CC_8
J5.039	AJ18	IO_L16N_6	J5.099	AT19	IO_L14N_VREF_8
J5.040	AL17	IO_L1N_6	J5.100	AR19	IO_L10N_CC_8
J5.041	AJ25	IO_L17P_6	J5.101	AP25	IO_L4P_8
J5.042	BB38	IO_L5P_29	J5.102	AP30	IO_L4P_GC_4
J5.043	AK24	IO_L19N_6	J5.103	AR25	IO_L4N_VREF_8
J5.044	BA39	IO_L5N_29	J5.104	AN29	IO_L4N_GC_VREF_4
J5.045	BB37	IO_L8P_CC_29	J5.105	AM28	IO_L6P_GC_4
J5.046	BB36	IO_L10P_CC_29	J5.106	AM29	IO_L2P_GC_D11_4
J5.047	BA37	IO_L8N_CC_29	J5.107	AN28	IO_L6N_GC_4
J5.048	BA36	IO_L10N_CC_29	J5.108	AN30	IO_L2N_GC_D10_4
J5.049	BB33	IO_L13P_29	J5.109	AL27	IO_L8P_CC_GC_4
J5.050	BB32	IO_L17P_29	J5.110	AK28	IO_L0P_GC_D15_4

J5.051	BA34	IO_L14N_VREF_29	J5.111	AM27	IO_L8N_CC_GC_4
J5.052	BB31	IO_L17N_29	J5.112	AK27	IO_L0N_GC_D14_4
J5.053	BA42	IO_L3P_29	J5.113	AP13	IO_L9P_CC_GC_4
J5.054	BA40	IO_L4P_29	J5.114	AM13	IO_L5P_GC_4
J5.055	BB41	IO_L3N_29	J5.115	AN13	IO_L9N_CC_GC_4
J5.056	BB39	IO_L4N_VREF_29	J5.116	AM14	IO_L5N_GC_4
J5.057	BA30	IO_L18P_29	J5.117	AN16	IO_L3N_GC_D9_4
J5.058	AY42	IO_L0P_29	J5.118	AL16	IO_L1N_GC_D13_4
J5.059	BA31	IO_L18N_29	J5.119	AM16	IO_L3N_GC_D8_4
J5.060	AW42	IO_L0N_29	J5.120	AK17	IO_L1N_GC_D12_4

Pin table of FPGA dedicated IO (J6)

J6 (Dedicated IO)					
Top/Bottom Connector	FPGA IO Pin	IO Description	Top/Bottom Connector	FPGA IO Pin	IO Description
J6.001		VDD5V	J6.061		GND
J6.002		VDD5V	J6.062		GND
J6.003	AV35	IO_L4P_25	J6.063	AJ35	IO_L17P_21
J6.004	AV33	IO_L2P_25	J6.064	AH36	IO_L16P_21
J6.005	AV36	IO_L4N_VREF_25	J6.065	AK35	IO_L17N_21
J6.006	AV34	IO_L9N_CC_25	J6.066	AJ36	IO_L16N_21
J6.007	AV31	IO_L10P_CC_25	J6.067	AH35	IO_L11P_CC_21
J6.008	AU36	IO_L5P_25	J6.068	AH34	IO_L10P_CC_21
J6.009	AU31	IO_L10N_CC_25	J6.069	AG36	IO_L11N_CC_21
J6.010	AT35	IO_L5N_25	J6.070	AG34	IO_L10N_CC_21
J6.011	AU34	IO_L6P_25	J6.071	AF35	IO_L9P_CC_21
J6.012	AU32	IO_L8P_CC_25	J6.072	AE35	IO_L8P_CC_21
J6.013	AT34	IO_L6N_25	J6.073	AF36	IO_L9N_CC_21
J6.014	AU33	IO_L8N_CC_25	J6.074	AF34	IO_L8N_CC_21
J6.015	AT32	IO_L11P_CC_25	J6.075	AE33	IO_L3P_21
J6.016	AR35	IO_L7P_25	J6.076	AD33	IO_L2P_21
J6.017	AT31	IO_L11N_CC_25	J6.077	AE34	IO_L3N_21
J6.018	AR34	IO_L7N_25	J6.078	AE32	IO_L2N_21
J6.019	AR33	IO_L14P_25	J6.079	AC33	IO_L1P_21
J6.020	AR32	IO_L13P_25	J6.080	AB33	IO_L0P_21
J6.021	AP33	IO_L14N_VREF_25	J6.081	AD32	IO_L1N_21
J6.022	AP32	IO_L13N_25	J6.082	AB32	IO_L0N_21
J6.023	AN33	IO_L15P_25	J6.083	AV40	IO_L10P_CC_17
J6.024	AL32	IO_L18P_25	J6.084	AT39	IO_L11P_CC_17

J6.025	AM33	IO_L15N_25	J6.085	AU39	IO_L10N_CC_17
J6.026	AM32	IO_L18N_25	J6.086	AR39	IO_L11N_CC_17
J6.027	AL31	IO_L19P_25	J6.087	AR40	IO_L9P_CC_17
J6.028	AK33	IO_L16P_25	J6.088	AN40	IO_L8P_CC_17
J6.029	AM31	IO_L19N_25	J6.089	AT40	IO_L9N_CC_17
J6.030	AJ33	IO_L16N_25	J6.090	AP40	IO_L8N_CC_17
J6.031		GND	J6.091		GND
J6.032		GND	J6.092		GND
J6.033	AJ32	IO_L17P_25	J6.093	AN39	IO_L17P_17
J6.034	AH33	IO_L2P_25	J6.094	AN38	IO_L18P_17
J6.035	AK32	IO_L17N_25	J6.095	AP38	IO_L17N_17
J6.036	AG32	IO_L2N_25	J6.096	AM38	IO_L18N_17
J6.037	AH31	IO_L3P_25	J6.097	AM37	IO_L19P_17
J6.038	AG31	IO_L0P_25	J6.098	AL39	IO_L16P_17
J6.039	AJ31	IO_L3N_25	J6.099	AL37	IO_L19N_17
J6.040	AF31	IO_L0N_25	J6.100	AM39	IO_L16N_17
J6.041	AF32	IO_L1P_25	J6.101	AK38	IO_L14P_17
J6.042	AV39	IO_L4P_21	J6.102	AJ38	IO_L13P_17
J6.043	AG33	IO_L1N_25	J6.103	AK37	IO_L14N_VREF_17
J6.044	AV38	IO_L4N_VREF_21	J6.104	AK39	IO_L13N_17
J6.045	AU38	IO_L5P_21	J6.105	AJ37	IO_L15P_17
J6.046	AT37	IO_L6P_21	J6.106	AG37	IO_L7P_17
J6.047	AU37	IO_L5N_21	J6.107	AH38	IO_L15N_17
J6.048	AR38	IO_L6N_21	J6.108	AF37	IO_L17N_17
J6.049	AR37	IO_L7P_21	J6.109	AF39	IO_L6P_17
J6.050	AP35	IO_L13P_21	J6.110	AE39	IO_L5P_17
J6.051	AT36	IO_L7N_21	J6.111	AG38	IO_L6N_17
J6.052	AN36	IO_L13N_21	J6.112	AE38	IO_L5N_17
J6.053	AN34	IO_L15P_21	J6.113	AE37	IO_L4P_17
J6.054	AM36	IO_L14P_21	J6.114	AD36	IO_L3P_17
J6.055	AM34	IO_L15N_21	J6.115	AD38	IO_L4N_VREF_17
J6.056	AN35	IO_L14N_VREF_21	J6.116	AD37	IO_L3N_17
J6.057	AL36	IO_L18P_21	J6.117	AC36	IO_L2N_17
J6.058	AL34	IO_L19P_21	J6.118	AC35	IO_L1N_17
J6.059	AL35	IO_L18N_21	J6.119	AD35	IO_L2N_17
J6.060	AK34	IO_L19N_21	J6.120	AB36	IO_L1N_17

Pin table of FPGA dedicated IO (J7)

J7 (Dedicated IO)

Top/Bottom Connector	FPGA IO Pin	IO Description	Top/Bottom Connector	FPGA IO Pin	IO Description
J7.001		VDD5V	J7.061		GND
J7.002		VDD5V	J7.062		GND
J7.003	AA35	IO_L16P_15	J7.063	N36	IO_L3P_19
J7.004	AA34	IO_L17P_15	J7.064	N35	IO_L1P_19
J7.005	AA36	IO_L16N_15	J7.065	P36	IO_L3N_19
J7.006	Y34	IO_L17N_15	J7.066	M36	IO_L1N_19
J7.007	Y35	IO_L18P_15	J7.067	L37	IO_L2P_19
J7.008	W36	IO_L19P_15	J7.068	L36	IO_L4P_19
J7.009	W35	IO_L8N_15	J7.069	M37	IO_L2N_19
J7.010	W37	IO_L19N_15	J7.070	L35	IO_L4N_VREF_19
J7.011	V39	IO_L15P_15	J7.071	K37	IO_L7P_19
J7.012	T39	IO_L14P_15	J7.072	K35	IO_L5P_19
J7.013	W38	IO_L15N_15	J7.073	J37	IO_L7N_19
J7.014	U39	IO_L14N_VREF_15	J7.074	J35	IO_L5N_19
J7.015	T37	IO_L13P_15	J7.075	H35	IO_L6P_19
J7.016	R39	IO_L4P_15	J7.076	F37	IO_L14P_19
J7.017	U38	IO_L13N_15	J7.077	J36	IO_L6N_19
J7.018	R38	IO_L4N_VREF_15	J7.078	E37	IO_L14N_VREF_19
J7.019	R37	IO_L5P_15	J7.079	F36	IO_L13P_19
J7.020	P38	IO_L6P_15	J7.080	E38	IO_L15P_19
J7.021	P37	IO_L5N_15	J7.081	G36	IO_L13N_19
J7.022	N38	IO_L6N_15	J7.082	D37	IO_L15N_19
J7.023	N39	IO_L7P_15	J7.083	U31	IO_L19P_23
J7.024	M38	IO_L8P_CC_15	J7.084	T32	IO_L18P_23
J7.025	M39	IO_L7N_15	J7.085	T31	IO_L19N_23
J7.026	L39	IO_L8N_CC_15	J7.086	U32	IO_L18N_23
J7.027	K40	IO_L11P_CC_15	J7.087	R33	IO_L17P_23
J7.028	K38	IO_L9P_CC_15	J7.088	P33	IO_L16P_23
J7.029	K39	IO_L11N_CC_15	J7.089	R32	IO_L17N_23
J7.030	J38	IO_L9N_CC_15	J7.090	P32	IO_L16N_23
J7.031		GND	J7.091		GND
J7.032		GND	J7.092		GND
J7.033	H40	IO_L10P_CC_15	J7.093	N33	IO_L0P_23
J7.034	H38	IO_L0P_15	J7.094	N31	IO_L3P_23
J7.035	J40	IO_L10N_CC_15	J7.095	N34	IO_L0N_23
J7.036	H39	IO_L0N_15	J7.096	P31	IO_L3N_23
J7.037	G38	IO_L1P_15	J7.097	M34	IO_L1P_23
J7.038	F39	IO_L2P_15	J7.098	M32	IO_L2P_23

J7.039	G39	IO_L1N_15	J7.099	M33	IO_L1N_23
J7.040	F40	IO_L2N_15	J7.100	M31	IO_L2N_23
J7.041	E39	IO_L3P_15	J7.101	L32	IO_L15P_23
J7.042	Y33	IO_L18P_19	J7.102	K33	IO_L13P_23
J7.043	E40	IO_L3N_15	J7.103	L31	IO_L15N_23
J7.044	W32	IO_L18N_19	J7.104	J33	IO_L13N_23
J7.045	Y32	IO_L19P_19	J7.105	K32	IO_L14P_23
J7.046	V35	IO_L16P_19	J7.106	H34	IO_L4P_23
J7.047	AA32	IO_L19N_19	J7.107	J32	IO_L14N_VREF_23
J7.048	V34	IO_L16N_19	J7.108	G34	IO_L4N_VREF_23
J7.049	V33	IO_L17P_19	J7.109	H31	IO_L7P_23
J7.050	U36	IO_L11P_CC_19	J7.110	G33	IO_L5P_23
J7.051	W33	IO_L17N_19	J7.111	J31	IO_L7N_23
J7.052	V36	IO_L11N_CC_19	J7.112	H33	IO_L5N_23
J7.053	U34	IO_L8P_CC_19	J7.113	G32	IO_L6P_23
J7.054	T34	IO_L9P_CC_19	J7.114	F35	IO_L8P_CC_23
J7.055	T35	IO_L8N_CC_19	J7.115	G31	IO_L6N_23
J7.056	U33	IO_L9N_CC_19	J7.116	E35	IO_L8N_CC_23
J7.057	R35	IO_L10P_CC_19	J7.117	F31	IO_L10P_CC_23
J7.058	R34	IO_L0P_19	J7.118	E34	IO_L9N_CC_23
J7.059	T36	IO_L10N_CC_19	J7.119	F32	IO_L10N_CC_23
J7.060	P35	IO_L0N_19	J7.120	F34	IO_L9N_CC_23

Pin table of FPGA dedicated IO (J8)

J8 (Dedicated IO)					
Top/Bottom Connector	FPGA IO Pin	IO Description	Top/Bottom Connector	FPGA IO Pin	IO Description
J8.001		VDD5V	J8.061		GND
J8.002		VDD5V	J8.062		GND
J8.003	R18	IO_L18P_5	J8.063	C36	IO_L9P_CC_27
J8.004	P25	IO_L8P_CC_5	J8.064	D35	IO_L8P_CC_27
J8.005	P18	IO_L18N_5	J8.065	C35	IO_L9N_CC_27
J8.006	N25	IO_L8N_CC_5	J8.066	D36	IO_L8N_CC_27
J8.007	N18	IO_L16P_5	J8.067	A34	IO_L7P_27
J8.008	M26	IO_L6P_5	J8.068	C34	IO_L6P_27
J8.009	N19	IO_L16N_5	J8.069	A35	IO_L7N_27
J8.010	L26	IO_L6N_5	J8.070	B34	IO_L6N_27
J8.011	M24	IO_L0P_5	J8.071	D32	IO_L5P_27
J8.012	M18	IO_L10P_CC_5	J8.072	B33	IO_L4P_27

J8.013	N24	IO_L0N_5	J8.073	D33	IO_L5N_27
J8.014	M19	IO_L10N_CC_5	J8.074	C33	IO_L4N_VREF_27
J8.015	M17	IO_L14P_5	J8.075	A32	IO_L3P_27
J8.016	L25	IO_L2P_5	J8.076	A30	IO_L2P_27
J8.017	L17	IO_L14N_VREF_5	J8.077	B32	IO_L3N_27
J8.018	L24	IO_L2N_5	J8.078	A31	IO_L2N_27
J8.019	K25	IO_L4P_5	J8.079	B31	IO_L1P_27
J8.020	K19	IO_L9P_CC_5	J8.080	D31	IO_L0P_27
J8.021	K24	IO_L4N_VRRF_5	J8.081	C30	IO_L1N_27
J8.022	L19	IO_L9N_CC_5	J8.082	C31	IO_L0N_27
J8.023	J28	IO_L13P_5	J8.083	P23	IO_L2P_7
J8.024	J26	IO_L11P_CC_5	J8.084	P21	IO_L3P_7
J8.025	K27	IO_L13N_5	J8.085	P22	IO_L2N_7
J8.026	J27	IO_L11N_CC_5	J8.086	N21	IO_L3N_7
J8.027	J17	IO_L5P_5	J8.087	P20	IO_L1P_7
J8.028	H29	IO_L19P_5	J8.088	N22	IO_L5P_7
J8.029	J16	IO_L5N_5	J8.089	N20	IO_L1N_7
J8.030	H30	IO_L19N_5	J8.090	M21	IO_L5N_7
J8.031		GND	J8.091		GND
J8.032		GND	J8.092		GND
J8.033	H16	IO_L3P_5	J8.093	M23	IO_L0P_7
J8.034	H15	IO_L7P_5	J8.094	L22	IO_L4P_7
J8.035	G16	IO_L3N_5	J8.095	N23	IO_L0N_7
J8.036	J15	IO_L7N_5	J8.096	M22	IO_L4N_VREF_7
J8.037	G28	IO_L17P_5	J8.097	L20	IO_L18P_7
J8.038	G27	IO_L15P_5	J8.098	K22	IO_L6P_7
J8.039	G29	IO_L17N_5	J8.099	L21	IO_L18N_7
J8.040	H28	IO_L15N_5	J8.100	K23	IO_L6N_7
J8.041	G14	IO_L1P_5	J8.101	K20	IO_L16P_7
J8.042	E42	IO_L19P_27	J8.102	J22	IO_L8P_CC_7
J8.043	H14	IO_L1N_5	J8.103	J20	IO_L16N_7
J8.044	D42	IO_L19N_27	J8.104	J23	IO_L8N_CC_7
J8.045	D40	IO_L18P_27	J8.105	J21	IO_L7P_7
J8.046	B42	IO_L17P_27	J8.106	J18	IO_L14P_7
J8.047	D41	IO_L18N_27	J8.107	H21	IO_L7N_7
J8.048	C41	IO_L17N_27	J8.108	H18	IO_L14N_VREF_7
J8.049	A41	IO_L16P_27	J8.109	G26	IO_L19P_7
J8.050	A39	IO_L15P_27	J8.110	H25	IO_L17P_7
J8.051	B41	IO_L16N_27	J8.111	H26	IO_L19N_7
J8.052	A40	IO_L15N_27	J8.112	J25	IO_L17N_7

J8.053	B39	IO_L14P_VREF_27	J8.113	H24	IO_L15P_7
J8.054	C39	IO_L13P_27	J8.114	H23	IO_L13P_7
J8.055	B38	IO_L14N_VREF_27	J8.115	G24	IO_L15N_7
J8.056	C40	IO_L13N_27	J8.116	G23	IO_L13N_7
J8.057	B37	IO_L11P_CC_27	J8.117	H19	IO_L10P_CC_7
J8.058	A37	IO_L10P_CC_27	J8.118	G22	IO_L11N_CC_7
J8.059	B36	IO_L11N_CC_27	J8.119	G19	IO_L10N_CC_7
J8.060	A36	IO_L10N_CC_27	J8.120	F22	IO_L11N_CC_7

FPGA Dedicated IO (Differential signals)

Pin table of FPGA Differential signals (J2)

J2 (Differential signals)					
Top/Bottom Connector	FPGA IO Pin	IO Description	Top/Bottom Connector	FPGA IO Pin	IO Description
J2.001		VDD5V	J2.061		GND
J2.002		VDD5V	J2.062		GND
J2.003	E2	IO_L0P_12	J2.063	G4	IO_L0P_16
J2.004	G2	IO_L1P_12	J2.064	E3	IO_L1P_16
J2.005	F2	IO_L0N_12	J2.065	F4	IO_L0N_16
J2.006	F1	IO_L1N_12	J2.066	E4	IO_L1N_16
J2.007		GND	J2.067		GND
J2.008		GND	J2.068		GND
J2.009	J2	IO_L2P_12	J2.069	H4	IO_L2P_16
J2.010	G1	IO_L3P_12	J2.070	H3	IO_L3P_16
J2.011	J1	IO_L2N_12	J2.071	J3	IO_L2N_16
J2.012	H1	IO_L3N_12	J2.072	G3	IO_L3N_16
J2.013		GND	J2.073		GND
J2.014		GND	J2.074		GND
J2.015	L2	IO_L4P_12	J2.075	L5	IO_L4P_16
J2.016	K2	IO_L5P_12	J2.076	K3	IO_L5P_16
J2.017	M2	IO_L4N_VREF_12	J2.077	L4	IO_L4N_VREF_16
J2.018	L1	IO_L5N_12	J2.078	K4	IO_L5N_16
J2.019		GND	J2.079		GND
J2.020		GND	J2.080		GND
J2.021	M1	IO_L6P_12	J2.081	N5	IO_L6P_16
J2.022	M3	IO_L7P_12	J2.082	N4	IO_L7P_16
J2.023	N1	IO_L6N_12	J2.083	P5	IO_L6N_16

J2.024	N3	IO_L7N_12	J2.084	M4	IO_L7N_16
J2.025		GND	J2.085		GND
J2.026		GND	J2.086		GND
J2.027	P3	IO_L8P_CC_12	J2.087	T6	IO_L8P_CC_16
J2.028	P2	IO_L9P_CC_12	J2.088	R5	IO_L9P_CC_16
J2.029	R3	IO_L8N_CC_12	J2.089	T5	IO_L8N_CC_16
J2.030	P1	IO_L9N_CC_12	J2.090	R4	IO_L9N_CC_16
J2.031		GND	J2.091		GND
J2.032		GND	J2.092		GND
J2.033	R2	IO_L10P_CC_12	J2.093	U4	IO_L10P_CC_16
J2.034	U1	IO_L11P_CC_12	J2.094	U6	IO_L11P_CC_16
J2.035	T2	IO_L10N_CC_12	J2.095	T4	IO_L10N_CC_16
J2.036	T1	IO_L11N_CC_12	J2.096	V5	IO_L11N_CC_16
J2.037		GND	J2.097		GND
J2.038		GND	J2.098		GND
J2.039	U3	IO_L12P_VRN_12	J2.099	W7	IO_L12P_VRN_16
J2.040	V3	IO_L13P_12	J2.100	W5	IO_L13P_16
J2.041	U2	IO_L12N_VRN_12	J2.101	W8	IO_L12N_VRN_16
J2.042	V4	IO_L13N_12	J2.102	V6	IO_L13N_16
J2.043		GND	J2.103		GND
J2.044		GND	J2.104		GND
J2.045	W2	IO_L14P_12	J2.105	Y7	IO_L14P_16
J2.046	W1	IO_L15P_12	J2.106	AA7	IO_L15P_16
J2.047	W3	IO_L14N_VREF_12	J2.107	W6	IO_L14N_VREF_16
J2.048	V1	IO_L15N_12	J2.108	AA6	IO_L15N_16
J2.049		GND	J2.109		GND
J2.050		GND	J2.110		GND
J2.051	AA4	IO_L16P_12	J2.111	Y8	IO_L16P_16
J2.052	Y5	IO_L17P_12	J2.112	Y1	IO_L17P_16
J2.053	Y4	IO_L16N_12	J2.113	AA9	IO_L16N_16
J2.054	AA5	IO_L17N_12	J2.114	Y9	IO_L17N_16
J2.055		GND	J2.115		GND
J2.056		GND	J2.116		GND
J2.057	AA1	IO_L18P_12	J2.117	W11	IO_L18P_16
J2.058	Y2	IO_L19P_12	J2.118	AA11	IO_L19P_16
J2.059	AA2	IO_L18N_12	J2.119	W10	IO_L18N_16
J2.060	Y3	IO_L19N_12	J2.120	AA10	IO_L19N_16

Pin table of FPGA Differential signals (J3)

J3 (Differential signals)

Top/Bottom Connector	FPGA IO Pin	IO Description	Top/Bottom Connector	FPGA IO Pin	IO Description
J3.001		VDD5V	J3.061		GND
J3.002		VDD5V	J3.062		GND
J3.003	AB6	IO_L0P_14	J3.063	AB11	IO_L0P_18
J3.004	AC4	IO_L1P_14	J3.064	AD11	IO_L1P_18
J3.005	AC6	IO_L0N_14	J3.065	AC11	IO_L0N_18
J3.006	AC5	IO_L1N_14	J3.066	AD10	IO_L1N_18
J3.007		GND	J3.067		GND
J3.008		GND	J3.068		GND
J3.009	AC3	IO_L2P_14	J3.069	AC10	IO_L2P_18
J3.010	AB3	IO_L3P_14	J3.070	AB8	IO_L3P_18
J3.011	AD3	IO_L2N_14	J3.071	AC9	IO_L2N_18
J3.012	AB4	IO_L3N_14	J3.072	AB9	IO_L3N_18
J3.013		GND	J3.073		GND
J3.014		GND	J3.074		GND
J3.015	AD2	IO_L4P_14	J3.075	AC8	IO_L4P_18
J3.016	AB2	IO_L5P_14	J3.076	AD5	IO_L5P_18
J3.017	AE3	IO_L4N_VREF_14	J3.077	AB7	IO_L4N_VREF_18
J3.018	AB1	IO_L5N_14	J3.078	AE5	IO_L5N_18
J3.019		GND	J3.079		GND
J3.020		GND	J3.080		GND
J3.021	AC1	IO_L6P_14	J3.081	AD6	IO_L6P_18
J3.022	AE2	IO_L7P_14	J3.082	AD7	IO_L7P_18
J3.023	AD1	IO_L6N_14	J3.083	AE7	IO_L6N_18
J3.024	AF1	IO_L7N_14	J3.084	AD8	IO_L7N_18
J3.025		GND	J3.085		GND
J3.026		GND	J3.086		GND
J3.027	AF2	IO_L8P_CC_14	J3.087	AE4	IO_L8P_CC_18
J3.028	AG3	IO_L9P_CC_14	J3.088	AF6	IO_L9P_CC_18
J3.029	AG2	IO_L8N_CC_14	J3.089	AF4	IO_L8N_CC_18
J3.030	AH3	IO_L9N_CC_14	J3.090	AF5	IO_L9N_CC_18
J3.031		GND	J3.091		GND
J3.032		GND	J3.092		GND
J3.033	AJ2	IO_L10P_CC_14	J3.093	AG6	IO_L10P_CC_18
J3.034	AH1	IO_L11P_CC_14	J3.094	AH4	IO_L11P_CC_18
J3.035	AJ3	IO_L10N_CC_14	J3.095	AH6	IO_L10N_CC_18
J3.036	AG1	IO_L11N_CC_14	J3.096	AG4	IO_L11N_CC_18
J3.037		GND	J3.097		GND

J3.038		GND	J3.098		GND
J3.039	AL2	IO_L12P_VRN_14	J3.099	AJ5	IO_L12P_VRN_18
J3.040	AK2	IO_L13P_14	J3.100	AJ6	IO_L13P_18
J3.041	AK3	IO_L12N_VRN_14	J3.101	AK4	IO_L12N_VRN_18
J3.042	AJ1	IO_L13N_14	J3.102	AH5	IO_L13N_18
J3.043		GND	J3.103		GND
J3.044		GND	J3.104		GND
J3.045	AM3	IO_L14P_14	J3.105	AL6	IO_L14P_18
J3.046	AM1	IO_L15P_14	J3.106	AK5	IO_L15P_18
J3.047	AM2	IO_L14N_VREF_14	J3.107	AL5	IO_L14N_VREF_18
J3.048	AL1	IO_L15N_14	J3.108	AL4	IO_L15N_18
J3.049		GND	J3.109		GND
J3.050		GND	J3.110		GND
J3.051	AR2	IO_L16P_14	J3.111	AP3	IO_L16P_18
J3.052	AP1	IO_L17P_14	J3.112	AN4	IO_L17P_18
J3.053	AP2	IO_L16N_14	J3.113	AN3	IO_L16N_18
J3.054	AN1	IO_L17N_14	J3.114	AM4	IO_L17N_18
J3.055		GND	J3.115		GND
J3.056		GND	J3.116		GND
J3.057	AT1	IO_L18P_14	J3.117	AR3	IO_L18P_18
J3.058	AU2	IO_L19P_14	J3.118	AM6	IO_L19P_18
J3.059	AT2	IO_L18N_14	J3.119	AR4	IO_L18N_18
J3.060	AU1	IO_L19N_14	J3.120	AN5	IO_L19N_18

Clock

Pin Table of the Half-size Oscillators

Clock Oscillator (Half Size)	
FPGA IO Pin	Connector
J30	J14

Pin Table of Configurable Clocks

Clock Generator	
FPGA IO Pin	Location
L29	Y1

Pin Table of SMA connector

SMA Connector	
FPGA IO Pin	Connector
K28	J15

Pin Table of 48MHz

48 MHz Clock Source
FPGA IO Pin
K13

User-defined LEDs

Pin Table of User-defined LEDs

User-defined LEDs	
Location	FPGA IO Pins
DS0	R28
DS1	P27
DS2	R27
DS3	P26
DS4	P17
DS5	R17
DS6	P16
DS7	R15

VeriEnterprise Xilinx Spartan-6 USB

Pin Tables

In this chapter, the FPGA pin, which connects to other devices such as I/O connectors, clocks etc, will be shown in the following tables.

XC6SLX150FGG676

	Pin Name	Comment
clk	B14	User circuit clock pin connect to this pin

input0	A11	Circuit input signal
input1	H12	
input2	G11	
input3	B12	
input4	A12	
input5	F12	
input6	E12	
input7	D12	
input8	C12	
input9	G13	
input10	F14	
input11	F13	
input12	D13	
input13	B16	
input14	A16	
input15	J14	
input16	G14	
input17	E14	
input18	D15	
input19	J13	
input20	K14	
input21	D16	
input22	C16	
input23	G16	
input24	F15	
input25	F17	
input26	E17	
input27	F16	
input28	A13	
input29	J16	
input30	A14	
input31	G17	
input32	C14	
input33	C20	
input34	E19	
input35	C17	
input36	A17	
input37	J15	
input38	H15	
input39	D18	

input40	C18	
input41	K15	
input42	AF4	
input43	AF5	
input44	AE5	
input45	AF6	
input46	AD6	
input47	AF7	
input48	AE7	
input49	AB9	
input50	AA9	
input51	AD9	
input52	AC9	
input53	AF8	
input54	AD8	
input55	W11	
Input56	V11	
Input57	AB10	
Input58	Y10	
Input59	U12	
Input60	U13	
Input61	AF10	
Input62	AD10	
Input63	AF9	
Output0	Y16	Circuit output signal
Output1	AF18	
Output2	AD18	
Output3	W17	
Output4	V16	
Output5	AD19	
Output6	AC19	
Output7	AA19	
Output8	Y18	
Output9	AD21	
Output10	AC20	
Output11	AF19	
Output12	AE19	
Output13	AF20	

Output14	AF21	
Output15	AE21	
Output16	A22	
Output17	Y14	
Output18	AA15	
Output19	Y15	
Output20	AC14	
Output21	AB15	
Output22	A2	
Output23	B4	
Output24	A4	
Output25	E6	
Output26	D5	
Output27	C5	
Output28	A5	
Output29	G8	
Output30	F7	
Output31	B6	
Output32	A6	
Output33	G9	
Output34	F8	
Output35	D6	
Output36	C6	
Output37	K12	
Output38	J11	
Output39	H10	
Output40	H9	
Output41	C7	
Output42	A7	
Output43	E8	
Output44	D7	
Output45	D8	
Output46	C8	
Output47	F9	
Output48	E9	
Output49	B8	
Output50	A8	
Output51	C9	
Output52	A9	
Output53	E10	

Output54	F10	
Output55	D10	
Output56	C10	
Output57	J12	
Output58	H13	
Output59	B10	
Output60	A10	
Output61	D11	
Output62	F11	
Output63	C11	

FPGA Dedicated IO

Pin table of FPGA U1 dedicated IO (J6)

Connector	FPGA IO Pin	Connector	FPGA IO Pin
J6.001	AC7	J6.061	W2
J6.002	AE3	J6.062	V3
J6.003	AD7	J6.063	W1
J6.004	AF2	J6.064	V1
J6.005	AC4	J6.065	U2
J6.006	AA8	J6.066	T3
J6.007	AD4	J6.067	U1
J6.008	AB8	J6.068	T1
J6.009	AA7	J6.069	V4
J6.010	AB7	J6.070	N8
J6.011	Y6	J6.071	W3
J6.012	AB6	J6.072	P8
J6.013	Y9	J6.073	R2
J6.014	AC5	J6.074	P7
J6.015	Y8	J6.075	R1
J6.016	AD5	J6.076	P6
J6.017	W8	J6.077	R4
J6.018	AA5	J6.078	N7
J6.019	W7	J6.079	R3
J6.020	AB5	J6.080	N6
J6.021	V7	J6.081	P3

J6.022	AB4	J6.082	P10
J6.023	AC3	J6.083	P1
J6.024	AC3	J6.084	R9
J6.025	W9	J6.085	P5
J6.026	AA4	J6.086	M10
J6.027	V8	J6.087	N5
J6.028	AA3	J6.088	N9
J6.029	W10	J6.089	N4
J6.030	W5	J6.090	M9
J6.031	V10	J6.091	N3
J6.032	Y5	J6.092	M8
J6.033	U8	J6.093	L4
J6.034	U5	J6.094	M6
J6.035	U7	J6.095	L3
J6.036	V5	J6.096	M4
J6.037	T10	J6.097	L7
J6.038	R10	J6.098	K8
J6.039	U9	J6.099	L6
J6.040	R10	J6.100	L8
J6.041	U4	J6.101	N2
J6.042	T8	J6.102	M3
J6.043	U3	J6.103	N1
J6.044	T4	J6.104	M1
J6.045	R5	J6.105	L2
J6.046	R7	J6.106	K3
J6.047	T4	J6.107	L1
J6.048	R6	J6.108	K1
J6.049	AB3	J6.109	J2
J6.050	AD3	J6.110	H3
J6.051	AB1	J6.111	J1
J6.052	AD1	J6.112	H1
J6.053	AC2	J6.113	G2
J6.054	AE2	J6.114	F3
J6.055	AC1	J6.115	G1
J6.056	AE1	J6.116	F1
J6.057	AA2	J6.117	E2
J6.058	Y3	J6.118	D3
J6.059	AA1	J6.119	E1
J6.060	Y1	J6.120	D1

SW2		
SW2.3	SW2.4	J8 VDDADJ
OFF	OFF	3.3V
ON	OFF	2.5V
OFF	ON	1.8V

Pin table of FPGA dedicated IO (J8)

Top/Bottom Connector	FPGA IO Pin	Top/Bottom Connector	FPGA IO Pin
J8.01	B23	J8.02	B24
J8.03	A23	J8.04	A25
J8.05	B25	J8.06	D23
J8.07	B26	J8.08	C24
J8.09	C25	J8.10	D24
J8.11	VDD5V	J8.12	GND
J8.13	D24	J8.14	E23
J8.15	D26	J8.16	E24
J8.17	E25	J8.18	F24
J8.19	E26	J8.20	F26
J8.21	G23	J8.22	G25
J8.23	G24	J8.24	G26
J8.25	H24	J8.26	J23
J8.27	H26	J8.28	J24
J8.29	VDDADJ	J8.30	GND
J8.31	J25	J8.32	J26
J8.33	K24	J8.34	L25
J8.35	K26	J8.36	L26
J8.37	M24	J8.38	N25
J8.39	M26	J8.40	N26

FPGA Dedicated IO (Differential signals)

Pin table of FPGA Differential signals (J4)

Connector	FPGA IO Pin	Connector	FPGA IO Pin
J4.001	M19	J4.061	GND
J4.002	R25	J4.062	GND
J4.003	L18	J4.063	R22
J4.004	R26	J4.064	P17
J4.005	GND	J4.065	R21
J4.006	GND	J4.066	P18
J4.007	M18	J4.067	GND
J4.008	N22	J4.068	GND
J4.009	N19	J4.069	R20
J4.010	N23	J4.070	R17
J4.011	GND	J4.071	R19
J4.012	GND	J4.072	R18
J4.013	N17	J4.073	GND
J4.014	R23	J4.074	GND
J4.015	N18	J4.075	T22
J4.016	R24	J4.076	T18
J4.017	GND	J4.077	U23
J4.018	GND	J4.078	T19
J4.019	N20	J4.079	GND
J4.020	P21	J4.080	GND
J4.021	M21	J4.081	U21
J4.022	P22	J4.082	U17
J4.023	GND	J4.083	U22
J4.024	GND	J4.084	V17
J4.025	V23	J4.085	GND
J4.026	U25	J4.086	GND
J4.027	W24	J4.087	AA23
J4.028	U26	J4.088	T20
J4.029	GND	J4.089	AA24
J4.030	GND	J4.090	U20
J4.031	W25	J4.091	GND
J4.032	V24	J4.092	GND
J4.033	W26	J4.093	AC23
J4.034	V26	J4.094	V18
J4.035	GND	J4.095	AC24

J4.036	GND	J4.096	V19
J4.037	T24	J4.097	GND
J4.038	Y24	J4.098	GND
J4.039	T26	J4.099	AE24
J4.040	Y24	J4.100	W18
J4.041	GND	J4.101	AF25
J4.042	GND	J4.102	W19
J4.043	AD24	J4.103	GND
J4.044	AB24	J4.104	GND
J4.045	AD26	J4.105	AB21
J4.046	AB26	J4.106	U19
J4.047	GND	J4.107	AB22
J4.048	GND	J4.108	V20
J4.049	AC25	J4.109	GND
J4.050	AA25	J4.110	GND
J4.051	AC26	J4.111	V22
J4.052	AA26	J4.112	Y20
J4.053	GND	J4.113	W22
J4.054	GND	J4.114	Y21
J4.055	AE25	J4.115	GND
J4.056	T23	J4.116	GND
J4.057	AE26	J4.117	Y22
J4.058	U24	J4.118	AE23
J4.059	GND	J4.119	AA22
J4.060	GND	J4.120	AF24

Clock

Pin Table of the Half-size Oscillators

Clock Oscillator (Half Size)	
FPGA IO Pin	Connector
AD14	J14

Pin Table of Configurable Clocks

Clock Generator	
FPGA IO Pin	Location
AE13	Y1

Pin Table of 48MHz

48 MHz Clock Source
FPGA IO Pin
C15

User-defined LEDs

Pin Table of User-defined LEDs

User-defined LEDs	
Location	FPGA IO Pins
DS0	D20
DS1	J17
DS2	H17
DS3	B18
DS4	A18
DS5	C19
DS6	AD16
DS7	AF22

User-defined Switch

Pin Table of User-defined switch

User-defined Switch	
Location	FPGA IO Pins
SW6.1	A19
SW6.2	B20
SW6.3	A20
SW6.4	C21

User-defined Button

Pin Table of User-defined Button

User-defined Button	
Location	FPGA IO Pins
BT1	A21

BT2	B22
-----	-----

UART

UART 1 J10		
Location	Signal	FPGA IO Pins
J10.1	TX	C3
J10.3	RX	C4
J10.5	GND	
J10.7	3.3V	

UART 2 J10		
Location	Signal	FPGA IO Pins
J10.2	3.3V	
J10.4	GND	
J10.6	RX	B2
J10.8	TX	B1

Soc-S150 Xilinx Spartan-6 USB

Pin Tables

In this chapter, the FPGA pin, which connects to other devices such as I/O connectors, clocks etc, will be shown in the following tables.

XC6SLX150FGG676

	Pin Name	Comment
clk	B14	User circuit clock pin connect to this pin
input0	Y18	Circuit input signal
input1	AA19	
input2	AF21	
input3	AC19	
input4	AD19	
input5	AF20	
input6	AE19	

input7	AD21	
input8	A22	
input9	AE21	
input10	A17	
input11	C14	
input12	K15	
input13	E20	
input14	E18	
input15	C11	
input16	A11	
input17	A10	
input18	D11	
input19	B8	
input20	A8	
input21	B10	
input22	F11	
input23	C10	
input24	J12	
input25	D10	
input26	H13	
input27	A9	
input28	C9	
input29	F10	
input30	E10	
input31	C8	
input32	F9	
input33	E9	
input34	D7	
input35	A7	
input36	E8	
input37	H10	
input38	H9	
input39	C7	
input40	D8	
input41	F8	
input42	G9	
input43	C6	
input44	D6	
input45	J11	
input46	K12	

input47	E6	
input48	J16	
input49	C5	
input50	D5	
input51	B6	
input52	A6	
input53	A4	
input54	B4	
input55	AE11	
Input56	A2	
Input57	A5	
Input58	G8	
Input59	V12	
Input60	W12	
Input61	AA11	
Input62	AC11	
Input63	AE9	
Output0	AC14	Circuit output signal
Output1	AC17	
Output2	AB15	
Output3	AE15	
Output4	AB17	
Output5	AD15	
Output6	AD17	
Output7	V15	
Output8	W16	
Output9	AF16	
Output10	AC15	
Output11	AC16	
Output12	AF15	
Output13	AF17	
Output14	AE5	
Output15	AF6	
Output16	AD6	
Output17	AF7	
Output18	AF4	
Output19	AF5	
Output20	AE7	

Output21	AB9	
Output22	AD9	
Output23	AA9	
Output24	AF8	
Output25	AC9	
Output26	W11	
Output27	AD8	
Output28	V11	
Output29	AB10	
Output30	U13	
Output31	AF10	
Output32	U12	
Output33	Y10	
Output34	AF9	
Output35	AD10	
Output36	K14	
Output37	F17	
Output38	C16	
Output39	F16	
Output40	E14	
Output41	J14	
Output42	E17	
Output43	G16	
Output44	B16	
Output45	J13	
Output46	D15	
Output47	G14	
Output48	F15	
Output49	D16	
Output50	A16	
Output51	D13	
Output52	E16	
Output53	AD18	
Output54	AA17	
Output55	AA18	
Output56	AB18	
Output57	AE17	
Output58	AC20	
Output59	AF19	
Output60	AF18	

Output61	Y16	
Output62	V16	
Output63	W17	

FPGA Dedicated IO

Pin table of FPGA U1 dedicated IO (J6)

Connector	FPGA IO Pin	Connector	FPGA IO Pin
J6.001	AC7	J6.061	W2
J6.002	AE3	J6.062	V3
J6.003	AD7	J6.063	W1
J6.004	AF2	J6.064	V1
J6.005	AC4	J6.065	U2
J6.006	AA8	J6.066	T3
J6.007	AD4	J6.067	U1
J6.008	AB8	J6.068	T1
J6.009	AA7	J6.069	V4
J6.010	AB7	J6.070	N8
J6.011	Y6	J6.071	W3
J6.012	AB6	J6.072	P8
J6.013	Y9	J6.073	R2
J6.014	AC5	J6.074	P7
J6.015	Y8	J6.075	R1
J6.016	AD5	J6.076	P6
J6.017	W8	J6.077	R4
J6.018	AA5	J6.078	N7
J6.019	W7	J6.079	R3
J6.020	AB5	J6.080	N6
J6.021	V7	J6.081	P3
J6.022	AB4	J6.082	P10
J6.023	AC3	J6.083	P1
J6.024	AC3	J6.084	R9
J6.025	W9	J6.085	P5
J6.026	AA4	J6.086	M10
J6.027	V8	J6.087	N5
J6.028	AA3	J6.088	N9

J6.029	W10	J6.089	N4
J6.030	W5	J6.090	M9
J6.031	V10	J6.091	N3
J6.032	Y5	J6.092	M8
J6.033	U8	J6.093	L4
J6.034	U5	J6.094	M6
J6.035	U7	J6.095	L3
J6.036	V5	J6.096	M4
J6.037	T10	J6.097	L7
J6.038	R10	J6.098	K8
J6.039	U9	J6.099	L6
J6.040	R10	J6.100	L8
J6.041	U4	J6.101	N2
J6.042	T8	J6.102	M3
J6.043	U3	J6.103	N1
J6.044	T4	J6.104	M1
J6.045	R5	J6.105	L2
J6.046	R7	J6.106	K3
J6.047	T4	J6.107	L1
J6.048	R6	J6.108	K1
J6.049	AB3	J6.109	J2
J6.050	AD3	J6.110	H3
J6.051	AB1	J6.111	J1
J6.052	AD1	J6.112	H1
J6.053	AC2	J6.113	G2
J6.054	AE2	J6.114	F3
J6.055	AC1	J6.115	G1
J6.056	AE1	J6.116	F1
J6.057	AA2	J6.117	E2
J6.058	Y3	J6.118	D3
J6.059	AA1	J6.119	E1
J6.060	Y1	J6.120	D1

SW2		
SW2.3	SW2.4	J8 VDDADJ
OFF	OFF	3.3V
ON	OFF	2.5V
OFF	ON	1.8V

Pin table of FPGA dedicated IO (J8)

Top/Bottom Connector	FPGA IO Pin	Top/Bottom Connector	FPGA IO Pin
J8.01	B23	J8.02	B24
J8.03	A23	J8.04	A25
J8.05	B25	J8.06	D23
J8.07	B26	J8.08	C24
J8.09	C25	J8.10	D24
J8.11	VDD5V	J8.12	GND
J8.13	D24	J8.14	E23
J8.15	D26	J8.16	E24
J8.17	E25	J8.18	F24
J8.19	E26	J8.20	F26
J8.21	G23	J8.22	G25
J8.23	G24	J8.24	G26
J8.25	H24	J8.26	J23
J8.27	H26	J8.28	J24
J8.29	VDDADJ	J8.30	GND
J8.31	J25	J8.32	J26
J8.33	K24	J8.34	L25
J8.35	K26	J8.36	L26
J8.37	M24	J8.38	N25
J8.39	M26	J8.40	N26

FPGA Dedicated IO (Differential signals)

Pin table of FPGA Differential signals (J4)

Connector	FPGA IO Pin	Connector	FPGA IO Pin
J4.001	M19	J4.061	GND
J4.002	R25	J4.062	GND
J4.003	L18	J4.063	R22

J4.004	R26	J4.064	P17
J4.005	GND	J4.065	R21
J4.006	GND	J4.066	P18
J4.007	M18	J4.067	GND
J4.008	N22	J4.068	GND
J4.009	N19	J4.069	R20
J4.010	N23	J4.070	R17
J4.011	GND	J4.071	R19
J4.012	GND	J4.072	R18
J4.013	N17	J4.073	GND
J4.014	R23	J4.074	GND
J4.015	N18	J4.075	T22
J4.016	R24	J4.076	T18
J4.017	GND	J4.077	U23
J4.018	GND	J4.078	T19
J4.019	N20	J4.079	GND
J4.020	P21	J4.080	GND
J4.021	M21	J4.081	U21
J4.022	P22	J4.082	U17
J4.023	GND	J4.083	U22
J4.024	GND	J4.084	V17
J4.025	V23	J4.085	GND
J4.026	U25	J4.086	GND
J4.027	W24	J4.087	AA23
J4.028	U26	J4.088	T20
J4.029	GND	J4.089	AA24
J4.030	GND	J4.090	U20
J4.031	W25	J4.091	GND
J4.032	V24	J4.092	GND
J4.033	W26	J4.093	AC23
J4.034	V26	J4.094	V18
J4.035	GND	J4.095	AC24
J4.036	GND	J4.096	V19
J4.037	T24	J4.097	GND
J4.038	Y24	J4.098	GND
J4.039	T26	J4.099	AE24
J4.040	Y24	J4.100	W18
J4.041	GND	J4.101	AF25
J4.042	GND	J4.102	W19
J4.043	AD24	J4.103	GND

J4.044	AB24	J4.104	GND
J4.045	AD26	J4.105	AB21
J4.046	AB26	J4.106	U19
J4.047	GND	J4.107	AB22
J4.048	GND	J4.108	V20
J4.049	AC25	J4.109	GND
J4.050	AA25	J4.110	GND
J4.051	AC26	J4.111	V22
J4.052	AA26	J4.112	Y20
J4.053	GND	J4.113	W22
J4.054	GND	J4.114	Y21
J4.055	AE25	J4.115	GND
J4.056	T23	J4.116	GND
J4.057	AE26	J4.117	Y22
J4.058	U24	J4.118	AE23
J4.059	GND	J4.119	AA22
J4.060	GND	J4.120	AF24

Clock

Pin Table of the Half-size Oscillators

Clock Oscillator (Half Size)	
FPGA IO Pin	Connector
AD14	J14

Pin Table of Configurable Clocks

Clock Generator	
FPGA IO Pin	Location
AE13	Y1

Pin Table of 48MHz

48 MHz Clock Source
FPGA IO Pin
C15

User-defined LEDs

Pin Table of User-defined LEDs

User-defined LEDs	
Location	FPGA IO Pins
DS0	D20
DS1	J17
DS2	H17
DS3	B18
DS4	A18
DS5	C19
DS6	AD16
DS7	AF22

User-defined Switch

Pin Table of User-defined switch

User-defined Switch	
Location	FPGA IO Pins
SW6.1	A19
SW6.2	B20
SW6.3	A20
SW6.4	C21

User-defined Button

Pin Table of User-defined Button

User-defined Button	
Location	FPGA IO Pins
BT1	A21
BT2	B22

UART

UART 1 J10		
Location	Signal	FPGA IO Pins
J10.1	TX	C3
J10.3	RX	C4
J10.5	GND	
J10.7	3.3V	
UART 2 J10		
Location	Signal	FPGA IO Pins
J10.2	3.3V	
J10.4	GND	
J10.6	RX	B2
J10.8	TX	B1

SoC-V330 Xilinx Virtex-5 USB

Pin Tables

In this chapter, the FPGA pin, which connects to other devices such as I/O connectors, clocks etc, will be shown in the following tables.

XC5VLX330FF1760

	Pin Name	Comment
clk	K15	User circuit clock pin connect to this pin
input0	AJ13	Circuit input signal
input1	AK12	
input2	AH16	
input3	AJ12	
input4	AJ11	
input5	AL14	
input6	AL15	
input7	AK13	
input8	AJ16	
input9	AJ15	

input10	M13	
input11	P10	
input12	P15	
input13	M16	
input14	N16	
input15	G8	
input16	E7	
input17	V11	
input18	V10	
input19	T9	
input20	R7	
input21	V8	
input22	F7	
input23	U11	
input24	T11	
input25	U9	
input26	V9	
input27	U7	
input28	R8	
input29	U8	
input30	T7	
input31	T10	
input32	R10	
input33	R9	
input34	N6	
input35	P7	
input36	N8	
input37	N9	
input38	M8	
input39	M9	
input40	P6	
input41	M7	
input42	J6	
input43	L7	
input44	M6	
input45	P8	
input46	L6	
input47	E5	
input48	M11	
input49	H6	
input50	G7	

input51	K7	
input52	K5	
input53	F5	
input54	F6	
input55	AH10	
Input56	G6	
Input57	J5	
Input58	H5	
Input59	AH9	
Input60	AG9	
Input61	AG7	
Input62	AJ7	
Input63	AG12	
Output0	AV9	Circuit output signal
Output1	AR10	
Output2	AU13	
Output3	AP11	
Output4	AR12	
Output5	AT11	
Output6	AU11	
Output7	AV11	
Output8	AU12	
Output9	AP12	
Output10	AR9	
Output11	AT10	
Output12	AT12	
Output13	AN9	
Output14	AT5	
Output15	AT6	
Output16	AU4	
Output17	AT4	
Output18	AV3	
Output19	AU3	
Output20	AV5	
Output21	AV4	
Output22	AP6	
Output23	AP7	
Output24	AP5	
Output25	AR5	
Output26	AM8	

Output27	AN8	
Output28	AM7	
Output29	AN6	
Output30	AK8	
Output31	AL7	
Output32	AK9	
Output33	AL9	
Output34	AG11	
Output35	AH11	
Output36	H13	
Output37	J12	
Output38	J10	
Output39	L10	
Output40	E13	
Output41	G11	
Output42	L11	
Output43	J11	
Output44	F11	
Output45	G12	
Output46	H11	
Output47	E12	
Output48	K12	
Output49	G13	
Output50	F12	
Output51	F10	
Output52	L9	
Output53	AK10	
Output54	AN11	
Output55	AN10	
Output56	AM9	
Output57	AP8	
Output58	AK14	
Output59	AK15	
Output60	AL12	
Output61	AL11	
Output62	AJ10	
Output63	AL10	

Clock

Pin Table of the Half-size Oscillators

Clock Oscillator (Half Size)	
FPGA IO Pin	Connector
J30	J14

← 格式化表格

Pin Table of Configurable Clocks

Clock Generator	
FPGA IO Pin	Location
L29	Y1

← 格式化表格

Pin Table of SMA connector

SMA Connector	
FPGA IO Pin	Connector
K28	J15

← 格式化表格

Pin Table of 48MHz

48 MHz Clock Source
FPGA IO Pin
K13

User-defined LEDs

Pin Table of User-defined LEDs

User-defined LEDs	
Location	FPGA IO Pins
DS0	R28
DS1	P27
DS2	R27
DS3	P26
DS4	P17
DS5	R17
DS6	P16
DS7	R15

Macube-VEXT2 (T3) Xilinx Spartan-6 USB

Pin Tables

In this chapter, the FPGA pin, which connects to other devices such as I/O connectors, clocks etc, will be shown in the following tables.

XC6SLX75TFGG676

	Pin Name	Comment
clk	AE15	User circuit clock pin connect to this pin
input0	L10	Circuit input signal
input1	K10	
input2	M3	
input3	M1	
input4	M8	
input5	M6	
input6	M10	
input7	M9	
input8	P3	
input9	P1	
input10	N6	
input11	P6	
input12	P10	
input13	N9	
input14	R2	
input15	R1	
input16	R7	
input17	R6	
input18	R9	
input19	P8	
input20	T3	
input21	T1	
input22	R8	
input23	T8	
input24	R10	

input25	T9	
input26	U2	
input27	U1	
input28	U4	
input29	U3	
input30	V4	
input31	W3	
input32	V5	
input33	W5	
input34	Y3	
input35	Y1	
input36	Y6	
input37	Y5	
input38	AA2	
input39	AA1	
input40	AA4	
input41	AA3	
input42	AB3	
input43	AB1	
input44	AC2	
input45	AC1	
input46	AB4	
input47	AC3	
input48	AB5	
input49	AC4	
input50	AD3	
input51	AD1	
input52	AE2	
input53	AE1	
input54	AD4	
input55	AF4	
Input56	AC5	
Input57	AD5	
Input58	AA7	
Input59	AA6	
Input60	AD6	
Input61	AF6	
Input62	W8	
Input63	W7	

Output0	G4	Circuit output signal
Output1	G3	
Output2	J2	
Output3	J1	
Output4	K5	
Output5	J5	
Output6	L2	
Output7	L1	
Output8	L9	
Output9	L8	
Output10	N2	
Output11	N1	
Output12	N5	
Output13	N4	
Output14	N8	
Output15	N7	
Output16	R4	
Output17	R3	
Output18	P5	
Output19	R5	
Output20	U5	
Output21	T4	
Output22	U7	
Output23	T6	
Output24	U9	
Output25	U8	
Output26	V3	
Output27	V1	
Output28	V7	
Output29	V6	
Output30	W2	
Output31	W1	
Output32	B2	
Output33	B1	
Output34	C2	
Output35	C1	
Output36	D3	
Output37	D1	
Output38	E2	

Output39	E1	
Output40	E4	
Output41	E3	
Output42	F3	
Output43	F1	
Output44	G2	
Output45	G1	
Output46	H3	
Output47	H1	
Output48	H6	
Output49	H5	
Output50	J4	
Output51	J3	
Output52	J9	
Output53	J7	
Output54	K3	
Output55	K1	
Output56	K7	
Output57	K6	
Output58	K9	
Output59	K8	
Output60	L4	
Output61	L3	
Output62	L7	
Output63	L6	

XC6SLX150TFGG676

	Pin Name	Comment
clk	AE15	User circuit clock pin connect to this pin
input0	L10	Circuit input signal
input1	K10	
input2	M3	
input3	M1	
input4	M8	

input5	M6	
input6	M10	
input7	M9	
input8	P3	
input9	P1	
input10	N6	
input11	P6	
input12	P10	
input13	N9	
input14	R2	
input15	R1	
input16	R7	
input17	R6	
input18	R9	
input19	P8	
input20	T3	
input21	T1	
input22	R8	
input23	T8	
input24	R10	
input25	T9	
input26	U2	
input27	U1	
input28	U4	
input29	U3	
input30	V4	
input31	W3	
input32	V5	
input33	W5	
input34	Y3	
input35	Y1	
input36	Y6	
input37	Y5	
input38	AA2	
input39	AA1	
input40	AA4	
input41	AA3	
input42	AB3	
input43	AB1	
input44	AC2	
input45	AC1	

input46	AB4	
input47	AC3	
input48	AB5	
input49	AC4	
input50	AD3	
input51	AD1	
input52	AE2	
input53	AE1	
input54	AD4	
input55	AF4	
Input56	AC5	
Input57	AD5	
Input58	AA7	
Input59	AA6	
Input60	AD6	
Input61	AF6	
Input62	W8	
Input63	W7	
Output0	G4	Circuit output signal
Output1	G3	
Output2	J2	
Output3	J1	
Output4	K5	
Output5	J5	
Output6	L2	
Output7	L1	
Output8	L9	
Output9	L8	
Output10	N2	
Output11	N1	
Output12	N5	
Output13	N4	
Output14	N8	
Output15	N7	
Output16	R4	
Output17	R3	
Output18	P5	
Output19	R5	
Output20	U5	
Output21	T4	

Output22	U7	
Output23	T6	
Output24	U9	
Output25	U8	
Output26	V3	
Output27	V1	
Output28	V7	
Output29	V6	
Output30	W2	
Output31	W1	
Output32	B2	
Output33	B1	
Output34	C2	
Output35	C1	
Output36	D3	
Output37	D1	
Output38	E2	
Output39	E1	
Output40	E4	
Output41	E3	
Output42	F3	
Output43	F1	
Output44	G2	
Output45	G1	
Output46	H3	
Output47	H1	
Output48	H6	
Output49	H5	
Output50	J4	
Output51	J3	
Output52	J9	
Output53	J7	
Output54	K3	
Output55	K1	
Output56	K7	
Output57	K6	
Output58	K9	
Output59	K8	
Output60	L4	
Output61	L3	
Output62	L7	

Output63	L6	

FPGA Dedicated IO

Pin table of FPGA dedicated IO (J30)

Top/Bottom Connector	FPGA IO Pin	Top/Bottom Connector	FPGA IO Pin
J30.01	AE25	J30.02	AD24
J30.03	AE26	J30.04	AD26
J30.05	AC25	J30.06	AC23
J30.07	AC26	J30.08	AC24
J30.09	AB24	J30.10	AB26
J30.11	VDD5V	J30.12	GND
J30.13	AA25	J30.14	AA23
J30.15	AA26	J30.16	AA24
J30.17	Y24	J30.18	W25
J30.19	Y26	J30.20	W26
J30.21	V23	J30.22	V24
J30.23	W24	J30.24	V26
J30.25	V20	J30.26	U25
J30.27	V21	J30.28	U26
J30.29	VDD3.3V	J30.30	GND
J30.31	U23	J30.32	U24
J30.33	U21	J30.34	U19
J30.35	U22	J30.36	U20
J30.37	T22	J30.38	T19
J30.39	T23	J30.40	T20

FPGA Dedicated IO (Differential signals)

Pin table of FPGA Differential signals (J29)

Connector	FPGA IO Pin	Connector	FPGA IO Pin
J29.001	B25	J29.061	GND

J29.002	B24	J29.062	GND
J29.003	B26	J29.063	L25
J29.004	A25	J29.064	L23
J29.005	GND	J29.065	L26
J29.006	GND	J29.066	L24
J29.007	C25	J29.067	GND
J29.008	D24	J29.068	GND
J29.009	C26	J29.069	L20
J29.010	D26	J29.070	L19
J29.011	GND	J29.071	L21
J29.012	GND	J29.072	K20
J29.013	D23	J29.073	GND
J29.014	E25	J29.074	GND
J29.015	C24	J29.075	M24
J29.016	E26	J29.076	M21
J29.017	GND	J29.077	M26
J29.018	GND	J29.078	M23
J29.019	E23	J29.079	GND
J29.020	F24	J29.080	GND
J29.021	E24	J29.081	M18
J29.022	F26	J29.082	N25
J29.023	GND	J29.083	M19
J29.024	GND	J29.084	N26
J29.025	F23	J29.085	GND
J29.026	F22	J29.086	GND
J29.027	G24	J29.087	N23
J29.028	G23	J29.088	N21
J29.029	GND	J29.089	N24
J29.030	GND	J29.090	N22
J29.031	G25	J29.091	GND
J29.032	H24	J29.092	GND
J29.033	G26	J29.093	N19
J29.034	H26	J29.094	N17
J29.035	GND	J29.095	N20
J29.036	GND	J29.096	N18
J29.037	H21	J29.097	GND
J29.038	H20	J29.098	GND
J29.039	H22	J29.099	P24
J29.040	G20	J29.100	P21
J29.041	GND	J29.101	P26

J29.042	GND	J29.102	P22
J29.043	J25	J29.103	GND
J29.044	J23	J29.104	GND
J29.045	J26	J29.105	P17
J29.046	J24	J29.106	R25
J29.047	GND	J29.107	P19
J29.048	GND	J29.108	R26
J29.049	J20	J29.109	GND
J29.050	K24	J29.110	GND
J29.051	J22	J29.111	R23
J29.052	K26	J29.112	R20
J29.053	GND	J29.113	R24
J29.054	GND	J29.114	R21
J29.055	K21	J29.115	GND
J29.056	K18	J29.116	GND
J29.057	K22	J29.117	R18
J29.058	K19	J29.118	T24
J29.059	GND	J29.119	R19
J29.060	GND	J29.120	T26

Clock

Pin Table of Configurable Clocks

Clock Generator	
FPGA IO Pin	Location
B12	Y3

Pin Table of 48MHz

48 MHz Clock Source
FPGA IO Pin
AE13

User-defined LEDs

Pin Table of User-defined LEDs

User-defined LEDs

Location	FPGA IO Pins
DS0	H18
DS1	F19
DS2	G19
DS3	H19
DS4	E20
DS5	F20
DS6	B21
DS7	C21

User-defined Switch

Pin Table of User-defined switch

User-defined Switch	
Location	FPGA IO Pins
SW6.1	D21
SW6.2	A22
SW6.3	B22
SW6.4	D22

User-defined Button

Pin Table of User-defined Button

User-defined Button	
Location	FPGA IO Pins
BT1	A23
BT2	B23

Macube-VEXV7 Xilinx Virtex-7 USB

Pin Tables

In this chapter, the FPGA pin, which connects to other devices such as I/O connectors, clocks etc, will be

shown in the following tables.

XC7V585TFFG1157

	Pin Name	Comment
clk	AA28	User circuit clock pin connect to this pin
input0	AE21	Circuit input signal
input1	AD19	
input2	AE19	
input3	AF21	
input4	AG21	
input5	AC18	
input6	AF25	
input7	AN33	
input8	AK34	
input9	AL34	
input10	AP33	
input11	AK32	
input12	AK33	
input13	AM32	
input14	AN32	
input15	AL33	
input16	AM33	
input17	AP30	
input18	AP31	
input19	AJ30	
input20	AK31	
input21	AM30	
input22	AN30	
input23	AJ29	
input24	AK29	
input25	AL31	
input26	AM31	
input27	AL29	
input28	AL30	
input29	AK28	
input30	AL28	
input31	AK26	
input32	AK27	
input33	AJ26	

input34	AJ27	
input35	AG25	
input36	AH25	
input37	AH24	
input38	AJ25	
input39	AL25	
input40	AL26	
input41	AM26	
input42	AM27	
input43	AN29	
input44	AP29	
input45	AM25	
input46	AN25	
input47	AM28	
input48	AN28	
input49	AP25	
input50	AP26	
input51	AN27	
input52	AP27	
input53	AF24	
input54	AC27	
input55	AG32	
Input56	AH32	
Input57	AH34	
Input58	AJ32	
Input59	AG33	
Input60	AH33	
Input61	AE32	
Input62	AE33	
Input63	AC30	
Output0	AG28	Circuit output signal
Output1	AH29	
Output2	AH27	
Output3	AH28	
Output4	AD26	
Output5	AD27	
Output6	AG26	
Output7	AG27	
Output8	AE26	
Output9	AF26	

Output10	AE23	
Output11	AE24	
Output12	AC25	
Output13	AD25	
Output14	AC24	
Output15	AD24	
Output16	AC23	
Output17	V23	
Output18	AA33	
Output19	AA34	
Output20	V34	
Output21	W34	
Output22	AB33	
Output23	AC34	
Output24	Y33	
Output25	Y34	
Output26	AC32	
Output27	AC33	
Output28	V32	
Output29	V33	
Output30	AB31	
Output31	AB32	
Output32	U30	
Output33	V30	
Output34	AA30	
Output35	AB30	
Output36	W32	
Output37	Y32	
Output38	Y31	
Output39	AA31	
Output40	AJ34	
Output41	AJ31	
Output42	Y29	
Output43	V27	
Output44	V28	
Output45	V29	
Output46	W29	
Output47	W26	
Output48	W27	
Output49	AB27	
Output50	AB28	

Output51	Y26	
Output52	Y27	
Output53	V24	
Output54	V25	
Output55	AA26	
Output56	AB26	
Output57	W24	
Output58	W25	
Output59	Y24	
Output60	AA25	
Output61	AA23	
Output62	AA24	
Output63	AB25	

XC7VX690TFFG1157

	Pin Name	Comment
clk	AA28	User circuit clock pin connect to this pin
input0	AE21	Circuit input signal
input1	AD19	
input2	AE19	
input3	AF21	
input4	AG21	
input5	AC18	
input6	AF25	
input7	AN33	
input8	AK34	
input9	AL34	
input10	AP33	
input11	AK32	
input12	AK33	
input13	AM32	
input14	AN32	
input15	AL33	
input16	AM33	
input17	AP30	

input18	AP31	
input19	AJ30	
input20	AK31	
input21	AM30	
input22	AN30	
input23	AJ29	
input24	AK29	
input25	AL31	
input26	AM31	
input27	AL29	
input28	AL30	
input29	AK28	
input30	AL28	
input31	AK26	
input32	AK27	
input33	AJ26	
input34	AJ27	
input35	AG25	
input36	AH25	
input37	AH24	
input38	AJ25	
input39	AL25	
input40	AL26	
input41	AM26	
input42	AM27	
input43	AN29	
input44	AP29	
input45	AM25	
input46	AN25	
input47	AM28	
input48	AN28	
input49	AP25	
input50	AP26	
input51	AN27	
input52	AP27	
input53	AF24	
input54	AC27	
input55	AG32	
Input56	AH32	
Input57	AH34	
Input58	AJ32	

Input59	AG33	
Input60	AH33	
Input61	AE32	
Input62	AE33	
Input63	AC30	
Output0	AG28	Circuit output signal
Output1	AH29	
Output2	AH27	
Output3	AH28	
Output4	AD26	
Output5	AD27	
Output6	AG26	
Output7	AG27	
Output8	AE26	
Output9	AF26	
Output10	AE23	
Output11	AE24	
Output12	AC25	
Output13	AD25	
Output14	AC24	
Output15	AD24	
Output16	AC23	
Output17	V23	
Output18	AA33	
Output19	AA34	
Output20	V34	
Output21	W34	
Output22	AB33	
Output23	AC34	
Output24	Y33	
Output25	Y34	
Output26	AC32	
Output27	AC33	
Output28	V32	
Output29	V33	
Output30	AB31	
Output31	AB32	
Output32	U30	
Output33	V30	
Output34	AA30	

Output35	AB30		
Output36	W32		
Output37	Y32		
Output38	Y31		
Output39	AA31		
Output40	AJ34		
Output41	AJ31		
Output42	Y29		
Output43	V27		
Output44	V28		
Output45	V29		
Output46	W29		
Output47	W26		
Output48	W27		
Output49	AB27		
Output50	AB28		
Output51	Y26		
Output52	Y27		
Output53	V24		
Output54	V25		
Output55	AA26		
Output56	AB26		
Output57	W24		
Output58	W25		
Output59	Y24		
Output60	AA25		
Output61	AA23		
Output62	AA24		
Output63	AB25		

XC7VX330TFFG1157

	Pin Name	Comment
clk	AA28	User circuit clock pin connect to this pin
input0	AE21	Circuit input signal
input1	AD19	
input2	AE19	

input3	AF21	
input4	AG21	
input5	AC18	
input6	AF25	
input7	AN33	
input8	AK34	
input9	AL34	
input10	AP33	
input11	AK32	
input12	AK33	
input13	AM32	
input14	AN32	
input15	AL33	
input16	AM33	
input17	AP30	
input18	AP31	
input19	AJ30	
input20	AK31	
input21	AM30	
input22	AN30	
input23	AJ29	
input24	AK29	
input25	AL31	
input26	AM31	
input27	AL29	
input28	AL30	
input29	AK28	
input30	AL28	
input31	AK26	
input32	AK27	
input33	AJ26	
input34	AJ27	
input35	AG25	
input36	AH25	
input37	AH24	
input38	AJ25	
input39	AL25	
input40	AL26	
input41	AM26	
input42	AM27	
input43	AN29	

input44	AP29	
input45	AM25	
input46	AN25	
input47	AM28	
input48	AN28	
input49	AP25	
input50	AP26	
input51	AN27	
input52	AP27	
input53	AF24	
input54	AC27	
input55	AG32	
Input56	AH32	
Input57	AH34	
Input58	AJ32	
Input59	AG33	
Input60	AH33	
Input61	AE32	
Input62	AE33	
Input63	AC30	
Output0	AG28	Circuit output signal
Output1	AH29	
Output2	AH27	
Output3	AH28	
Output4	AD26	
Output5	AD27	
Output6	AG26	
Output7	AG27	
Output8	AE26	
Output9	AF26	
Output10	AE23	
Output11	AE24	
Output12	AC25	
Output13	AD25	
Output14	AC24	
Output15	AD24	
Output16	AC23	
Output17	V23	
Output18	AA33	
Output19	AA34	

Output20	V34	
Output21	W34	
Output22	AB33	
Output23	AC34	
Output24	Y33	
Output25	Y34	
Output26	AC32	
Output27	AC33	
Output28	V32	
Output29	V33	
Output30	AB31	
Output31	AB32	
Output32	U30	
Output33	V30	
Output34	AA30	
Output35	AB30	
Output36	W32	
Output37	Y32	
Output38	Y31	
Output39	AA31	
Output40	AJ34	
Output41	AJ31	
Output42	Y29	
Output43	V27	
Output44	V28	
Output45	V29	
Output46	W29	
Output47	W26	
Output48	W27	
Output49	AB27	
Output50	AB28	
Output51	Y26	
Output52	Y27	
Output53	V24	
Output54	V25	
Output55	AA26	
Output56	AB26	
Output57	W24	
Output58	W25	
Output59	Y24	
Output60	AA25	

Output61	AA23	
Output62	AA24	
Output63	AB25	

User-defined LEDs

Pin Table of User-defined LEDs

User-defined LEDs	
User-defined Switch	
Location	FPGA IO Pins
SW6.1	AF20
SW6.2	AD21

Location	FPGA IO Pins
DS0	M32
DS1	M33
DS2	T33
DS3	R34
DS4	N33
DS5	N34

User-defined Switch

Pin Table of User-defined switch

User-defined Button

Pin Table of User-defined Button

User-defined Button	
Location	FPGA IO Pins

BT1	M12
-----	-----

VeriEnterprise Xilinx Artix-7 USB

Pin Tables

In this chapter, the FPGA pin, which connects to other devices such as I/O connectors, clocks etc, will be shown in the following tables.

XC7A200TFBG676

	Pin Name	Comment
clk	H21	User circuit clock pin connect to this pin
input0	D25	Circuit input signal
input1	D26	
input2	E25	
input3	E26	
input4	F24	
input5	F25	
input6	G24	
input7	G25	
input8	G26	
input9	H26	
input10	J24	
input11	J25	
input12	J26	
input13	K25	
input14	K26	
input15	L24	
input16	L25	
input17	M24	
input18	M25	
input19	M26	
input20	N26	
input21	P24	
input22	P25	
input23	P26	

input24	R25	
input25	R26	
input26	T24	
input27	T25	
input28	U24	
input29	U25	
input30	U26	
input31	V26	
input32	W24	
input33	W25	
input34	W26	
input35	Y25	
input36	Y26	
input37	AA24	
input38	AA25	
input39	AB24	
input40	AB25	
input41	AB26	
input42	AC26	
input43	E23	
input44	F22	
input45	F23	
input46	G22	
input47	H23	
input48	H24	
input49	J23	
input50	K22	
input51	K23	
input52	L22	
input53	L23	
input54	M22	
input55	N23	
Input56	N24	
Input57	P23	
Input58	R22	
Input59	R23	
Input60	T22	
Input61	T23	
Input62	U22	
Input63	V23	

Output0	V24	Circuit output signal
Output1	W23	
Output2	Y22	
Output3	Y23	
Output4	AA22	
Output5	AA23	
Output6	AC24	
Output7	V22	
Output8	V21	
Output9	Y21	
Output10	W21	
Output11	W20	
Output12	Y20	
Output13	U21	
Output14	U20	
Output15	G20	
Output16	G21	
Output17	J20	
Output18	K20	
Output19	L20	
Output20	M20	
Output21	M21	
Output22	N21	
Output23	N22	
Output24	P20	
Output25	P21	
Output26	R20	
Output27	R21	
Output28	T20	
Output29	H18	
Output30	H19	
Output31	J18	
Output32	J19	
Output33	K18	
Output34	L18	
Output35	L19	
Output36	M19	
Output37	N18	

Output38	N19		
Output39	P18		
Output40	P19		
Output41	R18		
Output42	T18		
Output43	T19		
Output44	U19		
Output45	V18		
Output46	V19		
Output47	W18		
Output48	W19		
Output49	V17		
Output50	R16		
Output51	R17		
Output52	P16		
Output53	N17		
Output54	N16		
Output55	M15		
Output56	L15		
Output57	K15		
Output58	J15		
Output59	L17		
Output60	T14		
Output61	U14		
Output62	V14		
Output63	T15		

VeriLite Altera Cyclone IV USB

EP4CE30F23C8

	Pin Name	Comment
clk	PIN_AB12	User circuit clock pin connect to this pin

input0	PIN_M3	Circuit input signal
input1	PIN_M2	
input2	PIN_M1	
input3	PIN_N1	
input4	PIN_P4	
input5	PIN_P3	
input6	PIN_P1	
input7	PIN_R2	
input8	PIN_R1	
input9	PIN_P6	
input10	PIN_R6	
input11	PIN_T4	
input12	PIN_M4	
input13	PIN_N2	
input14	PIN_U2	
input15	PIN_U1	
input16	PIN_V2	
input17	PIN_V1	
input18	PIN_W2	
input19	PIN_Y2	
input20	PIN_Y1	
input21	PIN_AA1	
input22	PIN_Y3	
input23	PIN_AA5	
input24	PIN_AB5	
input25	PIN_V5	
input26	PIN_V6	
input27	PIN_V7	
input28	PIN_Y6	
input29	PIN_W7	
input30	PIN_AB7	
input31	PIN_W8	
input32	PIN_Y8	
input33	PIN_AA8	
input34	PIN_U8	
input35	PIN_T8	
input36	PIN_T9	
input37	PIN_V8	
input38	PIN_U9	
input39	PIN_AA9	

input40	PIN_T10	
input41	PIN_T11	
input42	PIN_U10	
input43	PIN_V10	
input44	PIN_V11	
input45	PIN_W10	
input46	PIN_Y10	
input47	PIN_AA10	
input48	PIN_AB10	
input49	PIN_P2	
input50	PIN_M20	
input51	PIN_T12	
input52	PIN_T13	
input53	PIN_U12	
Output0	PIN_W1	Circuit output signal
Output1	PIN_AA4	
Output2	PIN_W6	
Output3	PIN_Y7	
Output4	PIN_AA7	
Output5	PIN_AB8	
Output6	PIN_U7	
Output7	PIN_AB9	
Output8	PIN_U13	
Output9	PIN_V13	
Output10	PIN_W13	
Output11	PIN_Y13	
Output12	PIN_AA13	
Output13	PIN_AB13	
Output14	PIN_U14	
Output15	PIN_V14	
Output16	PIN_AA14	
Output17	PIN_AB14	
Output18	PIN_R14	
Output19	PIN_R15	
Output20	PIN_T14	
Output21	PIN_T15	
Output22	PIN_U15	
Output23	PIN_V15	

Output24	PIN_W15	
Output25	PIN_AA15	
Output26	PIN_AB15	
Output27	PIN_AA16	
Output28	PIN_AB16	
Output29	PIN_U16	
Output30	PIN_U17	
Output31	PIN_W17	
Output32	PIN_Y17	
Output33	PIN_AA17	
Output34	PIN_AB17	
Output35	PIN_AB18	
Output36	PIN_AA20	
Output37	PIN_AB20	
Output38	PIN_AA21	
Output39	PIN_Y21	
Output40	PIN_Y22	
Output41	PIN_W20	
Output42	PIN_W21	
Output43	PIN_W22	
Output44	PIN_V21	
Output45	PIN_V22	
Output46	PIN_U19	
Output47	PIN_U20	
Output48	PIN_U21	
Output49	PIN_U22	
Output50	PIN_R21	
Output51	PIN_R22	
Output52	PIN_P21	
Output53	PIN_P22	

I/O

LED (Low Active)		
D0	PIN_B2	LED on PCB
D1	PIN_B1	

D2	PIN_C2	
D3	PIN_C1	
D4	PIN_D2	
D5	PIN_D1	
D6	PIN_E2	
D7	PIN_E1	

Switch (Low Active)		
S1	PIN_H2	
S2	PIN_H1	

Button (Low Active)		
BT1	PIN_F2	
BT2	PIN_F1	

J8 jumper			
VCC_ADJ		2.5V	short
VCC_ADJ		3.3V	open

General I/O J3			
PIN_B3	J3.1	J3.2	PIN_B4
PIN_A3	J3.3	J3.4	PIN_A4
PIN_C4	J3.5	J3.6	PIN_B6
PIN_C3	J3.7	J3.8	PIN_A6
PIN_E5 (Output Only)	J3.9	J3.10	PIN_E6 (Output Only)
VCC5V	J3.11	J3.12	GND
PIN_D7	J3.13	J3.14	PIN_B7
PIN_C6	J3.15	J3.16	PIN_A7
PIN_C7	J3.17	J3.18	PIN_G7
PIN_C8	J3.19	J3.20	PIN_F7
PIN_B8	J3.21	J3.22	PIN_G8
PIN_A8	J3.23	J3.24	PIN_F8
PIN_E7	J3.25	J3.26	PIN_B9
PIN_F9	J3.27	J3.28	PIN_A9
VCC_ADJ	J3.29	J3.30	GND
PIN_B11(Input Only)	J3.31	J3.32	PIN_A11 (Input Only)
PIN_B10	J3.33	J3.34	PIN_E10
PIN_A10	J3.35	J3.36	PIN_D10
PIN_F10	J3.37	J3.38	PIN_H10
PIN_G10	J3.39	J3.40	PIN_H11
General I/O J9			
PIN_F11	J9.1	J9.2	PIN_E12
PIN_E11	J9.3	J9.4	PIN_E14
PIN_D13	J9.5	J9.6	PIN_B13
PIN_C13	J9.7	J9.8	PIN_A13
PIN_B12 (Input Only)	J9.9	J9.10	PIN_A12(Input Only)

VCC5V	J9.11	J9.12	GND
PIN_F13	J9.13	J9.14	PIN_H14
PIN_G13	J9.15	J9.16	PIN_H15
PIN_G14	J9.17	J9.18	PIN_F14
PIN_E15	J9.19	J9.20	PIN_G15
PIN_G16	J9.21	J9.22	PIN_E16
PIN_F15	J9.23	J9.24	PIN_F16
PIN_B14	J9.25	J9.26	PIN_B15
PIN_A14	J9.27	J9.28	PIN_A15
VCC_ADJ	J9.29	J9.30	GND
PIN_B20(Output Only)	J9.31	J9.32	PIN_A20(Output Only)
PIN_B18	J9.33	J9.34	PIN_D19
PIN_A18	J9.35	J9.36	PIN_C19
PIN_B17	J9.37	J9.38	PIN_B16
PIN_A17	J9.39	J9.40	PIN_A16

VeriLite Xilinx Spartan-6 USB

XC6SLX16

	Pin Name	Comment
clk	H4	User circuit clock pin connect to this pin
input0	F1	Circuit input signal]
input1	G3	
input2	G1	
input3	H3	
input4	H2	
input5	H1	
input6	H5	
input7	J3	
input8	J1	
input9	K5	
input10	K6	
input11	J4	
input12	K2	
input13	K1	
input14	L4	
input15	L5	
input16	L3	

input17	L1	
input18	M5	
input19	M4	
input20	M3	
input21	M2	
input22	M1	
input23	N1	
input24	P2	
input25	P1	
input26	R2	
input27	P4	
input28	T4	
input29	N5	
input30	T5	
input31	M6	
input32	N6	
input33	P6	
input34	L7	
input35	P7	
input36	M7	
input37	R7	
input38	T7	
input39	P8	
input40	M9	
input41	N8	
input42	N9	
input43	P9	
input44	R9	
input45	T9	
input46	L10	
input47	M10	
input48	P11	
input49	T10	
input50	M12	
input51	M11	
input52	N12	
input53	P12	
Output0	N4	Circuit output signal

Output1	N3	
Output2	R1	
Output3	P5	
Output4	R5	
Output5	T6	
Output6	L8	
Output7	T8	
Output8	R12	
Output9	T12	
Output10	T14	
Output11	T13	
Output12	R14	
Output13	T15	
Output14	R15	
Output15	R16	
Output16	P15	
Output17	P16	
Output18	N14	
Output19	N16	
Output20	M13	
Output21	M14	
Output22	M15	
Output23	M16	
Output24	L12	
Output25	L13	
Output26	L14	
Output27	L16	
Output28	K12	
Output29	K11	
Output30	K15	
Output31	K16	
Output32	J11	
Output33	J12	
Output34	J13	
Output35	K14	
Output36	J14	
Output37	J16	
Output38	H13	
Output39	H14	
Output40	H15	

Output41	H16	
Output42	G14	
Output43	G16	
Output44	G12	
Output45	H11	
Output46	F12	
Output47	G11	
Output48	F13	
Output49	F14	
Output50	F15	
Output51	F16	
Output52	E13	
Output53	E12	

XC6SLX25

	Pin Name	Comment
clk	H4	User circuit clock pin connect to this pin
input0	F1	Circuit input signal
input1	G3	
input2	G1	
input3	H3	
input4	H2	
input5	H1	
input6	H5	
input7	J3	
input8	J1	
input9	K5	
input10	K6	
input11	J4	
input12	K2	
input13	K1	
input14	L4	
input15	L5	
input16	L3	

input17	L1	
input18	M5	
input19	M4	
input20	M3	
input21	M2	
input22	M1	
input23	N1	
input24	P2	
input25	P1	
input26	R2	
input27	P4	
input28	T4	
input29	N5	
input30	T5	
input31	M6	
input32	N6	
input33	P6	
input34	L7	
input35	P7	
input36	M7	
input37	R7	
input38	T7	
input39	P8	
input40	M9	
input41	N8	
input42	N9	
input43	P9	
input44	R9	
input45	T9	
input46	L10	
input47	M10	
input48	P11	
input49	T10	
input50	M12	
input51	M11	
input52	N12	
input53	P12	
Output0	N4	Circuit output signal

Output1	N3	
Output2	R1	
Output3	P5	
Output4	R5	
Output5	T6	
Output6	L8	
Output7	T8	
Output8	R12	
Output9	T12	
Output10	T14	
Output11	T13	
Output12	R14	
Output13	T15	
Output14	R15	
Output15	R16	
Output16	P15	
Output17	P16	
Output18	N14	
Output19	N16	
Output20	M13	
Output21	M14	
Output22	M15	
Output23	M16	
Output24	L12	
Output25	L13	
Output26	L14	
Output27	L16	
Output28	K12	
Output29	K11	
Output30	K15	
Output31	K16	
Output32	J11	
Output33	J12	
Output34	J13	
Output35	K14	
Output36	J14	
Output37	J16	
Output38	H13	
Output39	H14	
Output40	H15	

Output41	H16	
Output42	G14	
Output43	G16	
Output44	G12	
Output45	H11	
Output46	F12	
Output47	G11	
Output48	F13	
Output49	F14	
Output50	F15	
Output51	F16	
Output52	E13	
Output53	E12	

I/O

LED (Low Active)		
D0	A3	LED on PCB
D1	B3	
D2	A2	
D3	B2	
D4	B1	
D5	C1	
D6	C2	
D7	C3	

Switch (Low Active)		
SW1.1	E1	
SW1.2	E2	

Button (Low Active)		
BT1	D1	
BT2	D3	

J8 jumper			
VCC_ADJ		2.5V	short
VCC_ADJ		3.3V	open

--	--	--	--

General I/O J3			
D5	J3.1	J3.2	B5
C5	J3.3	J3.4	A5
F7	J3.5	J3.6	D6
E6	J3.7	J3.8	C6
E7	J3.9	J3.10	E8
VCC5V	J3.11	J3.12	GND
B6	J3.13	J3.14	C7
A6	J3.15	J3.16	A7
D8	J3.17	J3.18	B8
C8	J3.19	J3.20	A8
F9	J3.21	J3.22	C9(Input Only)
D9	J3.23	J3.24	A9(Input Only)
F10	J3.25	J3.26	E10
E11	J3.27	J3.28	C10
VCC_ADJ	J3.29	J3.30	GND
B10(Input Only)	J3.31	J3.32	A10(Input Only)
D11	J3.33	J3.34	C11
D12	J3.35	J3.36	A11
B12	J3.37	J3.38	C13
A12	J3.39	J3.40	A13

VeriLink In Matlab

Introduction

SMIMS VeriLink provides an easy way for user to compile, synthesis, download and verify the designed circuit with the SMIMS FPGA device.

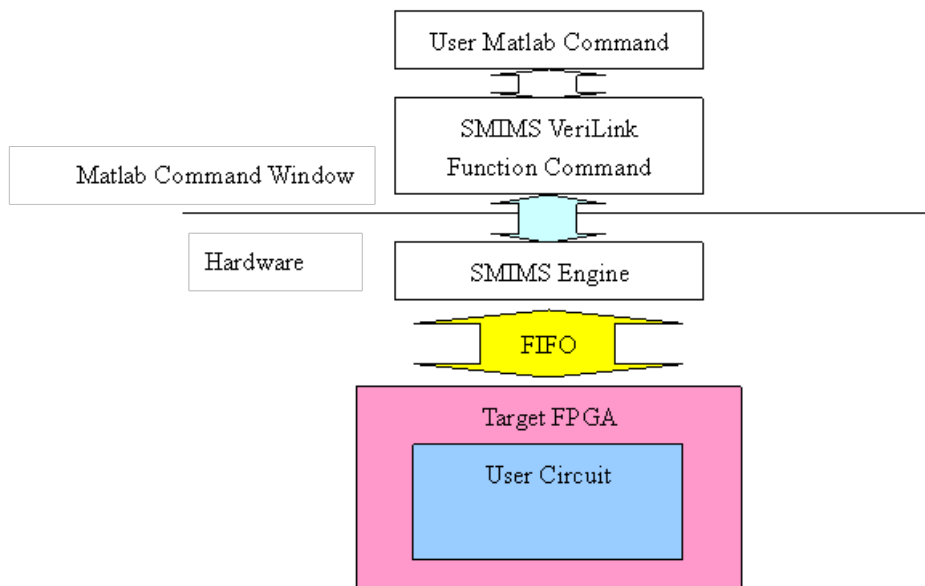
In the Matlab environment, user is able to progress the verification of the designed circuit.

VeriLink Matlab Interface

When SMIMS FPGA board is working in the VeriLink Matlab mode, the user FPGA chip will communicate with the USB device directly.

At this time, user can design the circuit in FPGA to deal with the data received from VeriLink API through the USB or to prepare the data for VeriLink API getting back.

The data transmission uses the FIFO interface. The main components of the messages are Read, Write, Data Input, Data, Output etc. In that point of view, the user FPGA treats itself as a FIFO for the PC side to access data.



Data Type Conversion Function

Function Name	Function description
SMIMS_Double2UINT16 (UserArray)	Convert Array From Double format to UINT16 format ° The conversion uses two's

	complement method °
SMIMS_Double2UINT16 (UserArray,Scale)	Convert Array From Double format to UINT16 format ° The conversion uses two's complement method ° The value in "scaled" user array is required to be between -32768 (intmin('int16')) and 32767 (intmax('int16')) °
SMIMS_PosDouble2UINT16 (UserArray)	Convert Array From Positive Double format to UINT16 format °
SMIMS_PosDouble2UINT16 (UserArray,Scale)	Convert Array From Double format to UINT16 format ° The value in "scaled" user array is required to be between 0 and 65535 °
UserArray : 1D Matlab built-in array (Double format) ° Scale : scale up factor , the value of UserArray will be scaled up by Scale and then trim to integer ° Return value : if flag is true , the conversion is successful, vice versa °	

SMIMS_UINT16ToDouble (UserArray)	Convert Array From Double format to UINT16 format ° The conversion uses two's complement method °
SMIMS_UINT16ToDouble (UserArray,Scale)	Convert Array From UINT16 format to Double format with a scale ° The conversion uses two's complement method
SMIMS_UINT16ToPosDouble (UserArray)	Convert Array From Double format to UINT16 format °
SMIMS_UINT16ToPosDouble (UserArray,Scale)	Convert Array From UINT16 to Positive Double format with a scale ° The conversion uses two's complement method °
UserArray : 2D Matlab built-in array (uint16 format) °	

Scale : scale factor , UserArray will be transformed to double then multiply by Scale °
Return value : if flag is true , the conversion is successful , vice versa °

VeriEnterprise Xilinx NV5 USB

VeriEnterprise Xilinx NV5 USB in Matlab command window

Support Models:

VEXNV5-USB-STD

VEXNV5-USB-EDU-STD

Target FPGA : Xilinx XC5VLX110 、XC5VLX330

Requirement

1. SMIMS VeriEnterprise Xilinx NV5 USB FPGA hardware device.
2. Matlab®/Simulink® of Mathwork®
3. Xilinx ISE design software.

VeriLink Function Introduction

This section introduces the way of the Matlab program to communicate with the user circuit in the FPGA by veriLink functions. After installed VeriLink in the PC, Matlab program is able to use VeriLink function.

SMIMS Engine VeriLink function descriptions:

Since the VeriLink function supports up to 2 same type SMIMS FPGA boards in a single system, the 1st parameter of those functions indicates the selected board to operate. The board ID of the first plug-in board is 0.













Function Name	Function description
<code>smMatVEXNV5(iBoardID, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx', ClkMode)</code>	Initial the SMIMS Engine. Return true if success, false vice versa. The first input parameter is the board ID. The second parameter is the serial number. The third

	one is FPGA clock operation mode, the clock mode and clock speed shows below.								
	<table> <tr> <th colspan="2">ClkMode</th></tr> <tr> <th>Value</th><th>Value</th></tr> <tr> <td>0</td><td>48 MHz</td></tr> <tr> <td>1</td><td>User offer clock input(Refer to the VEXNV5_Manual.pdf)</td></tr> </table>	ClkMode		Value	Value	0	48 MHz	1	User offer clock input(Refer to the VEXNV5_Manual.pdf)
ClkMode									
Value	Value								
0	48 MHz								
1	User offer clock input(Refer to the VEXNV5_Manual.pdf)								
<code>smMatVEXNV5(iBoardID,'Close')</code>	Close the SMIMS Engine, Return true if success, false vice versa.								
<code>[retvalue ReadBuf] = smMatVEXNV5(iBoardID,'Read', ReadCount)</code>	Read data from hardware. The return value <code>ReadBuf</code> is buffer to store data. The third parameter <code>ReadCount</code> is total data count to read. If the operation is successful, the return value <code>retvalue</code> will be true, false vice versa.								
<code>smMatVEXNV5(iBoardID,'Write', WriteBuf, WriteCount)</code>	Send fata to hardware. The third parameter <code>WriteBuf</code> is data buffer to send and the data type is unit 16. The fourth parameter <code>WriteCount</code> is total data count to Send. If the operation is successful, this function will return true, false vice versa.								
<code>smMatVEXNV5(iBoardID,'ChSel', Channel)</code>	Set 8 bits user-define data. Note: If the function is called without FIFO data empty, this function will be block and SDK_CH[7:0] will not changed immediately. SDK_CH[7:0] will be altered after all data in FIFO buffer is read out. The third parameter <code>Channel</code> is the setting value.								
<code>smMatVEXNV5(iBoardID,'Program', 'BitFilePath')</code>	Program first FPGA. The third parameter <code>'BitFilePath'</code> is the FPGA programming binary file.								
<code>smMatVEXNV5(iBoardID,'ProgramFPGA2', 'BitFilePath')</code>	Program second FPGA. The third parameter <code>'BitFilePath'</code> is the FPGA programming binary file.								
<code>smMatVEXNV5(iBoardID,'GetErrMsg')</code>	Retrieve SMIMS Engine Error Message, if an error occurs.								
<code>smMatVEXNV5('H')</code>	Print the help message in the Matlab Command Window.								

Hardware Interface

The hardware interface for user's design in FPGA is used the FIFO interface. The signals and descriptions of hardware interface are listed as the following table.

Signal Name	Signal Type	Function Description
-------------	-------------	----------------------

SDK_FIFO_CLK	In → 	ClkMode =0, 48 MHz Clock Source for VEX platform.
SDK_CLK_OUT	← Out 	ClkMode =1, User offer clock for SMIMS Engine.
SDK_RSTN	In → 	After programming a circuit into FPGA, the signal will be set to 0 . When ' Open ' command is called, the signal will become 1 . And when ' Close ' command is called, the signal will become 0 .
SDK_CH[7:0]	In → 	User can use this 8 bits channel for sending signal to the user Circuit.
SDK_FIFO_RD	← Out 	User design invokes this signal to read FIFO data from SMIMS engine.
SDK_FIFO_WR	← Out 	User design invokes this signal to write data to SMIMS engine FIFO buffer.
SDK_FIFO_DI[15:0]	In → 	Data bus from SMIMS Engine FIFO.
SDK_FIFO_DO[15:0]	← Out 	Data bus from user design to be written to SMIMS Engine FIFO buffer.
SDK_FIFO_Full	In → 	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is full and it can not be written any data more.
SDK_FIFO_Empty	In → 	When this signal is 1, SMIMS Engine FIFO buffer for user design is empty. There is no data to be read.
SDK_FIFO_AlmostFull	In → 	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is only one space left.
SDK_FIFO_AlmostEmpty	In → 	When this signal is 1, SMIMS Engine FIFO buffer for user design is only one data available.

VeriEnterprise Xilinx USB in VeriLink function mode provides the 16bits data bus. Because the data transmit as FIFO, if users want to use a SRAM, they should generate their own address for the SRAM.

There are two 1024 x 16 bits FIFO inside the SMIMS Engine. In the example "Inverter", if write a data size bigger than the FIFO buffer size of the SMIMS Engine, the program will be blocked in the function `smMatVEXNV5(0, 'Write', WriteBuf, WriteCount)`. The picture shows this situation below.

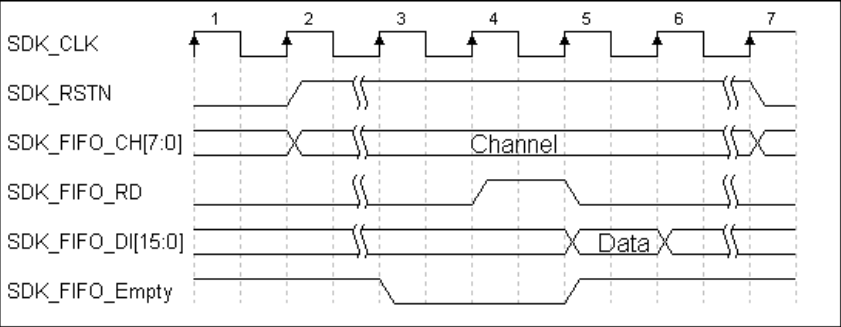


Timing

Write Single Data from PC to FPGA:

VeriLink Function : `smMatVEXNV5(0, 'Write', WriteBuf, 1)`

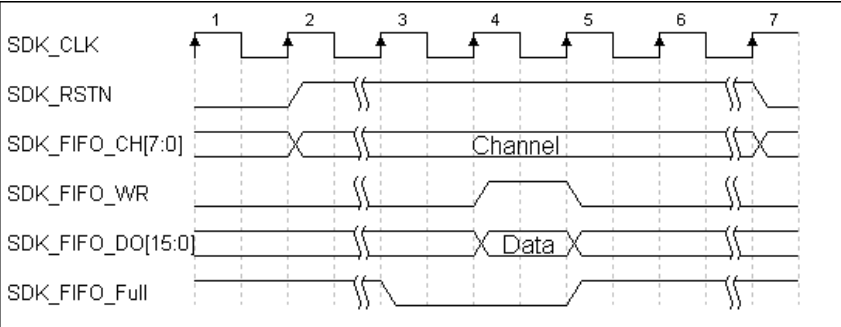
The timing diagram is shown as follows. When SDK_FIFO_Empty signal is low and SDK_FIFO_RD activates for one clock period, the data will be read in the next clock period.



Read Single Data from FPGA to PC:

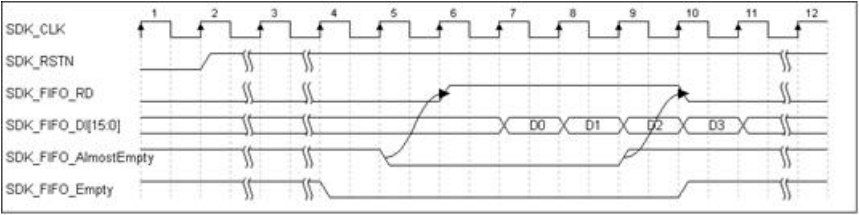
VeriLink Function : `smMatVEXNV5(0, 'Read', 1)`

The timing diagram is shown as follows. When SDK_FIFO_Full signal is low, sends out the SDK_FIFO_WR and SDK_FIFO_DO data in the same time.

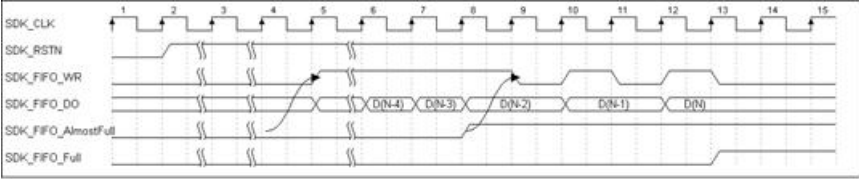


Write Multi Data from PC to FPGA:

VeriLink Function : `smMatVEXNV5 (0, 'Write', WriteBuf, 4)`



■ Read Multi Date from FPGA to PC:
VeriLink Function : `smMatVEXNV5 (0, 'Read', 5)`



PIN Map Information

(XC5VLX110、XC5VLX220、XC5VLX330)

Signal	FPGA IO Pin	Signal	FPGA IO Pin
SDK_CLK	K13	SDK_CLK_OUT	L9
SDK_RSTN	AU6	SDK_CH[0]	AH8
SDK_FIFO_WR	AV8	SDK_CH[1]	AG8
SDK_FIFO_RD	AU8	SDK_CH[2]	AG7
SDK_FIFO_Full	AV6	SDK_CH[3]	AF7
SDK_FIFO_Empty	AF11	SDK_CH[4]	AF10
SDK_FIFO_AlmostFull	AU7	SDK_CH[5]	AF9
SDK_FIFO_AlmostEmpty	AE10	SDK_CH[6]	AE9
SDK_FIFO_Interrupt	AT7	SDK_CH[7]	AE8
SDK_FIFO_DI [0]	E7	SDK_FIFO_DO [0]	AL11
SDK_FIFO_DI [1]	E8	SDK_FIFO_DO [1]	AL12
SDK_FIFO_DI [2]	K8	SDK_FIFO_DO [2]	AK10
SDK_FIFO_DI [3]	K9	SDK_FIFO_DO [3]	AL10
SDK_FIFO_DI [4]	H8	SDK_FIFO_DO [4]	AJ10
SDK_FIFO_DI [5]	J8	SDK_FIFO_DO [5]	AJ11
SDK_FIFO_DI [6]	F9	SDK_FIFO_DO [6]	AJ12
SDK_FIFO_DI [7]	G9	SDK_FIFO_DO [7]	AK12
SDK_FIFO_DI [8]	E10	SDK_FIFO_DO [8]	AJ13
SDK_FIFO_DI [9]	E9	SDK_FIFO_DO [9]	AK13
SDK_FIFO_DI [10]	H9	SDK_FIFO_DO [10]	AK14
SDK_FIFO_DI [11]	H10	SDK_FIFO_DO [11]	AK15

SDK_FIFO_DI [12]	F10	Input	SDK_FIFO_DO [12]	AL15	Output
SDK_FIFO_DI [13]	F11	Input	SDK_FIFO_DO [13]	AL14	Output
SDK_FIFO_DI [14]	F12	Input	SDK_FIFO_DO [14]	AH16	Output
SDK_FIFO_DI [15]	G11	Input	SDK_FIFO_DO [15]	AJ15	Output

Note : Please change the setting of Pin I/O Std. to be LVCMOS33.

Example

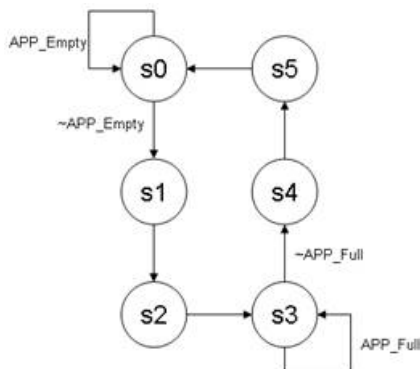
Example Inverter

Example Description: (example\Inverter)

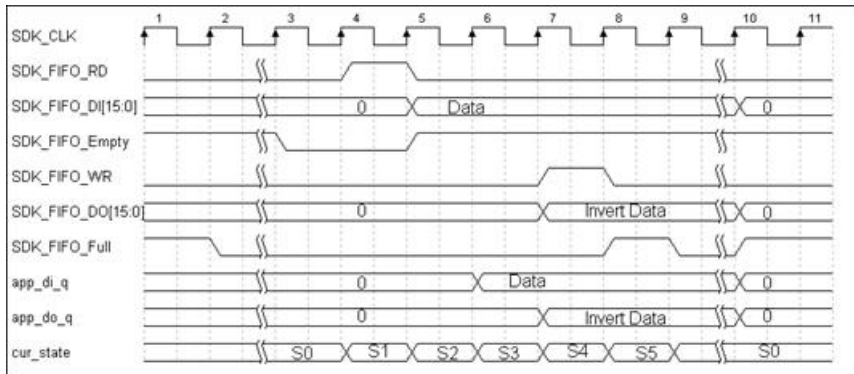
This example, Inverter, show how to use the VeriLink Function to design the Matlab program working with user's circuit. There are hardware and software two parts.

Hardware

The hardware design (single read/write) is to read data from Input FIFO, pass through an inverter, and then output the data to the Output FIFO. Here use a FSM (Finite state machine) to implement it. The state diagram is shown as follows.



The detailed timing diagram is shown in the following figure.



Software

The software part runs the script file in Matlab. This example uses the least SMIMS function to explain how to use the function. It is a simple and quick introduction. The software source code Matlab script file is TestVEXNV5SDK.m

Software program explanation (TestVEX NV5SDK.m)

```
% Test run SMIMS VEX NV5 SDK
```

```
Clear
```

```
%Program .bit file tp FPGA
```

```
smMatVEXNV5(0, 'Program', 'c:\app_example.bit');
```

```
% Open FPGA Board and set serial number.
```

```
smMatVEXNV5(0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx', 0);
```

```
for i=1:10
```

```
    writebuf(i)=uint16(i);
```

```
end
```

```
%Write to FPGA Board
```

```
smMatVEXNV5(0, 'Write', writebuf, 10);
```

```
%Read From FPGA Board
```

```
[ret readbuf] = smMatVEXNV5(0, 'Read', 10);
```

```
disp(readbuf);
```

```
%close FPGA Board  
smMatVEXNV5(0, 'Close');
```

Run and see the result as follows.

This circuit does invert, so the input data 1 to 10 will get the result 65534 to 65525.

```
>> TestVEXNV5SDK  
65534  
65533  
65532  
65531  
65530  
65529  
65528  
65527  
65526  
65525
```

```
>>
```

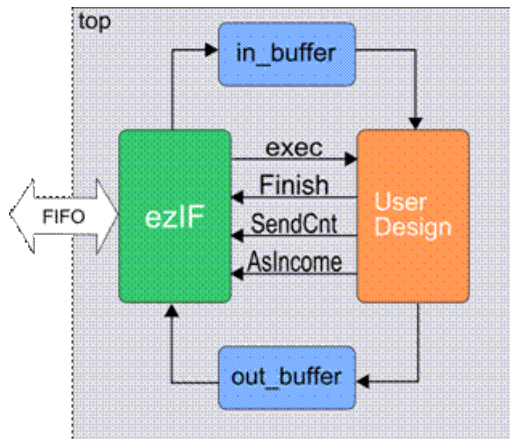
ezIF Interface

Introduce

Interface Description: (example\ Matlab\ezIF)

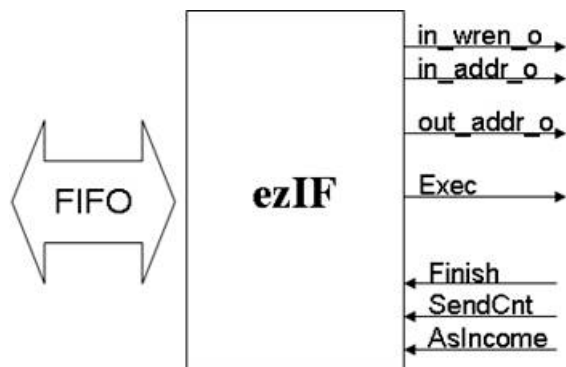
The ezIF is an interface to isolate users' design and FIFO interface. It can simplify user's design flow. The data received from software will be written in the memory by the ezIF. After ezIF writes all data in the memory, it will send a signal to info the User Design starts to work. After the User Design finished the process, it needs to send a signal to info the ezIF and a signal to tell whether to send data back to software side or not. The ezIF will automatically send user data back to the software side. The integration of ezIF and user design is shown as follow picture.

The ezIF is just an example to help user the combine the design with SMIMS development interface. If user needs more efficient or flexible operation mode, please design another mechanism to replace the ezIF.



Interface Operation

The interface operation of ezIF shows as follow diagram. The FIFO side is skipped. Here just introduces the right side signals.

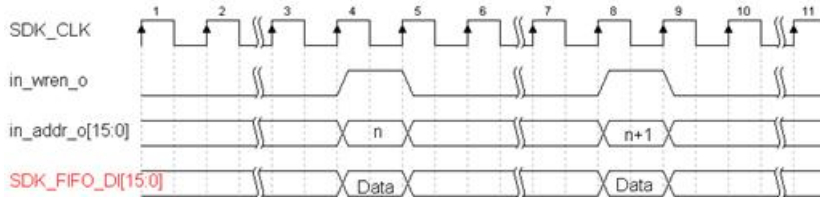


Signal Name	IO Type	Functional description
in_wren_o	output	Memory writing signal. 1 means to write data to the memory.
in_addr_o	output	Memory writing address
out_addr_o	output	Memory reading address for sending data to PC side.
Exec	output	Starts the user design
Finish	input	Info the user design finished the process.
SendCnt	input	The data count of sending data back to PC.
AsIncome	input	If the sending back data has the same counts as received

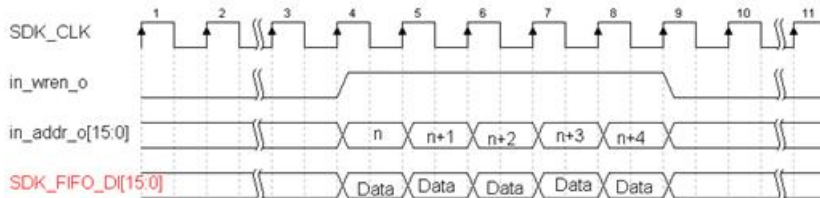
data, then set this signal be 1.

Timing

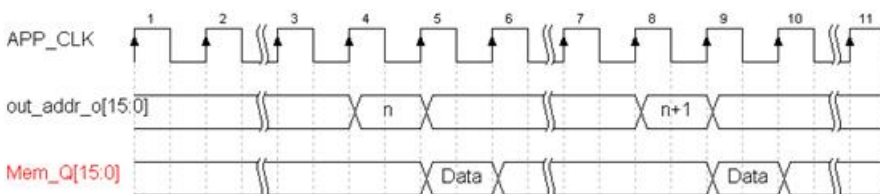
- ezIF writes single data to the memory (the data source is SDK_FIFO_DI of the FIFO).



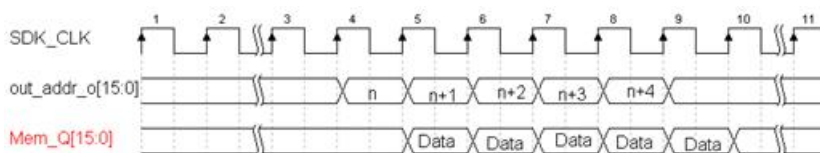
- ezIF writes burst data to the memory (the data source is SDK_FIFO_DI of the FIFO).



- ezIF reads single data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_FIFO_DO).

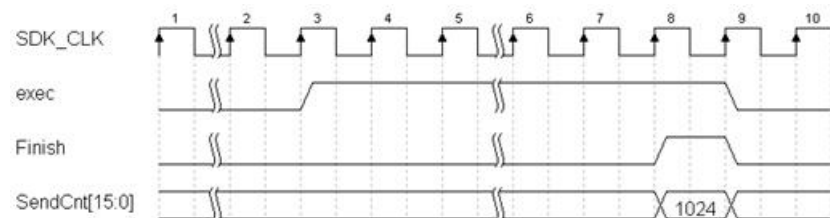


- ezIF reads burst data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_FIFO_DO).

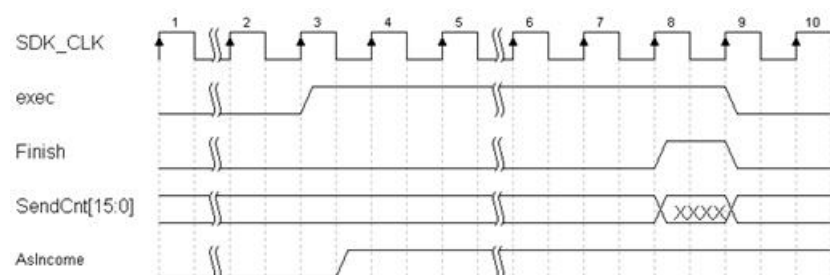


ezIF After the ezIF finished the receiving date, it will keep the Exec signal to be 1. The user design receives this signal will start to operate until the circuit finished the process, it will send out a Finish signal. Thus the Exec signal will be 0. When the user design sends out the Finish signal, the SendCnt should be set

at the same time with how many data counts needs to send back to the software side. Therefore, the ezIF will follow the signals to send the data back.



If the sending back data has the same counts as received data, then set this signal be 1. In the mean time, the SendCnt will be ignored.



Software Control Interface

In the software side, user needs to set up the data counts before sending data. Before setting the data counts, the APP_CH of FIFO needs to be 1. After setting the data counts, set the APP_CH back to 0. The source code shown as follows. The important is that the settings needs a lot of time. Only if every time sends different data counts, otherwise please do not set it before every sending. The ideal way is setting once and then sends data continuously, until the data count is different then setup the new data count.

```
smMatVEXNV5(0, 'ChSel', 1);
Cmd = data counts
smMatVEXNV5(0, 'Write', Cmd,1);
smMatVEXNV5(0, 'ChSel', 0);
```

Example Code

```
% Test run SMIMS VEX NV5 SDK
close all;
clear all;

rand('seed',0);

% Open FPGA Board and set serial number.
smMatVEXNV5(0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx', 0);

for i=1:4095

    smMatVEXNV5(0, 'ChSel', 1);
    Cmd = i;
    smMatVEXNV5(0, 'Write', Cmd,1);
    smMatVEXNV5(0, 'ChSel', 0);

    fprintf('Send to HW Data Count : %d ',i);
    flag = 1;
    // generate the random data
    WriteBuffer=rem(round(4096*rand(1,i)),4096);

    // send data to the circuit
    smMatVEXNV5(0, 'Write', WriteBuffer,i);

    // receive data from the circuit
    [ret ReadBuffer] = smMatVEXNV5(0, 'Read', i);

    // check the operation result
    for j=1:i
        if ReadBuffer(j) ~= (4096-WriteBuffer(j))
            flag = 0;
        end
    end
end
```

```

    if(flag == 1)
        fprintf('Check ok...\n');
    else
        fprintf('Error !\n');
        break;
    end
end

smMatVEXNV5(0, 'Close');

```

VeriEnterprise Xilinx SP6 USB

VeriEnterprise Xilinx SP6 USB in Matlab command window

Support Models:

VEXSP6-USB-STD

VEXSP6-USB-EDU-STD

Target FPGA : Xilinx XC6SLX150

Requirement

1. SMIMS VeriEnterprise Xilinx SP6 USB FPGA hardware device.
2. Matlab®/Simulink® of Mathwork®
3. Xilinx ISE design software.

VeriLink Function Introduction

This section introduces the way of the Matlab program to communicate with the user circuit in the FPGA by veriLink functions. After installed VeriLink in the PC, Matlab program is able to use VeriLink function.


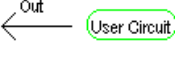
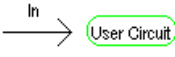


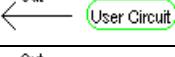
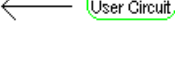
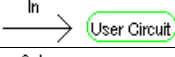
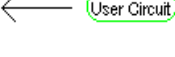


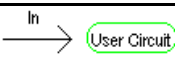
SMIMS Engine VeriLink function descriptions:

Since the VeriLink function supports up to 2 same type SMIMS FPGA boards in a single system, the 1st parameter of those functions indicates the selected board to operate. The board ID of the first plug-in board is 0.

Function Name	Function description								
smMatVEXSP6 (iBoardID, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx', ClkMode)	Initial the SMIMS Engine. Return true if success, false vice versa. The first input parameter is the board ID. The second parameter is the serial number. The third one is FPGA clock operation mode, the clock mode and clock speed shows below. <table> <tr> <th colspan="2">ClkMode</th></tr> <tr> <th>Value</th><th>Value</th></tr> <tr> <td>0</td><td>48 MHz</td></tr> <tr> <td>1</td><td>User offer clock input(Refer to the VEXSP6_Manual.pdf)</td></tr> </table>	ClkMode		Value	Value	0	48 MHz	1	User offer clock input(Refer to the VEXSP6_Manual.pdf)
ClkMode									
Value	Value								
0	48 MHz								
1	User offer clock input(Refer to the VEXSP6_Manual.pdf)								
smMatVEXSP6 (iBoardID, 'Close')	Close the SMIMS Engine, Return true if success, false vice versa.								
[retvalue ReadBuf] = smMatVEXSP6 (iBoardID, 'Read', ReadCount)	Read data from hardware. The return value ReadBuf is buffer to store data. The third parameter ReadCount is total data count to read. If the operation is successful, the return value retvalue will be true, false vice versa.								
smMatVEXSP6 (iBoardID, 'Write', WriteBuf, WriteCount)	Send fata to hardware. The third parameter WriteBuf is data buffer to send and the data type is unit 16. The fourth parameter WriteCount is total data count to Send. If the operation is successful, this function will return true, false vice versa.								
smMatVEXSP6 (iBoardID, 'ChSel', Channel)	Set 8 bits user-define data. Note: If the function is called without FIFO data empty, this function will be block and SDK_CH[7:0] will not changed immediately. SDK_CH[7:0] will be altered after all data in FIFO buffer is read out. The third parameter Channel is the setting value.								
smMatVEXSP6 (iBoardID, 'Program', 'BitFilePath')	Program first FPGA. The third parameter 'BitFilePath'is the FPGA programming binary file.								
smMatVEXSP6 (iBoardID, 'ProgramFPGA2', 'BitFilePath')	Program second FPGA. The third parameter 'BitFilePath'is the FPGA programming binary file.								
smMatVEXSP6 (iBoardID, 'GetErrMsg')	Retrieve SMIMS Engine Error Message, if an error occurs.								
smMatVEXSP6 ('H')	Print the help message in the Matlab Command Window.								

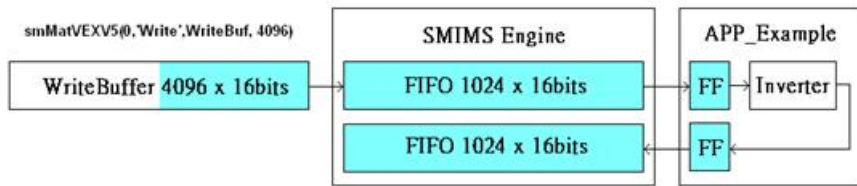
Hardware Interface

The hardware interface for user's design in FPGA is used the FIFO interface. The signals and descriptions of hardware interface are listed as the following table.

Signal Name	Signal Type	Function Description
SDK_FIFO_CLK	 In	ClkMode =0, 48 MHz Clock Source for VEX platform.
SDK_CLK_OUT	 Out	ClkMode =1, User offer clock for SMIMS Engine.
SDK_RSTN	 In	After programming a circuit into FPGA, the signal will be set to 0 . When ' Open ' command is called, the signal will become 1 .
SDK_CH[7:0]	 In	User can use this 8 bits channel for sending signal to the user Circuit.
SDK_FIFO_RD	 Out	User design invokes this signal to read FIFO data from SMIMS engine.
SDK_FIFO_WR	 Out	User design invokes this signal to write data to SMIMS engine FIFO buffer.
SDK_FIFO_DI[15:0]	 In	Data bus from SMIMS Engine FIFO.
SDK_FIFO_DO[15:0]	 Out	Data bus from user design to be written to SMIMS Engine FIFO buffer.
SDK_FIFO_Full	 In	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is full and it can not be written any data more.
SDK_FIFO_Empty	 In	When this signal is 1, SMIMS Engine FIFO buffer for user design is empty. There is no data to be read.
SDK_FIFO_AlmostFull	 In	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is only one space left.
SDK_FIFO_AlmostEmpty	 In	When this signal is 1, SMIMS Engine FIFO buffer for user design is only one data available.

VeriEnterprise Xilinx USB in VeriLink function mode provides the 16bits data bus. Because the data transmit as FIFO, if users want to use a SRAM, they should generate their own address for the SRAM.

There are two 1024 x 16 bits FIFO inside the SMIMS Engine. In the example "Inverter", if write a data size bigger than the FIFO buffer size of the SMIMS Engine, the program will be blocked in the function [smMatVEXSP6\(0, 'Write', WriteBuf, WriteCount\)](#). The picture shows this situation below.

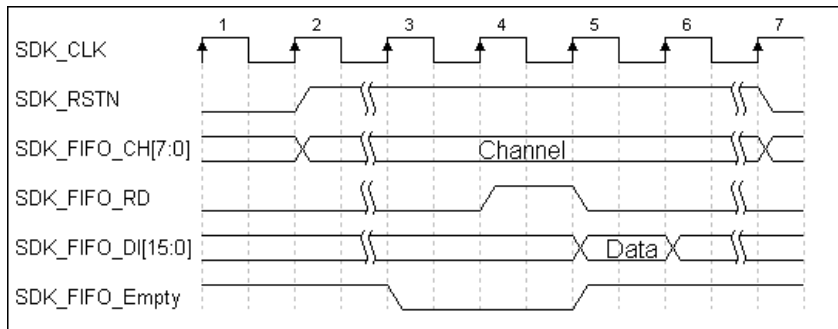


Timing

■ Write Single Data from PC to FPGA:

VeriLink Function : `smMatVEXSP6(0, 'Write', WriteBuf, 1)`

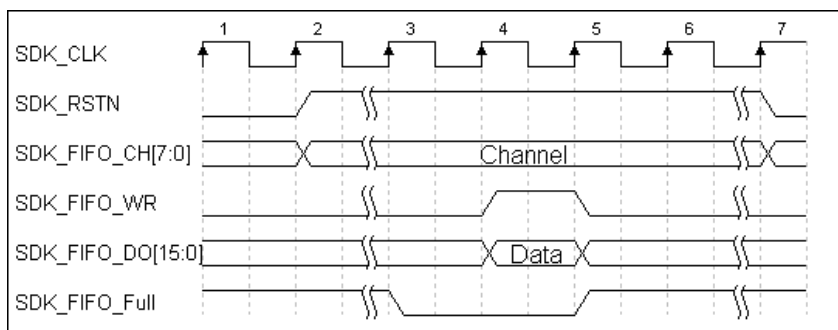
The timing diagram is shown as follows. When `SDK_FIFO_Empty` signal is low and `SDK_FIFO_RD` activates for one clock period, the data will be read in the next clock period.



■ Read Single Date from FPGA to PC:

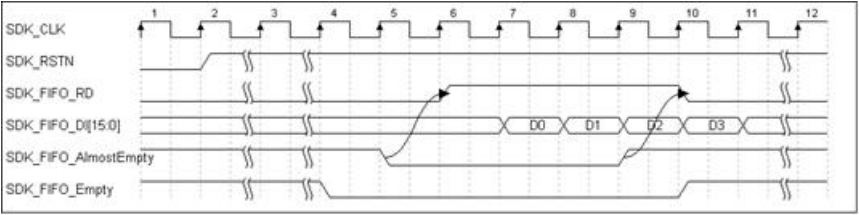
VeriLink Function : `smMatVEXSP6(0, 'Read', 1)`

The timing diagram is shown as follows. When `SDK_FIFO_Full` signal is low, sends out the `SDK_FIFO_WR` and `SDK_FIFO_DO` data in the same time.

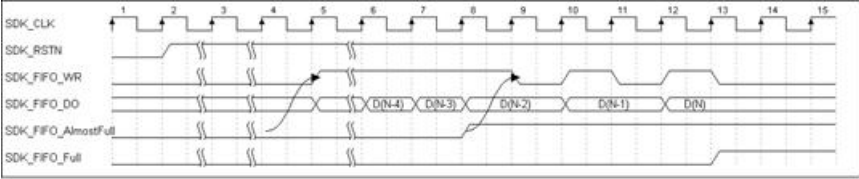


■ Write Multi Data from PC to FPGA:

VeriLink Function : `smMatVEXSP6 (0, 'Write', WriteBuf, 4)`



■ Read Multi Data from FPGA to PC:
VeriLink Function : smMatVEXSP6 (0, 'Read', 5)



PIN Map Information

(XC6SLX150)

Signal	FPGA IO Pin	Signal	FPGA IO Pin
SDK_CLK	F2	SDK_CLK_OUT	AC13
SDK_RSTN	K3	SDK_CH[0]	V13
SDK_FIFO_WR	P15	SDK_CH[1]	AB11
SDK_FIFO_RD	P16	SDK_CH[2]	AA11
SDK_FIFO_Full	L3	SDK_CH[3]	V14
SDK_FIFO_Empty	M5	SDK_CH[4]	U15
SDK_FIFO_AlmostFull	L1	SDK_CH[5]	AC12
SDK_FIFO_AlmostEmpty	M4	SDK_CH[6]	AA12
SDK_FIFO_Interrupt	C15	SDK_CH[7]	AB13
SDK_FIFO_DI [0]	A11	SDK_FIFO_DO [0]	Y16
SDK_FIFO_DI [1]	H12	SDK_FIFO_DO [1]	AF18
SDK_FIFO_DI [2]	G11	SDK_FIFO_DO [2]	AD18
SDK_FIFO_DI [3]	B12	SDK_FIFO_DO [3]	W17
SDK_FIFO_DI [4]	A12	SDK_FIFO_DO [4]	V16
SDK_FIFO_DI [5]	F12	SDK_FIFO_DO [5]	AD19
SDK_FIFO_DI [6]	E12	SDK_FIFO_DO [6]	AC19
SDK_FIFO_DI [7]	D12	SDK_FIFO_DO [7]	AA19
SDK_FIFO_DI [8]	C12	SDK_FIFO_DO [8]	Y18
SDK_FIFO_DI [9]	G13	SDK_FIFO_DO [9]	AD21
SDK_FIFO_DI [10]	F14	SDK_FIFO_DO [10]	AC20
SDK_FIFO_DI [11]	F13	SDK_FIFO_DO [11]	AF19

SDK_FIFO_DI [12]	D13	Input	SDK_FIFO_DO [12]	AE19	Output
SDK_FIFO_DI [13]	B16	Input	SDK_FIFO_DO [13]	AF20	Output
SDK_FIFO_DI [14]	A16	Input	SDK_FIFO_DO [14]	AF21	Output
SDK_FIFO_DI [15]	J14	Input	SDK_FIFO_DO [15]	AE21	Output

Note : Please change the setting of Pin I/O Std. to be LVCMOS33.

Example

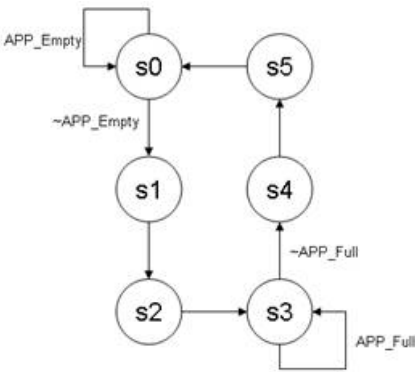
Example Inverter

Example Description: (example\Inverter)

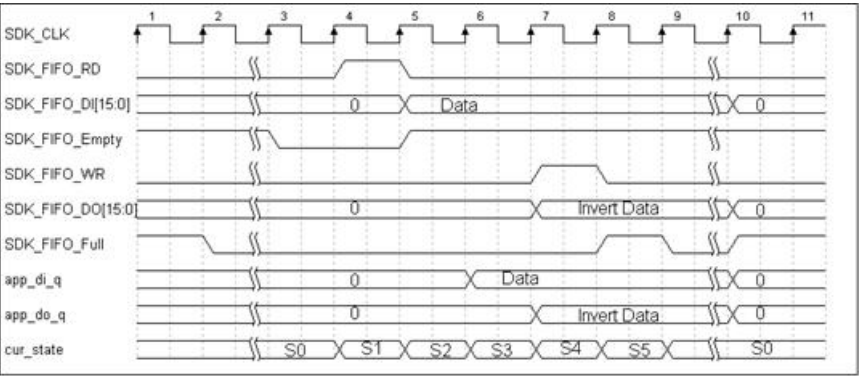
This example, Inverter, show how to use the VeriLink Function to design the Matlab program working with user's circuit. There are hardware and software two parts.

Hardware

The hardware design (single read/write) is to read data from Input FIFO, pass through an inverter, and then output the data to the Output FIFO. Here use a FSM (Finite state machine) to implement it. The state diagram is shown as follows.



The detailed timing diagram is shown in the following figure.



Software

The software part runs the script file in Matlab. This example uses the least SMIMS function to explain how to use the function. It is a simple and quick introduction. The software source code Matlab script file is TestVEXSP6SDK.m

Software program explanation (TestVEXSP6SDK.m)

```
% Test run SMIMS VEX SP6 SDK
```

```
Clear
```

```
%Program .bit file tp FPGA
```

```
smMatVEXSP6(0, 'Program', 'c:\app_example.bit');
```

```
% Open FPGA Board and set serial number.
```

```
smMatVEXSP6(0, 'Open', 'xxx-xxx-xxx-xxx-xxx-xxx', 0);
```

```
for i=1:10
```

```
    writebuf(i)=uint16(i);
```

```
end
```

```
%Write to FPGA Board
```

```
smMatVEXSP6(0, 'Write', writebuf, 10);
```

```
%Read From FPGA Board
```

```
[ret readbuf] = smMatVEXSP6(0, 'Read', 10);
```

```
disp(readbuf);
```

```
%close FPGA Board
```

```
smMatVEXSP6(0, 'Close');
```

Run and see the result as follows.

This circuit does invert, so the input data 1 to 10 will get the result 65534 to 65525.

```
>> TestVEXSP6SDK
```

```
65534
```

```
65533
```

```
65532
```

```
65531
```

```
65530
```

65529
65528
65527
65526
65525

>>

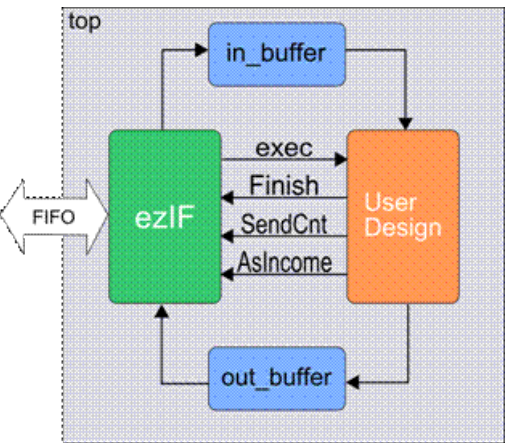
ezIF Interface

Introduce

Interface Description: (example\ Matlab\ezIF)

The ezIF is an interface to isolate users' design and FIFO interface. It can simplify user's design flow. The data received from software will be written in the memory by the ezIF. After ezIF writes all data in the memory, it will send a signal to info the User Design starts to work. After the User Design finished the process, it needs to send a signal to info the ezIF and a signal to tell whether to send data back to software side or not. The ezIF will automatically send user data back to the software side. The integration of ezIF and user design is shown as follow picture.

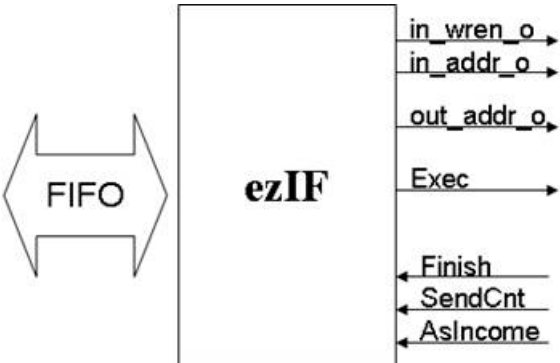
The ezIF is just an example to help user the combine the design with SMIMS development interface. If user needs more efficient or flexible operation mode, please design another mechanism to replace the ezIF.



Interface Operation

The interface operation of ezIF shows as follow diagram. The FIFO side is skipped. Here just introduces

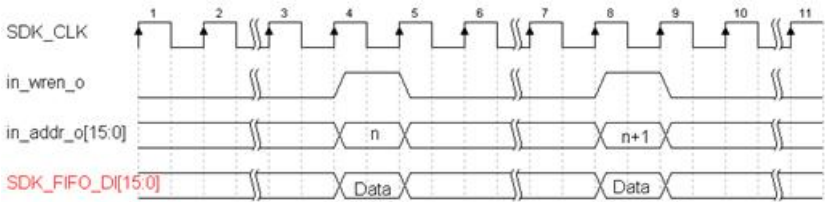
the right side signals.



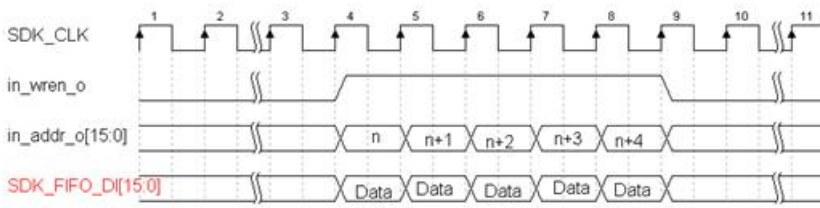
Signal Name	IO Type	Functional description
in_wren_o	output	Memory writing signal. 1 means to write data to the memory.
in_addr_o	output	Memory writing address
out_addr_o	output	Memory reading address for sending data to PC side.
Exec	output	Starts the user design
Finish	input	Info the user design finished the process.
SendCnt	input	The data count of sending data back to PC.
AsIncome	input	If the sending back data has the same counts as received data, then set this signal be 1.

Timing

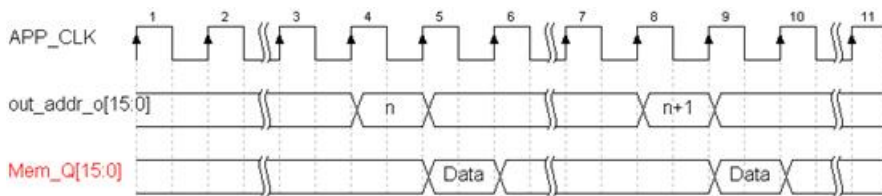
- ezIF writes single data to the memory (the data source is SDK_FIFO_DI of the FIFO).



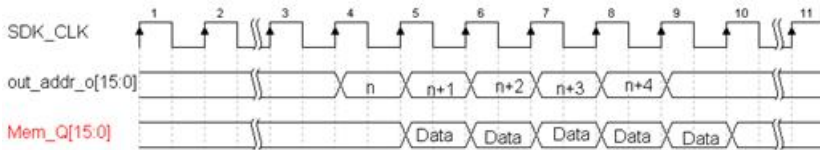
- ezIF writes burst data to the memory (the data source is SDK_FIFO_DI of the FIFO).



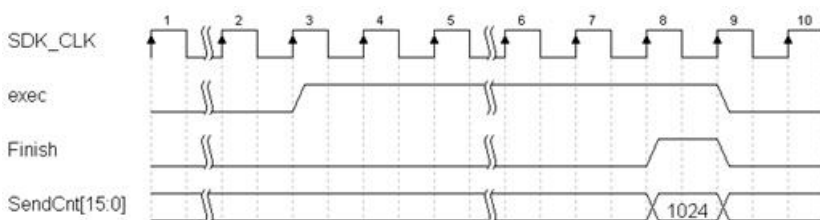
- ezIF reads single data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_FIFO_DO).



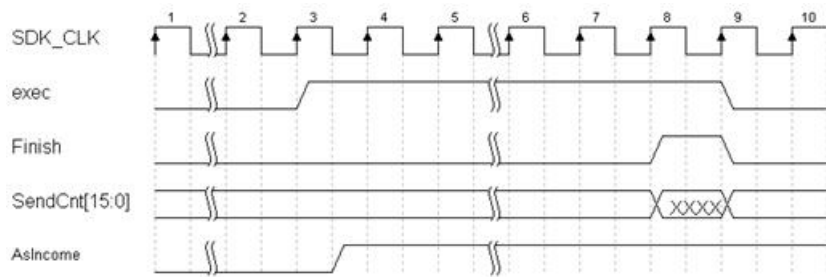
- ezIF reads burst data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_FIFO_DO).



ezIF After the ezIF finished the receiving date, it will keep the Exec signal to be 1. The user design receives this signal will start to operate until the circuit finished the process, it will send out a Finish signal. Thus the Exec signal will be 0. When the user design sends out the Finish signal, the SendCnt should be set at the same time with how many data counts needs to send back to the software side. Therefore, the ezIF will follow the signals to send the data back.



If the sending back data has the same counts as received data, then set this signal be 1. In the mean time, the SendCnt will be ignored.



Software Control Interface

In the software side, user needs to set up the data counts before sending data. Before setting the data counts, the APP_CH of FIFO needs to be 1. After setting the data counts, set the APP_CH back to 0. The source code shown as follows. The important is that the settings needs a lot of time. Only if every time sends different data counts, otherwise please do not set it before every sending. The ideal way is setting once and then sends data continuously, until the data count is different then setup the new data count.

```
smMatVEXSP6(0, 'ChSel', 1);
Cmd = data counts
smMatVEXSP6(0, 'Write', Cmd,1);
smMatVEXSP6(0, 'ChSel', 0);
```

Example Code

```
% Test run SMIMS VEX SP6 SDK
close all;
clear all;
```

```
rand('seed',0);
```

```
% Open FPGA Board and set serial number.
```

```
smMatVEXSP6(0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx', 0);
```

```
for i=1:4095
```

```
    smMatVEXSP6(0, 'ChSel', 1);
```

```
    Cmd = i;
```

```
    smMatVEXSP6(0, 'Write', Cmd,1);
```

```
    smMatVEXSP6(0, 'ChSel', 0);
```

```
    fprintf('Send to HW Data Count : %d    ',i);
```

```
    flag = 1;
```

```
    // generate the random data
```

```
    WriteBuffer=rem(round(4096*rand(1,i)),4096);
```

```
    // send data to the circuit
```

```
    smMatVEXSP6(0, 'Write', WriteBuffer,i);
```

```
    // receive data from the circuit
```

```
    [ret ReadBuffer] = smMatVEXSP6(0, 'Read', i);
```

```
    // check the operation result
```

```
    for j=1:i
```

```
        if ReadBuffer(j) ~= (4096-WriteBuffer(j))
```

```
            flag = 0;
```

```
        end
```

```
    end
```

```
    if(flag == 1)
```

```
        fprintf('Check ok...\n');
```

```
    else
```

```
        fprintf('Error !\n');
```

```
        break;
```

```
    end
```

```
end
```

```
smMatVEXSP6(0, 'Close');
```


SoC-S150 USB

SoC-S150 USB in Matlab command window

Support Models:
SoC-S150-USB-STD
SoC-S150-USB-EDU-STD
Target FPGA : Xilinx XC6SLX150

Requirement

- 1. SMIMS SoC-S150 FPGA hardware device.
- 2. Matlab®/Simulink® of Mathwork®
- 3. Xilinx ISE design software.

VeriLink Function Introduction

This section introduces the way of the Matlab program to communicate with the user circuit in the FPGA by veriLink functions. After installed VeriLink in the PC, Matlab program is able to use VeriLink function.





SMIMS Engine VeriLink function descriptions:
Since the VeriLink function supports up to 2 same type SMIMS FPGA boards in a single system, the 1st parameter of those functions indicates the selected board to operate. The board ID of the first plug-in board is 0.

Function Name	Function description	
smMatSoCS150(iBoardID, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx', ClkMode)	Initial the SMIMS Engine. Return true if success, false vice versa. The first input parameter is the board ID. The second parameter is the serial number. The third one is FPGA clock operation mode, the clock mode and clock speed shows below.	
	ClkMode	
	Value	Value
	0	48 MHz
	1	User offer clock input(Refer to the SoCS150_Manual.pdf)

<code>smMatSoCS150(iBoardID,'Close')</code>	Close the SMIMS Engine, Return true if success, false vice versa.
<code>[retvalue ReadBuf] = smMatSoCS150(iBoardID,'Read', ReadCount)</code>	Read data from hardware. The return value <code>ReadBuf</code> is buffer to store data. The third parameter <code>ReadCount</code> is total data count to read. If the operation is successful, the return value <code>retvalue</code> will be true, false vice versa.
<code>smMatSoCS150(iBoardID,'Write', WriteBuf, WriteCount)</code>	Send fata to hardware. The third parameter <code>WriteBuf</code> is data buffer to send and the data type is unit 16. The fourth parameter <code>WriteCount</code> is total data count to Send. If the operation is successful, this function will return true, false vice versa.
<code>smMatSoCS150(iBoardID,'ChSel', Channel)</code>	Set 8 bits user-define data. Note: If the function is called without FIFO data empty, this function will be block and <code>SDK_CH[7:0]</code> will not changed immediately. <code>SDK_CH[7:0]</code> will be altered after all data in FIFO buffer is read out. The third parameter <code>Channel</code> is the setting value.
<code>smMatSoCS150(iBoardID,'Program', 'BitFilePath')</code>	Program first FPGA. The third parameter <code>'BitFilePath'</code> is the FPGA programming binary file.
<code>smMatSoCS150(iBoardID,'ProgramFPGA2', 'BitFilePath')</code>	Program second FPGA. The third parameter <code>'BitFilePath'</code> is the FPGA programming binary file.
<code>smMatSoCS150(iBoardID,'GetErrMsg')</code>	Retrieve SMIMS Engine Error Message, if an error occurs.
<code>smMatSoCS150('H')</code>	Print the help message in the Matlab Command Window.

Hardware Interface

The hardware interface for user's design in FPGA is used the FIFO interface. The signals and descriptions of hardware interface are listed as the following table.

Signal Name	Signal Type	Function Description
<code>SDK_FIFO_CLK</code>	In → 	<code>ClkMode</code> =0, 48 MHz Clock Source for VEX platform.
<code>SDK_CLK_OUT</code>	← Out 	<code>ClkMode</code> =1, User offer clock for SMIMS Engine.
<code>SDK_RSTN</code>	In → 	After programming a circuit into FPGA, the signal will be set to 0 . When <code>'Open'</code> command is called, the signal will become 1
<code>SDK_CH[7:0]</code>	In → 	User can use this 8 bits channel for sending signal to the user Circuit.

SDK_FIFO_RD	Out ← User Circuit	User design invokes this signal to read FIFO data from SMIMS engine.
SDK_FIFO_WR	Out ← User Circuit	User design invokes this signal to write data to SMIMS engine FIFO buffer.
SDK_FIFO_DI[15:0]	In → User Circuit	Data bus from SMIMS Engine FIFO.
SDK_FIFO_DO[15:0]	Out ← User Circuit	Data bus from user design to be written to SMIMS Engine FIFO buffer.
SDK_FIFO_Full	In → User Circuit	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is full and it can not be written any data more.
SDK_FIFO_Empty	In → User Circuit	When this signal is 1, SMIMS Engine FIFO buffer for user design is empty. There is no data to be read.
SDK_FIFO_AlmostFull	In → User Circuit	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is only one space left.
SDK_FIFO_AlmostEmpty	In → User Circuit	When this signal is 1, SMIMS Engine FIFO buffer for user design is only one data available.

VeriEnterprise Xilinx USB in VeriLink function mode provides the 16bits data bus. Because the data transmit as FIFO, if users want to use a SRAM, they should generate their own address for the SRAM.

There are two 1024 x 16 bits FIFO inside the SMIMS Engine. In the example "Inverter", if write a data size bigger than the FIFO buffer size of the SMIMS Engine, the program will be blocked in the function `smMatSoCS150(0, 'Write', WriteBuf, WriteCount)`. The picture shows this situation below.

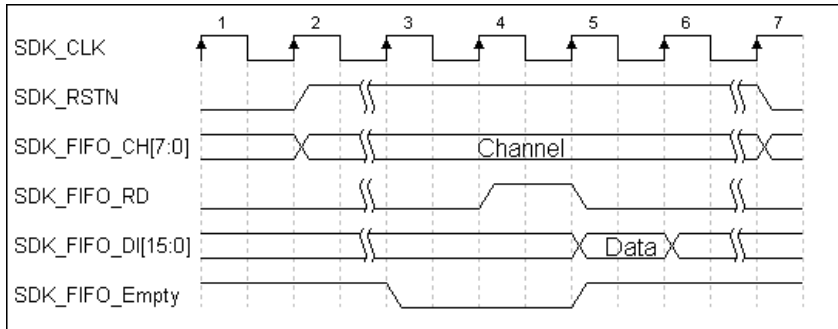


Timing

■ Write Single Data from PC to FPGA:

VeriLink Function : `smMatSoCS150(0, 'Write', WriteBuf, 1)`

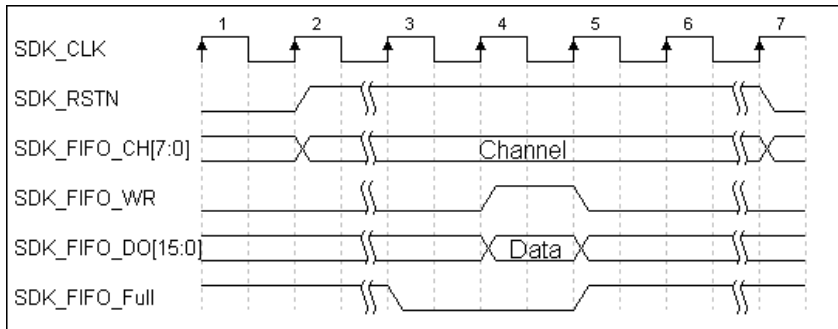
The timing diagram is shown as follows. When SDK_FIFO_Empty signal is low and SDK_FIFO_RD activates for one clock period, the data will be read in the next clock period.



■ Read Single Date from FPGA to PC:

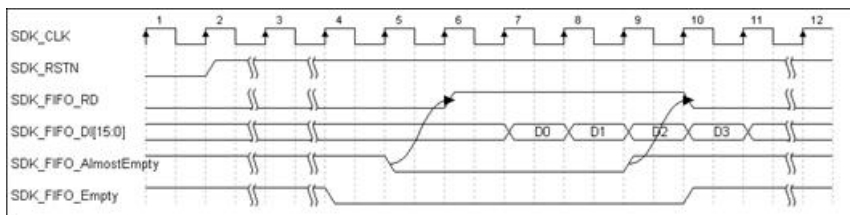
VeriLink Function : [smMatSoCS150\(0, 'Read', 1\)](#)

The timing diagram is shown as follows. When SDK_FIFO_Full signal is low, sends out the SDK_FIFO_WR and SDK_FIFO_DO data in the same time.



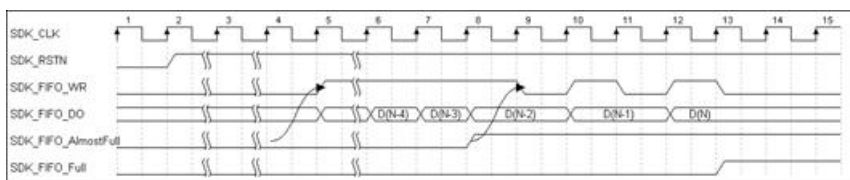
■ Write Multi Data from PC to FPGA:

VeriLink Function : [smMatSoCS150 \(0, 'Write', WriteBuf, 4\)](#)



■ Read Multi Date from FPGA to PC:

VeriLink Function : [smMatSoCS150 \(0, 'Read', 5\)](#)



PIN Map Information

(XC6SLX150)

Signal Name	Pin Name		Signal Name	Pin Name	
SDK_CLK	C15	Input	SDK_CLK_OUT	F7	Output
SDK_RSTN	C13	Input	SDK_CH[0]	AD11	Input
SDK_FIFO_WR	AA12	Output	SDK_CH[1]	Y13	Input
SDK_FIFO_RD	AA15	Output	SDK_CH[2]	AD13	Input
SDK_FIFO_Full	AA14	Input	SDK_CH[3]	AF12	Input
SDK_FIFO_Empty	C18	Input	SDK_CH[4]	W13	Input
SDK_FIFO_AlmostFull	Y15	Input	SDK_CH[5]	AC13	Input
SDK_FIFO_AlmostEmpty	Y14	Input	SDK_CH[6]	AD12	Input
SDK_FIFO_Interrupt	AC12	Output	SDK_CH[7]	H16	Input
SDK_FIFO_DI [0]	Y18	Input	SDK_FIFO_DO [0]	AC14	Output
SDK_FIFO_DI [1]	AA19	Input	SDK_FIFO_DO [1]	AC17	Output
SDK_FIFO_DI [2]	AF21	Input	SDK_FIFO_DO [2]	AB15	Output
SDK_FIFO_DI [3]	AC19	Input	SDK_FIFO_DO [3]	AE15	Output
SDK_FIFO_DI [4]	AD19	Input	SDK_FIFO_DO [4]	AB17	Output
SDK_FIFO_DI [5]	AF20	Input	SDK_FIFO_DO [5]	AD15	Output
SDK_FIFO_DI [6]	AE19	Input	SDK_FIFO_DO [6]	AD17	Output
SDK_FIFO_DI [7]	AD21	Input	SDK_FIFO_DO [7]	V15	Output
SDK_FIFO_DI [8]	A22	Input	SDK_FIFO_DO [8]	W16;	Output
SDK_FIFO_DI [9]	AE21	Input	SDK_FIFO_DO [9]	AF16	Output
SDK_FIFO_DI [10]	A17	Input	SDK_FIFO_DO [10]	AC15	Output
SDK_FIFO_DI [11]	C14	Input	SDK_FIFO_DO [11]	AC16	Output
SDK_FIFO_DI [12]	K15	Input	SDK_FIFO_DO [12]	AF15	Output
SDK_FIFO_DI [13]	E20	Input	SDK_FIFO_DO [13]	AF17	Output
SDK_FIFO_DI [14]	E18	Input	SDK_FIFO_DO [14]	AE5	Output
SDK_FIFO_DI [15]	C11	Input	SDK_FIFO_DO [15]	AF6	Output

Note : Please change the setting of Pin I/O Std. to be LVCMOS33.

Example

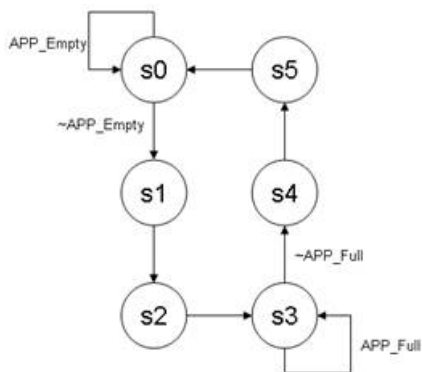
Example Inverter

Example Description: (example\Inverter)

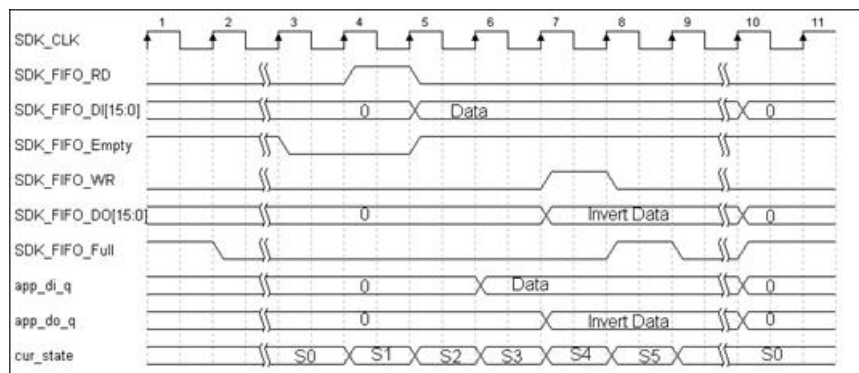
This example, Inverter, show how to use the VeriLink Function to design the Matlab program working with user’s circuit. There are hardware and software two parts.

Hardware

The hardware design (single read/write) is to read data from Input FIFO, pass through an inverter, and then output the data to the Output FIFO. Here use a FSM (Finite state machine) to implement it. The state diagram is shown as follows.



The detailed timing diagram is shown in the following figure.



Software

The software part runs the script file in Matlab. This example uses the least SMIMS function to explain how to use the function. It is a simple and quick introduction. The software source code Matlab script file is TestSoCS150SDK.m

Software program explanation (TestSoCS150SDK.m)

```
% Test run SMIMS SoCS150 SDK
```

```
Clear
```

```

%Program .bit file tp FPGA
smMatSoCS150(0, 'Program','c:\app_example.bit');

% Open FPGA Board and set serial number.
smMatSOoCS150(0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx', 0);

for i=1:10
    writebuf(i)=uint16(i);
end

%Write to FPGA Board
smMatSoCS150(0, 'Write', writebuf, 10);

%Read From FPGA Board
[ret readbuf] = smMatSoCS150(0, 'Read', 10);
disp(readbuf);

%close FPGA Board
smMatSoCS150(0, 'Close');

```

Run and see the result as follows.

This circuit does invert, so the input data 1 to 10 will get the result 65534 to 65525.

```

>> TestSoCS150SDK
    65534
    65533
    65532
    65531
    65530
    65529
    65528
    65527
    65526
    65525

```

```

>>

```

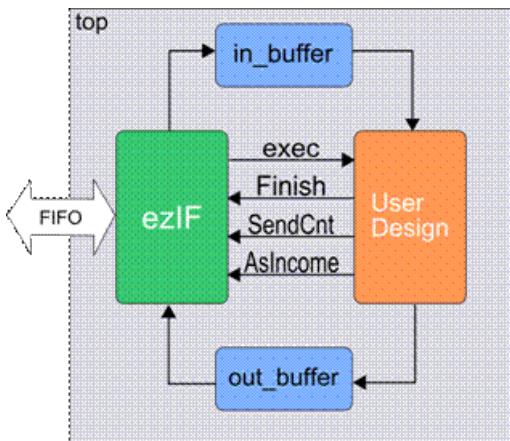
ezIF Interface

Introduce

Interface Description: (example\ Matlab\ezIF)

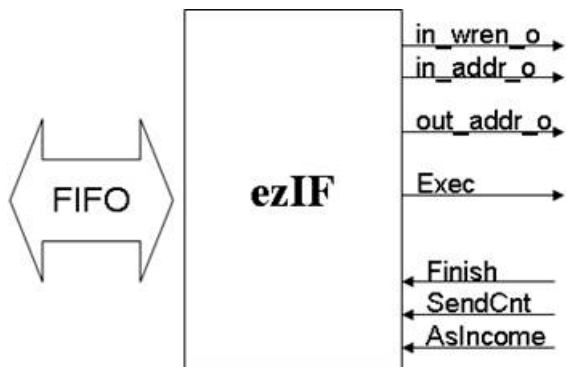
The ezIF is an interface to isolate users' design and FIFO interface. It can simplify user's design flow. The data received from software will be written in the memory by the ezIF. After ezIF writes all data in the memory, it will send a signal to info the User Design starts to work. After the User Design finished the process, it needs to send a signal to info the ezIF and a signal to tell whether to send data back to software side or not. The ezIF will automatically send user data back to the software side. The integration of ezIF and user design is shown as follow picture.

The ezIF is just an example to help user the combine the design with SMIMS development interface. If user needs more efficient or flexible operation mode, please design another mechanism to replace the ezIF.



Interface Operation

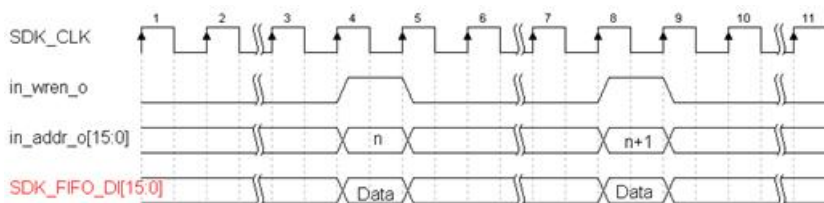
The interface operation of ezIF shows as follow diagram. The FIFO side is skipped. Here just introduces the right side signals.



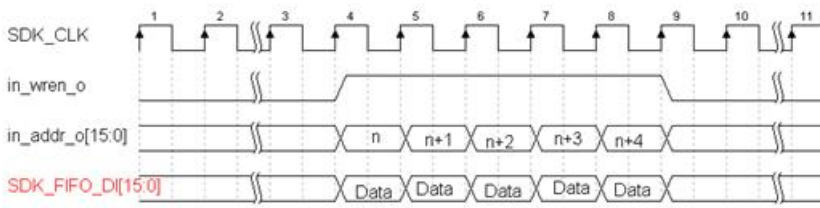
Signal Name	IO Type	Functional description
<code>in_wren_o</code>	output	Memory writing signal. 1 means to write data to the memory.
<code>in_addr_o</code>	output	Memory writing address
<code>out_addr_o</code>	output	Memory reading address for sending data to PC side.
<code>Exec</code>	output	Starts the user design
<code>Finish</code>	input	Info the user design finished the process.
<code>SendCnt</code>	input	The data count of sending data back to PC.
<code>AslIncome</code>	input	If the sending back data has the same counts as received data, then set this signal be 1.

Timing

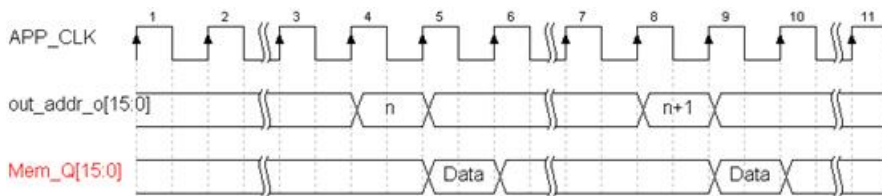
- ezIF writes single data to the memory (the data source is SDK_FIFO_DI of the FIFO).



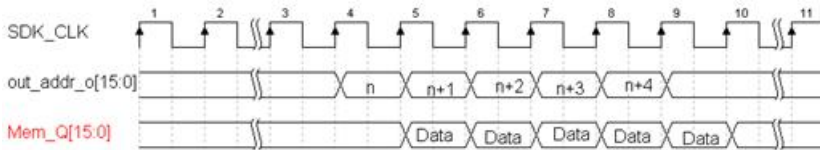
- ezIF writes burst data to the memory (the data source is SDK_FIFO_DI of the FIFO).



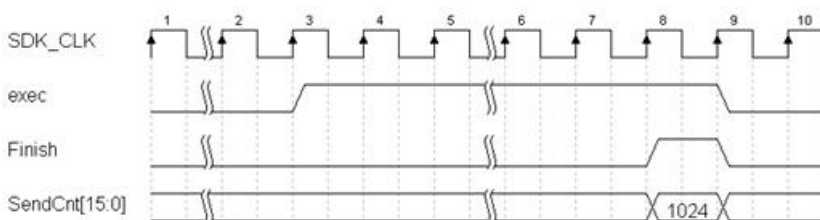
- ezIF reads single data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_FIFO_DO).



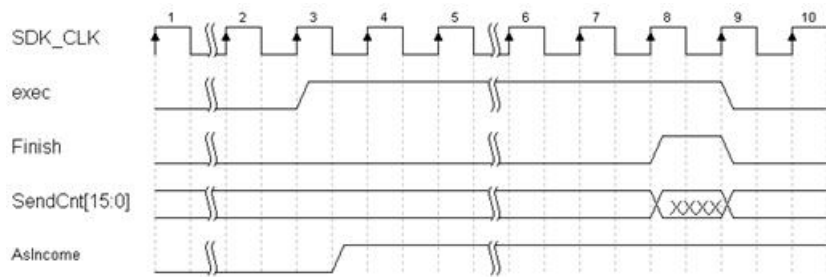
- ezIF reads burst data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_FIFO_DO).



ezIF After the ezIF finished the receiving date, it will keep the Exec signal to be 1. The user design receives this signal will start to operate until the circuit finished the process, it will send out a Finish signal. Thus the Exec signal will be 0. When the user design sends out the Finish signal, the SendCnt should be set at the same time with how many data counts needs to send back to the software side. Therefore, the ezIF will follow the signals to send the data back.



If the sending back data has the same counts as received data, then set this signal be 1. In the mean time, the SendCnt will be ignored.



Software Control Interface

In the software side, user needs to set up the data counts before sending data. Before setting the data counts, the APP_CH of FIFO needs to be 1. After setting the data counts, set the APP_CH back to 0. The source code shown as follows. The important is that the settings needs a lot of time. Only if every time sends different data counts, otherwise please do not set it before every sending. The ideal way is setting once and then sends data continuously, until the data count is different then setup the new data count.

```
smMatSoCS150(0, 'ChSel', 1);
Cmd = data counts
smMatSoCS150(0, 'Write', Cmd,1);
smMatSoCS150(0, 'ChSel', 0);
```

Example Code

```
% Test run SMIMS SoCS150 SDK
close all;
clear all;
```

```
rand('seed',0);
```

```
% Open FPGA Board and set serial number.
```

```
smMatSoCS150(0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx', 0);
```

```
for i=1:4095
```

```
    smMatSoCS150(0, 'ChSel', 1);
```

```
    Cmd = i;
```

```
    smMatSoCS150(0, 'Write', Cmd,1);
```

```
    smMatSoCS150(0, 'ChSel', 0);
```

```
    fprintf('Send to HW Data Count : %d    ',i);
```

```
    flag = 1;
```

```
    // generate the random data
```

```
    WriteBuffer=rem(round(4096*rand(1,i)),4096);
```

```
    // send data to the circuit
```

```
    smMatSoCS150(0, 'Write', WriteBuffer,i);
```

```
    // receive data from the circuit
```

```
    [ret ReadBuffer] = smMatSoCS150(0, 'Read', i);
```

```
    // check the operation result
```

```
    for j=1:i
```

```
        if ReadBuffer(j) ~= (4096-WriteBuffer(j))
```

```
            flag = 0;
```

```
        end
```

```
    end
```

```
    if(flag == 1)
```

```
        fprintf('Check ok...\n');
```

```
    else
```

```
        fprintf('Error !\n');
```

```
        break;
```

```
    end
```

```
end
```

```
smMatSoCS150(0, 'Close');
```

SoCV330 USB

SoCV330 USB in Matlab command window

Support Models:
SoCV330-USB-STD
SoCV330-USB-EDU-STD
Target FPGA : Xilinx XC5VLX330

Requirement

- 1. SMIMS SoCV330 FPGA hardware device.
- 2. Matlab®/Simulink® of Mathwork®
- 3. Xilinx ISE design software.

VeriLink Function Introduction

This section introduces the way of the Matlab program to communicate with the user circuit in the FPGA by veriLink functions. After installed VeriLink in the PC, Matlab program is able to use VeriLink function.




SMIMS Engine VeriLink function descriptions:
Since the VeriLink function supports up to 2 same type SMIMS FPGA boards in a single system, the 1st parameter of those functions indicates the selected board to operate. The board ID of the first plug-in board is 0.










Function Name	Function description	
smMatSoCV5(iBoardID, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx', ClkMode)	Initial the SMIMS Engine. Return true if success, false vice versa. The first input parameter is the board ID. The second parameter is the serial number. The third one is FPGA clock operation mode, the clock mode and clock speed shows below.	
	ClkMode	
	Value	Value
	0	48 MHz
	1	User offer clock input(Refer to the

	VEXT2_Manual.pdf)
<code>smMatSoCV5(iBoardID,'Close')</code>	Close the SMIMS Engine, Return true if success, false vice versa.
<code>[retval ReadBuf] = smMatSoCV5(iBoardID,'Read', ReadCount)</code>	Read data from hardware. The return value <code>ReadBuf</code> is buffer to store data. The third parameter <code>ReadCount</code> is total data count to read. If the operation is successful, the return value <code>retval</code> will be true, false vice versa.
<code>smMatSoCV5(iBoardID,'Write', WriteBuf, WriteCount)</code>	Send data to hardware. The third parameter <code>WriteBuf</code> is data buffer to send and the data type is unit 16. The fourth parameter <code>WriteCount</code> is total data count to Send. If the operation is successful, this function will return true, false vice versa.
<code>smMatSoCV5(iBoardID,'ChSel', Channel)</code>	Set 8 bits user-define data. Note: If the function is called without FIFO data empty, this function will be block and <code>SDK_CH[7:0]</code> will not changed immediately. <code>SDK_CH[7:0]</code> will be altered after all data in FIFO buffer is read out. The third parameter <code>Channel</code> is the setting value.
<code>smMatSoCV5(iBoardID,'Program', 'BitFilePath')</code>	Program first FPGA. The third parameter <code>'BitFilePath'</code> is the FPGA programming binary file.
<code>smMatSoCV5(iBoardID,'ProgramFPGA2', 'BitFilePath')</code>	Program second FPGA. The third parameter <code>'BitFilePath'</code> is the FPGA programming binary file.
<code>smMatSoCV5(iBoardID,'GetErrMsg')</code>	Retrieve SMIMS Engine Error Message, if an error occurs.
<code>smMatSoCV5('H')</code>	Print the help message in the Matlab Command Window.

Hardware Interface

The hardware interface for user's design in FPGA is used the FIFO interface. The signals and descriptions of hardware interface are listed as the following table.

Signal Name	Signal Type	Function Description
SDK_FIFO_CLK	In → 	<code>ClkMode</code> =0, 48 MHz Clock Source for VEX platform.
SDK_CLK_OUT	← Out 	<code>ClkMode</code> =1, User offer clock for SMIMS Engine.
SDK_RSTN	In → 	After programming a circuit into FPGA, the signal will be set to 0 . When <code>'Open'</code> command is called,

		the signal will become 1
SDK_CH[7:0]	In → 	User can use this 8 bits channel for sending signal to the user Circuit.
SDK_FIFO_RD	← Out 	User design invokes this signal to read FIFO data from SMIMS engine.
SDK_FIFO_WR	← Out 	User design invokes this signal to write data to SMIMS engine FIFO buffer.
SDK_FIFO_DI[15:0]	In → 	Data bus from SMIMS Engine FIFO.
SDK_FIFO_DO[15:0]	← Out 	Data bus from user design to be written to SMIMS Engine FIFO buffer.
SDK_FIFO_Full	In → 	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is full and it can not be written any data more.
SDK_FIFO_Empty	In → 	When this signal is 1, SMIMS Engine FIFO buffer for user design is empty. There is no data to be read.
SDK_FIFO_AlmostFull	In → 	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is only one space left.
SDK_FIFO_AlmostEmpty	In → 	When this signal is 1, SMIMS Engine FIFO buffer for user design is only one data available.

VeriEnterprise Xilinx USB in VeriLink function mode provides the 16bits data bus. Because the data transmit as FIFO, if users want to use a SRAM, they should generate their own address for the SRAM.

There are two 1024 x 16 bits FIFO inside the SMIMS Engine. In the example “Inverter”, if write a data size bigger than the FIFO buffer size of the SMIMS Engine, the program will be blocked in the function `smMatSoCV5(0, 'Write', WriteBuf, WriteCount)`. The picture shows this situation below.

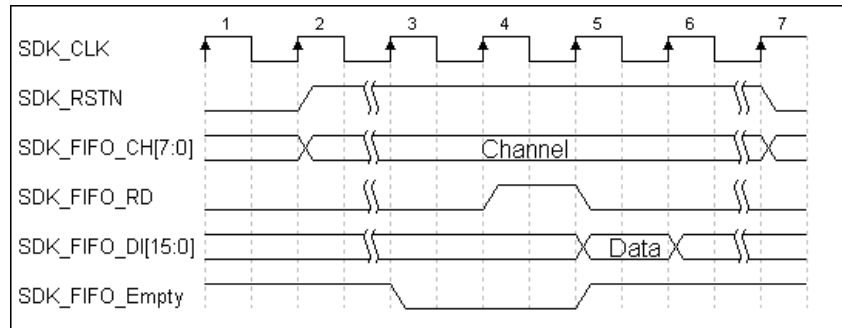


Timing

■ Write Single Data from PC to FPGA:

VeriLink Function : [smMatSoCV5\(0, 'Write', WriteBuf, 1\)](#)

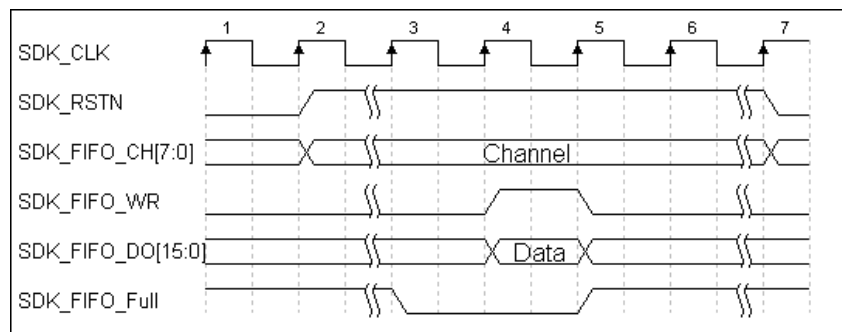
The timing diagram is shown as follows. When SDK_FIFO_Empty signal is low and SDK_FIFO_RD activates for one clock period, the data will be read in the next clock period.



■ Read Single Date from FPGA to PC:

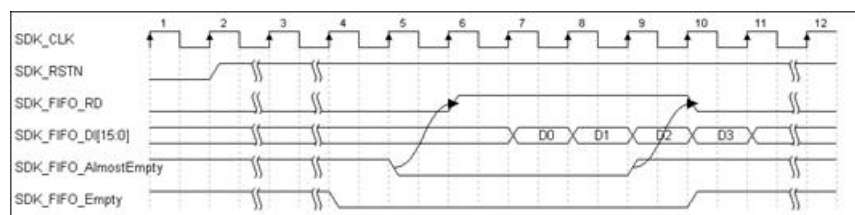
VeriLink Function : [smMatSoCV5\(0, 'Read', 1\)](#)

The timing diagram is shown as follows. When SDK_FIFO_Full signal is low, sends out the SDK_FIFO_WR and SDK_FIFO_DO data in the same time.



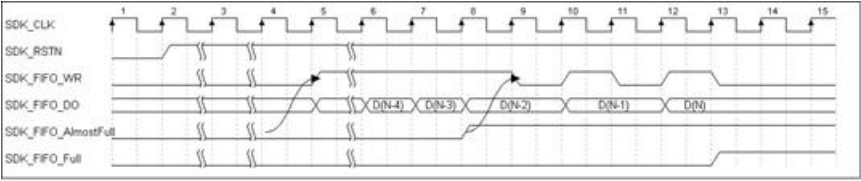
■ Write Multi Data from PC to FPGA:

VeriLink Function : [smMatSoCV5 \(0, 'Write', WriteBuf, 4\)](#)



■ Read Multi Date from FPGA to PC:

VeriLink Function : [smMatSoCV5 \(0, 'Read', 5\)](#)



PIN Map Information

(XC5VLX330)

Signal Name	Pin Name		Signal Name	Pin Name	
SDK_CLK	L15	Input	SDK_CLK_OUT	J7	Output
SDK_RSTN	K10	Input	SDK_CH[0]	AJ8	Input
SDK_FIFO_WR	AE9	Output	SDK_CH[1]	AE10	Input
SDK_FIFO_RD	AT9	Output	SDK_CH[2]	AE8	Input
SDK_FIFO_Full	AR8	Input	SDK_CH[3]	AF7	Input
SDK_FIFO_Empty	N15	Input	SDK_CH[4]	AF10	Input
SDK_FIFO_AlmostFull	AV10	Input	SDK_CH[5]	AG8	Input
SDK_FIFO_AlmostEmpty	AU9	Input	SDK_CH[6]	N11	Input
SDK_FIFO_Interrupt	AF9	Output	SDK_CH[7]	L14	Input
SDK_FIFO_DI [0]	AJ13	Input	SDK_FIFO_DO [0]	AV9	Output
SDK_FIFO_DI [1]	AK12	Input	SDK_FIFO_DO [1]	AR10	Output
SDK_FIFO_DI [2]	AH16	Input	SDK_FIFO_DO [2]	AU13	Output
SDK_FIFO_DI [3]	AJ12	Input	SDK_FIFO_DO [3]	AP11	Output
SDK_FIFO_DI [4]	AJ11	Input	SDK_FIFO_DO [4]	AR12	Output
SDK_FIFO_DI [5]	AL14	Input	SDK_FIFO_DO [5]	AT11	Output
SDK_FIFO_DI [6]	AL15	Input	SDK_FIFO_DO [6]	AU11	Output
SDK_FIFO_DI [7]	AK13	Input	SDK_FIFO_DO [7]	AV11	Output
SDK_FIFO_DI [8]	AJ16	Input	SDK_FIFO_DO [8]	AU12	Output
SDK_FIFO_DI [9]	AJ15	Input	SDK_FIFO_DO [9]	AP12	Output
SDK_FIFO_DI [10]	M13	Input	SDK_FIFO_DO [10]	AR9	Output
SDK_FIFO_DI [11]	P10	Input	SDK_FIFO_DO [11]	AT10	Output
SDK_FIFO_DI [12]	P15	Input	SDK_FIFO_DO [12]	AT12	Output
SDK_FIFO_DI [13]	M16	Input	SDK_FIFO_DO [13]	AN9	Output
SDK_FIFO_DI [14]	N16	Input	SDK_FIFO_DO [14]	AT5	Output
SDK_FIFO_DI [15]	G8	Input	SDK_FIFO_DO [15]	AT6	Output

Note : Please change the setting of Pin I/O Std. to be LVCMOS33.

Example

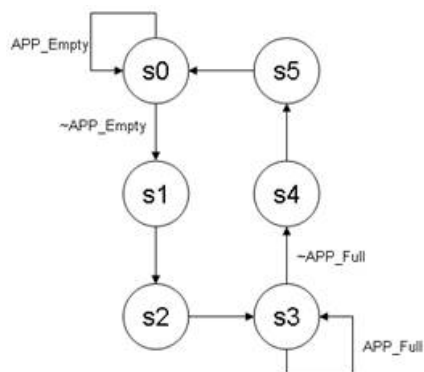
Example Inverter

Example Description: (example\Inverter)

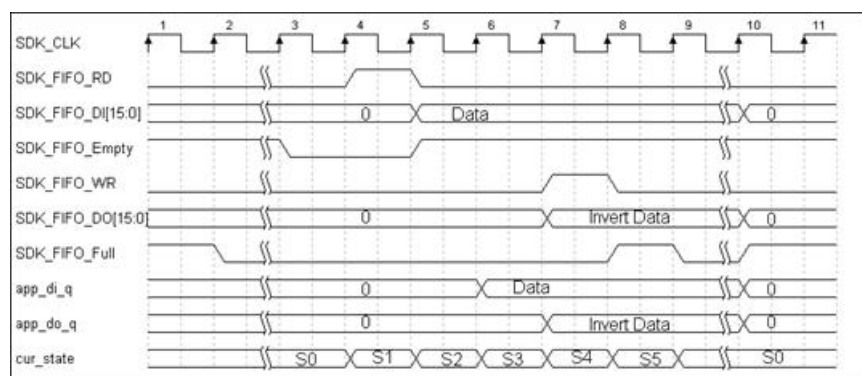
This example, Inverter, show how to use the VeriLink Function to design the Matlab program working with user's circuit. There are hardware and software two parts.

Hardware

The hardware design (single read/write) is to read data from Input FIFO, pass through an inverter, and then output the data to the Output FIFO. Here use a FSM (Finite state machine) to implement it. The state diagram is shown as follows.



The detailed timing diagram is shown in the following figure.



Software

The software part runs the script file in Matlab. This example uses the least SMIMS function to explain how to use the function. It is a simple and quick introduction. The software source code Matlab script file is TestSoCV330SDK.m

Software program explanation (TestSoCV330SDK.m)

```
% Test run SMIMS SoCV330 SDK
Clear

%Program .bit file tp FPGA
smMatSoCV5(0, 'Program','c:\app_example.bit');

% Open FPGA Board and set serial number.
smMatSoCV5 (0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx', 0);

for i=1:10
    writebuf(i)=uint16(i);
end

%Write to FPGA Board
smMatSoCV5(0, 'Write', writebuf, 10);

%Read From FPGA Board
[ret readbuf] = smMatSoCV5(0, 'Read', 10);
disp(readbuf);

%close FPGA Board
smMatSoCV5(0, 'Close');
```

Run and see the result as follows.

This circuit does invert, so the input data 1 to 10 will get the result 65534 to 65525.

```
>> TestSoCV330SDK
65534
65533
65532
65531
65530
65529
65528
65527
65526
```

>>

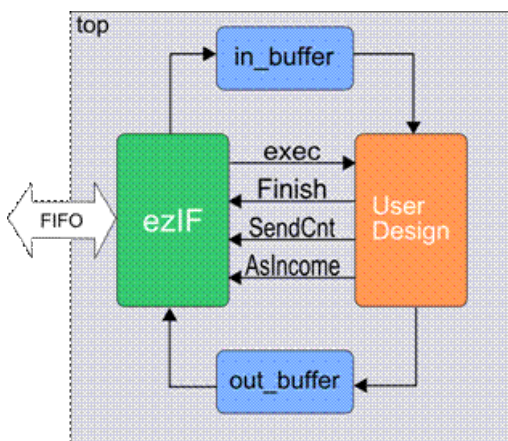
ezIF Interface

Introduce

Interface Description: (example\ Matlab\ezIF)

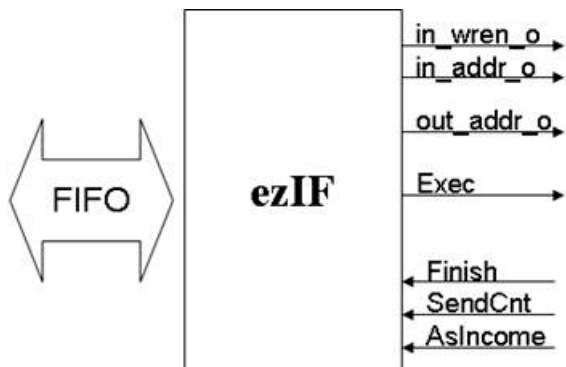
The ezIF is an interface to isolate users' design and FIFO interface. It can simplify user's design flow. The data received from software will be written in the memory by the ezIF. After ezIF writes all data in the memory, it will send a signal to info the User Design starts to work. After the User Design finished the process, it needs to send a signal to info the ezIF and a signal to tell whether to send data back to software side or not. The ezIF will automatically send user data back to the software side. The integration of ezIF and user design is shown as follow picture.

The ezIF is just an example to help user the combine the design with SMIMS development interface. If user needs more efficient or flexible operation mode, please design another mechanism to replace the ezIF.



Interface Operation

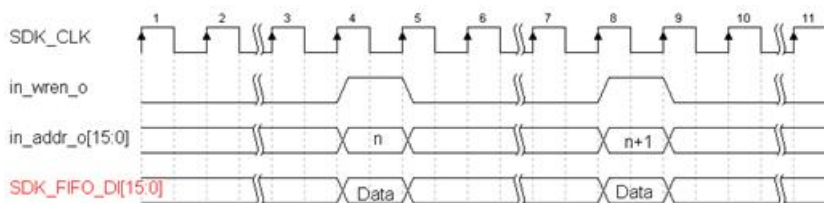
The interface operation of ezIF shows as follow diagram. The FIFO side is skipped. Here just introduces the right side signals.



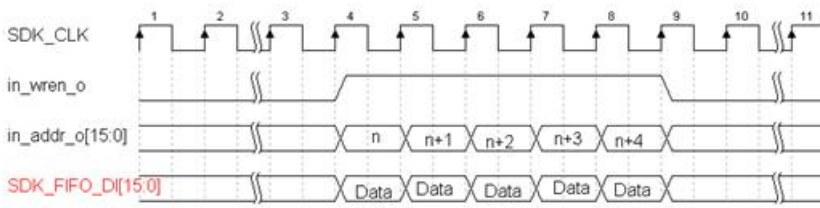
Signal Name	IO Type	Functional description
in_wren_o	output	Memory writing signal. 1 means to write data to the memory.
in_addr_o	output	Memory writing address
out_addr_o	output	Memory reading address for sending data to PC side.
Exec	output	Starts the user design
Finish	input	Info the user design finished the process.
SendCnt	input	The data count of sending data back to PC.
AsIncome	input	If the sending back data has the same counts as received data, then set this signal be 1.

Timing

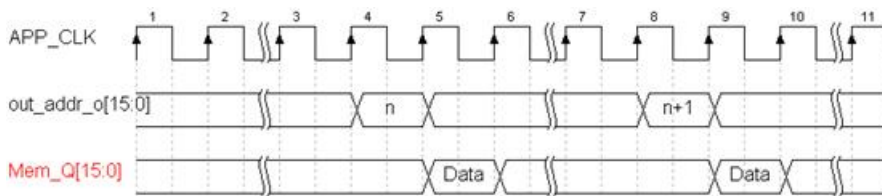
- ezIF writes single data to the memory (the data source is SDK_FIFO_DI of the FIFO).



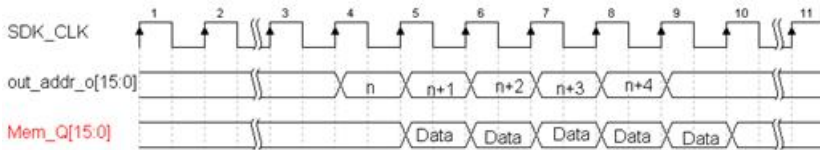
- ezIF writes burst data to the memory (the data source is SDK_FIFO_DI of the FIFO).



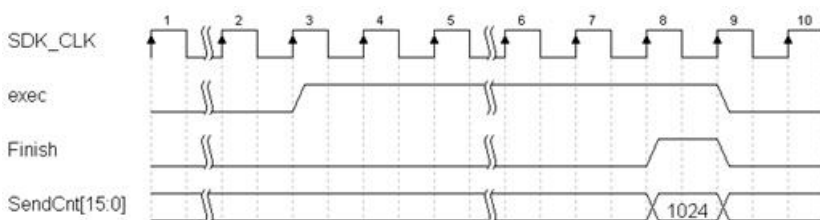
■ ezIF reads single data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_FIFO_DO).



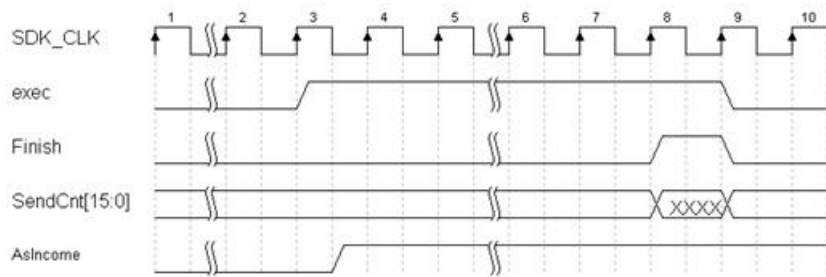
■ ezIF reads burst data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_FIFO_DO).



ezIF After the ezIF finished the receiving date, it will keep the Exec signal to be 1. The user design receives this signal will start to operate until the circuit finished the process, it will send out a Finish signal. Thus the Exec signal will be 0. When the user design sends out the Finish signal, the SendCnt should be set at the same time with how many data counts needs to send back to the software side. Therefore, the ezIF will follow the signals to send the data back.



If the sending back data has the same counts as received data, then set this signal be 1. In the mean time, the SendCnt will be ignored.



Software Control Interface

In the software side, user needs to set up the data counts before sending data. Before setting the data counts, the APP_CH of FIFO needs to be 1. After setting the data counts, set the APP_CH back to 0. The source code shown as follows. The important is that the settings needs a lot of time. Only if every time sends different data counts, otherwise please do not set it before every sending. The ideal way is setting once and then sends data continuously, until the data count is different then setup the new data count.

```
smMatSoCV5(0, 'ChSel', 1);
Cmd = data counts
smMatSoCV5(0, 'Write', Cmd,1);
smMatSoCV5(0, 'ChSel', 0);
```

Example Code

```
% Test run SMIMS SoCV330 SDK
close all;
clear all;
```

```
rand('seed',0);
```

```
% Open FPGA Board and set serial number.
```

```
smMatSoCV5(0, 'Open', 'xxx-xxx-xxx-xxx-xxx-xxx', 0);
```

```
for i=1:4095
```

```
    smMatSoCV5(0, 'ChSel', 1);
```

```
    Cmd = i;
```

```
    smMatSoCV5(0, 'Write', Cmd,1);
```

```
    smMatSoCV5(0, 'ChSel', 0);
```

```
    fprintf('Send to HW Data Count : %d    ',i);
```

```
    flag = 1;
```

```
    // generate the random data
```

```
    WriteBuffer=rem(round(4096*rand(1,i)),4096);
```

```
    // send data to the circuit
```

```
    smMatSoCV5(0, 'Write', WriteBuffer,i);
```

```
    // receive data from the circuit
```

```
    [ret ReadBuffer] = smMatSoCV5(0, 'Read', i);
```

```
    // check the operation result
```

```
    for j=1:i
```

```
        if ReadBuffer(j) ~= (4096-WriteBuffer(j))
```

```
            flag = 0;
```

```
        end
```

```
    end
```

```
    if(flag == 1)
```

```
        fprintf('Check ok...\n');
```

```
    else
```

```
        fprintf('Error !\n');
```

```
        break;
```

```
    end
```

```
end
```

```
smMatSoCV5(0, 'Close');
```


Macube-VEXT2 Xilinx Spartan-6 USB

VEXT2 USB in Matlab command window

Support Models:
VEXT2-USB-STD
VEXT2-USB-EDU-STD
Target FPGA : Xilinx XC6SLX75T 、XC6SLX150T

Requirement

- 1. SMIMS VEXT2 FPGA hardware device.
- 2. Matlab®/Simulink® of Mathwork®
- 3. Xilinx ISE design software.

VeriLink Function Introduction

This section introduces the way of the Matlab program to communicate with the user circuit in the FPGA by veriLink functions. After installed VeriLink in the PC, Matlab program is able to use VeriLink function.

SMIMS Engine VeriLink function descriptions:




Since the VeriLink function supports up to 2 same type SMIMS FPGA boards in a single system, the 1st parameter of those functions indicates the selected board to operate. The board ID of the first plug-in board is 0.










Function Name	Function description	
smMatVEXT2(iBoardID, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx', ClkMode)	Initial the SMIMS Engine. Return true if success, false vice versa. The first input parameter is the board ID. The second parameter is the serial number. The third one is FPGA clock operation mode, the clock mode and clock speed shows below.	
	ClkMode	
	Value	Value
	0	48 MHz
	1	User offer clock input(Refer to the VEXT2_Manual.pdf)

<code>smMatVEXT2(iBoardID,'Close')</code>	Close the SMIMS Engine, Return true if success, false vice versa.
<code>[retvalue ReadBuf] = smMatVEXT2(iBoardID,'Read', ReadCount)</code>	Read data from hardware. The return value <code>ReadBuf</code> is buffer to store data. The third parameter <code>ReadCount</code> is total data count to read. If the operation is successful, the return value <code>retvalue</code> will be true, false vice versa.
<code>smMatVEXT2(iBoardID,'Write', WriteBuf, WriteCount)</code>	Send fata to hardware. The third parameter <code>WriteBuf</code> is data buffer to send and the data type is unit 16. The fourth parameter <code>WriteCount</code> is total data count to Send. If the operation is successful, this function will return true, false vice versa.
<code>smMatVEXT2(iBoardID,'ChSel', Channel)</code>	Set 8 bits user-define data. Note: If the function is called without FIFO data empty, this function will be block and <code>SDK_CH[7:0]</code> will not changed immediately. <code>SDK_CH[7:0]</code> will be altered after all data in FIFO buffer is read out. The third parameter <code>Channel</code> is the setting value.
<code>smMatVEXT2(iBoardID,'Program', 'BitFilePath')</code>	Program first FPGA. The third parameter <code>'BitFilePath'</code> is the FPGA programming binary file.
<code>smMatVEXT2(iBoardID,'ProgramFPGA2', 'BitFilePath')</code>	Program second FPGA. The third parameter <code>'BitFilePath'</code> is the FPGA programming binary file.
<code>smMatVEXT2(iBoardID,'GetErrMsg')</code>	Retrieve SMIMS Engine Error Message, if an error occurs.
<code>smMatVEXT2('H')</code>	Print the help message in the Matlab Command Window.

Hardware Interface

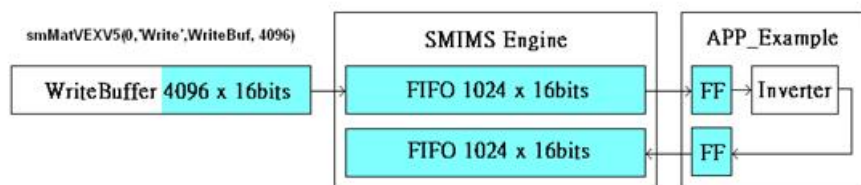
The hardware interface for user's design in FPGA is used the FIFO interface. The signals and descriptions of hardware interface are listed as the following table.

Signal Name	Signal Type	Function Description
SDK_FIFO_CLK	In → 	<code>ClkMode</code> =0, 48 MHz Clock Source for VEX platform.
SDK_CLK_OUT	← Out 	<code>ClkMode</code> =1, User offer clock for SMIMS Engine.
SDK_RSTN	In → 	After programming a circuit into FPGA, the signal will be set to 0 . When <code>'Open'</code> command is called, the signal will become 1

SDK_CH[7:0]	In → 	User can use this 8 bits channel for sending signal to the user Circuit.
SDK_FIFO_RD	← Out 	User design invokes this signal to read FIFO data from SMIMS engine.
SDK_FIFO_WR	← Out 	User design invokes this signal to write data to SMIMS engine FIFO buffer.
SDK_FIFO_DI[15:0]	In → 	Data bus from SMIMS Engine FIFO.
SDK_FIFO_DO[15:0]	← Out 	Data bus from user design to be written to SMIMS Engine FIFO buffer.
SDK_FIFO_Full	In → 	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is full and it can not be written any data more.
SDK_FIFO_Empty	In → 	When this signal is 1, SMIMS Engine FIFO buffer for user design is empty. There is no data to be read.
SDK_FIFO_AlmostFull	In → 	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is only one space left.
SDK_FIFO_AlmostEmpty	In → 	When this signal is 1, SMIMS Engine FIFO buffer for user design is only one data available.

VeriEnterprise Xilinx USB in VeriLink function mode provides the 16bits data bus. Because the data transmit as FIFO, if users want to use a SRAM, they should generate their own address for the SRAM.

There are two 1024 x 16 bits FIFO inside the SMIMS Engine. In the example “Inverter”, if write a data size bigger than the FIFO buffer size of the SMIMS Engine, the program will be blocked in the function `smMatVEXT2(0, 'Write', WriteBuf, WriteCount)`. The picture shows this situation below.

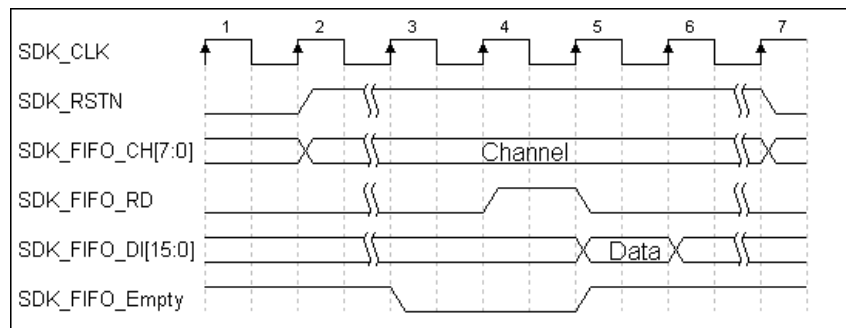


Timing

■ Write Single Data from PC to FPGA:

VeriLink Function : `smMatVEXT2(0, 'Write', WriteBuf, 1)`

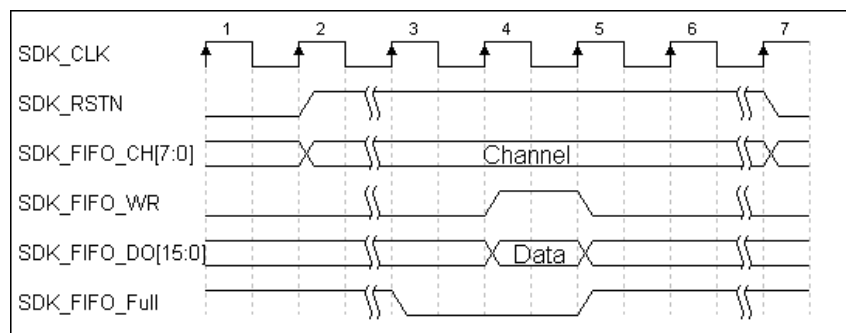
The timing diagram is shown as follows. When SDK_FIFO_Empty signal is low and SDK_FIFO_RD activates for one clock period, the data will be read in the next clock period.



■ Read Single Date from FPGA to PC:

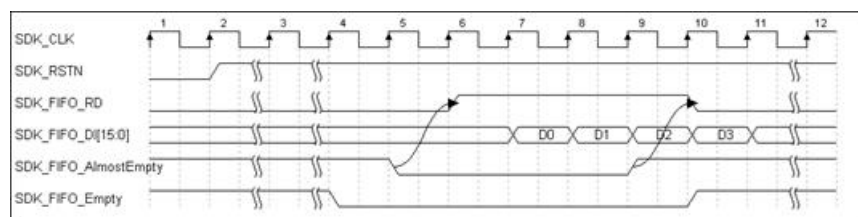
VeriLink Function : [smMatVEXT2\(0, 'Read', 1\)](#)

The timing diagram is shown as follows. When SDK_FIFO_Full signal is low, sends out the SDK_FIFO_WR and SDK_FIFO_DO data in the same time.



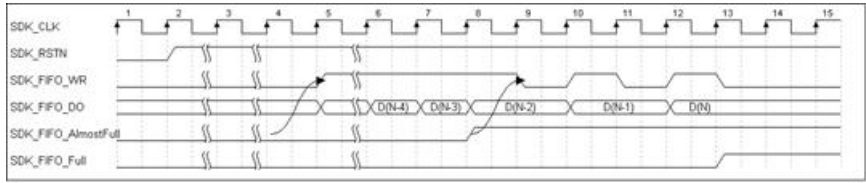
■ Write Multi Data from PC to FPGA:

VeriLink Function : [smMatVEXT2 \(0, 'Write', WriteBuf, 4\)](#)



■ Read Multi Date from FPGA to PC:

VeriLink Function : [smMatVEXT2 \(0, 'Read', 5\)](#)



PIN Map Information

(XC6SLX75T)

Signal Name	Pin Name		Signal Name	Pin Name	
SDK_CLK	AE13	Input	SDK_CLK_OUT	M4	Output
SDK_RSTN	W19	Input	SDK_CH[0]	Y12	Input
SDK_FIFO_WR	AA17	Output	SDK_CH[1]	AA12	Input
SDK_FIFO_RD	Y17	Output	SDK_CH[2]	W14	Input
SDK_FIFO_Full	W17	Input	SDK_CH[3]	Y13	Input
SDK_FIFO_Empty	AA15	Input	SDK_CH[4]	AD14	Input
SDK_FIFO_AlmostFull	W18	Input	SDK_CH[5]	AF14	Input
SDK_FIFO_AlmostEmpty	AB15	Input	SDK_CH[6]	U15	Input
SDK_FIFO_Interrupt	V18	Output	SDK_CH[7]	V16	Input
SDK_FIFO_DI [0]	L10	Input	SDK_FIFO_DO [0]	G4	Output
SDK_FIFO_DI [1]	K10	Input	SDK_FIFO_DO [1]	G3	Output
SDK_FIFO_DI [2]	M3	Input	SDK_FIFO_DO [2]	J2	Output
SDK_FIFO_DI [3]	M1	Input	SDK_FIFO_DO [3]	J1	Output
SDK_FIFO_DI [4]	M8	Input	SDK_FIFO_DO [4]	K5	Output
SDK_FIFO_DI [5]	M6	Input	SDK_FIFO_DO [5]	J5	Output
SDK_FIFO_DI [6]	M10	Input	SDK_FIFO_DO [6]	L2	Output
SDK_FIFO_DI [7]	M9	Input	SDK_FIFO_DO [7]	L1	Output
SDK_FIFO_DI [8]	P3	Input	SDK_FIFO_DO [8]	L9	Output
SDK_FIFO_DI [9]	P1	Input	SDK_FIFO_DO [9]	L8	Output
SDK_FIFO_DI [10]	N6	Input	SDK_FIFO_DO [10]	N2	Output
SDK_FIFO_DI [11]	P6	Input	SDK_FIFO_DO [11]	N1	Output
SDK_FIFO_DI [12]	P10	Input	SDK_FIFO_DO [12]	N5	Output
SDK_FIFO_DI [13]	N9	Input	SDK_FIFO_DO [13]	N4	Output
SDK_FIFO_DI [14]	R2	Input	SDK_FIFO_DO [14]	N8	Output
SDK_FIFO_DI [15]	R1	Input	SDK_FIFO_DO [15]	N7	Output

Note : Please change the setting of Pin I/O Std. to be LVCMOS33.

Example

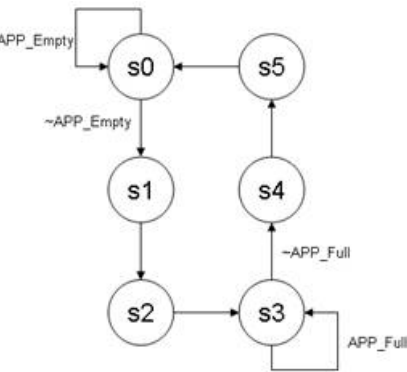
Example Inverter

Example Description: (example\Inverter)

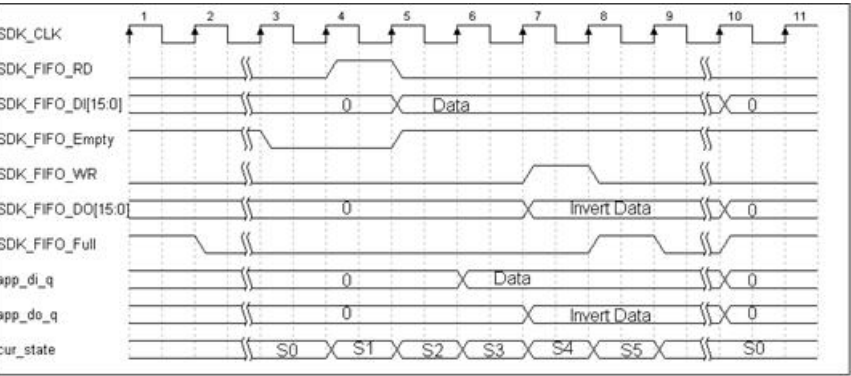
This example, Inverter, show how to use the VeriLink Function to design the Matlab program working with user's circuit. There are hardware and software two parts.

Hardware

The hardware design (single read/write) is to read data from Input FIFO, pass through an inverter, and then output the data to the Output FIFO. Here use a FSM (Finite state machine) to implement it. The state diagram is shown as follows.



The detailed timing diagram is shown in the following figure.



Software

The software part runs the script file in Matlab. This example uses the least SMIMS function to explain how to use the function. It is a simple and quick introduction. The software source code Matlab script file is TestVEXT2SDK.m

Software program explanation (TestVEXT2SDK.m)

```
% Test run SMIMS VEXT2 SDK
```

```
Clear
```

```
%Program .bit file tp FPGA
```

```
smMatVEXT2(0, 'Program','c:\app_example.bit');
```

```
% Open FPGA Board and set serial number.
```

```
smMatVEXT2 (0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx', 0);
```

```
for i=1:10
```

```
    writebuf(i)=uint16(i);
```

```
end
```

```
%Write to FPGA Board
```

```
smMatVEXT2(0, 'Write', writebuf, 10);
```

```
%Read From FPGA Board
```

```
[ret readbuf] = smMatVEXT2(0, 'Read', 10);
```

```
disp(readbuf);
```

```
%close FPGA Board
```

```
smMatVEXT2(0, 'Close');
```

Run and see the result as follows.

This circuit does invert, so the input data 1 to 10 will get the result 65534 to 65525.

```
>> TestVEXT2SDK
```

```
65534
```

```
65533
```

```
65532
```

```
65531
```

```
65530
```

```
65529
```

```
65528
```

```
65527
```

```
65526
```

65525

>>

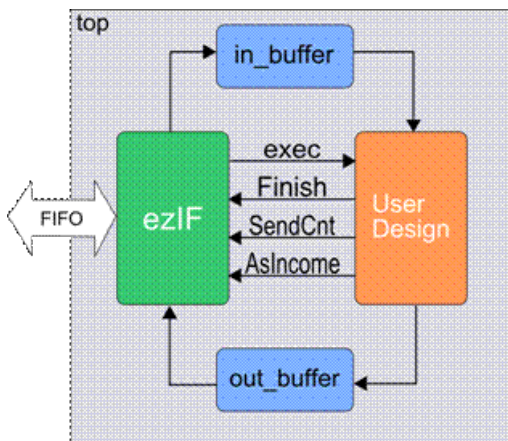
ezIF Interface

Introduce

Interface Description: (example\ Matlab\ezIF)

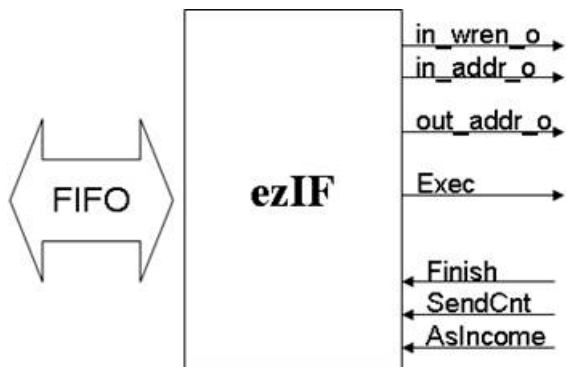
The ezIF is an interface to isolate users' design and FIFO interface. It can simplify user's design flow. The data received from software will be written in the memory by the ezIF. After ezIF writes all data in the memory, it will send a signal to info the User Design starts to work. After the User Design finished the process, it needs to send a signal to info the ezIF and a signal to tell whether to send data back to software side or not. The ezIF will automatically send user data back to the software side. The integration of ezIF and user design is shown as follow picture.

The ezIF is just an example to help user the combine the design with SMIMS development interface. If user needs more efficient or flexible operation mode, please design another mechanism to replace the ezIF.



Interface Operation

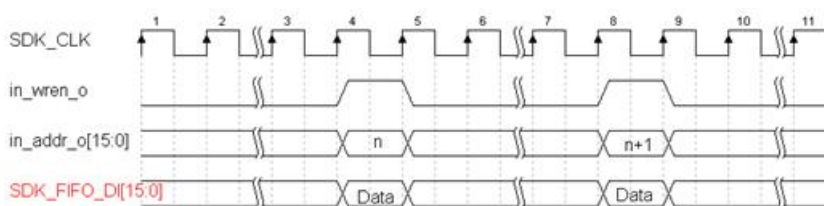
The interface operation of ezIF shows as follow diagram. The FIFO side is skipped. Here just introduces the right side signals.



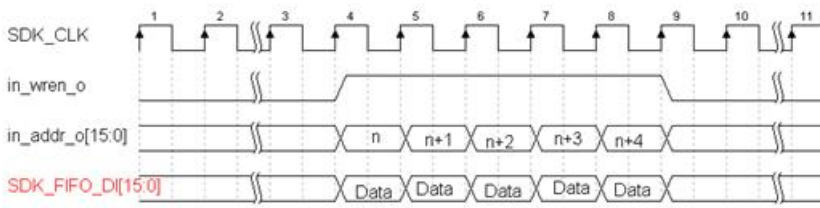
Signal Name	IO Type	Functional description
in_wren_o	output	Memory writing signal. 1 means to write data to the memory.
in_addr_o	output	Memory writing address
out_addr_o	output	Memory reading address for sending data to PC side.
Exec	output	Starts the user design
Finish	input	Info the user design finished the process.
SendCnt	input	The data count of sending data back to PC.
AslIncome	input	If the sending back data has the same counts as received data, then set this signal be 1.

Timing

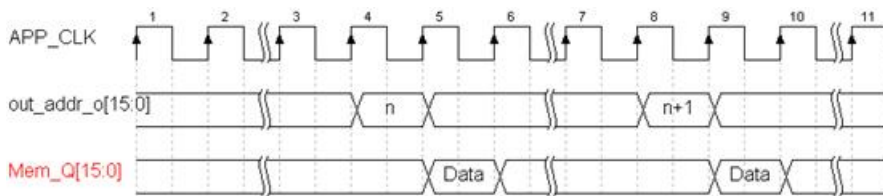
- ezIF writes single data to the memory (the data source is SDK_FIFO_DI of the FIFO).



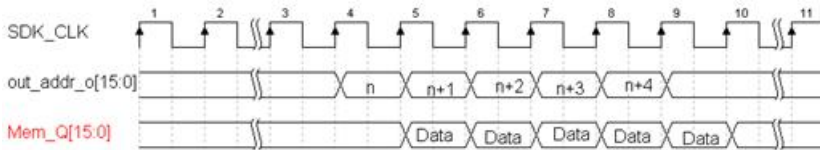
- ezIF writes burst data to the memory (the data source is SDK_FIFO_DI of the FIFO).



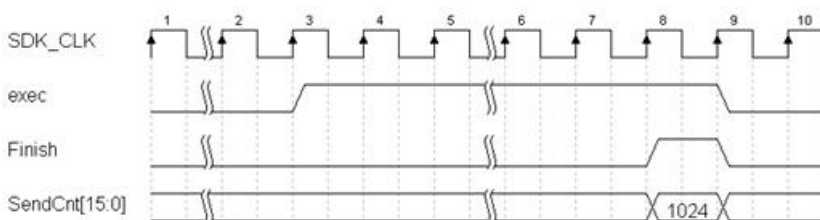
■ ezIF reads single data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_FIFO_DO).



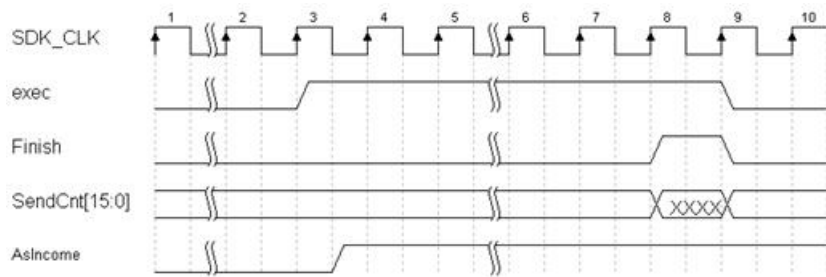
■ ezIF reads burst data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_FIFO_DO).



ezIF After the ezIF finished the receiving date, it will keep the Exec signal to be 1. The user design receives this signal will start to operate until the circuit finished the process, it will send out a Finish signal. Thus the Exec signal will be 0. When the user design sends out the Finish signal, the SendCnt should be set at the same time with how many data counts needs to send back to the software side. Therefore, the ezIF will follow the signals to send the data back.



If the sending back data has the same counts as received data, then set this signal be 1. In the mean time, the SendCnt will be ignored.



Software Control Interface

In the software side, user needs to set up the data counts before sending data. Before setting the data counts, the APP_CH of FIFO needs to be 1. After setting the data counts, set the APP_CH back to 0. The source code shown as follows. The important is that the settings needs a lot of time. Only if every time sends different data counts, otherwise please do not set it before every sending. The ideal way is setting once and then sends data continuously, until the data count is different then setup the new data count.

```
smMatVEXT2(0, 'ChSel', 1);
Cmd = data counts
smMatVEXT2(0, 'Write', Cmd,1);
smMatVEXT2(0, 'ChSel', 0);
```

Example Code

```
% Test run SMIMS VEXT2 SDK
close all;
clear all;
```

```
rand('seed',0);
```

```
% Open FPGA Board and set serial number.
```

```
smMatVEXT2(0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx', 0);
```

```
for i=1:4095
```

```
    smMatVEXT2(0, 'ChSel', 1);
```

```
    Cmd = i;
```

```
    smMatVEXT2(0, 'Write', Cmd,1);
```

```
    smMatVEXT2(0, 'ChSel', 0);
```

```
    fprintf('Send to HW Data Count : %d    ',i);
```

```
    flag = 1;
```

```
    // generate the random data
```

```
    WriteBuffer=rem(round(4096*rand(1,i)),4096);
```

```
    // send data to the circuit
```

```
    smMatVEXT2(0, 'Write', WriteBuffer,i);
```

```
    // receive data from the circuit
```

```
    [ret ReadBuffer] = smMatVEXT2(0, 'Read', i);
```

```
    // check the operation result
```

```
    for j=1:i
```

```
        if ReadBuffer(j) ~= (4096-WriteBuffer(j))
```

```
            flag = 0;
```

```
        end
```

```
    end
```

```
    if(flag == 1)
```

```
        fprintf('Check ok...\n');
```

```
    else
```

```
        fprintf('Error !\n');
```

```
        break;
```

```
    end
```

```
end
```

```
smMatVEXT2(0, 'Close');
```

Macube-VEXV7 Xilinx Virtex-7 USB

VEXV7 USB in Matlab command window

Support Models:
VEXV7-USB-STD
VEXV7-USB-EDU-STD
Target FPGA : Xilinx XC7V585T 、XC7VX690T

Requirement

- 4. SMIMS VEXV7 FPGA hardware device.
- 5. Matlab®/Simulink® of Mathwork®
- 6. Xilinx ISE design software.

VeriLink Function Introduction

This section introduces the way of the Matlab program to communicate with the user circuit in the FPGA by veriLink functions. After installed VeriLink in the PC, Matlab program is able to use VeriLink function.

















SMIMS Engine VeriLink function descriptions:
Since the VeriLink function supports up to 2 same type SMIMS FPGA boards in a single system, the 1st parameter of those functions indicates the selected board to operate. The board ID of the first plug-in board is 0.

Function Name	Function description
smMatVEXV7(iBoardID, 'FIFOOpen', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx')	Initial the SMIMS Engine for FIFO interface. Return true if success, false vice versa. The first input parameter is the board ID. The second parameter is the serial number.
smMatVEXV7(iBoardID, 'Close')	Close the SMIMS Engine, Return true if success, false vice versa.
[retvalue ReadBuf] = smMatVEXV7(iBoardID, 'FIFORead', ReadCount)	Read data from hardware. The return value ReadBuf is buffer to store data. The third parameter ReadCount is total data count to read. If the operation is successful, the return value retvalue will be true, false vice versa.
smMatVEXV7(iBoardID, 'FIFOWrite',	Send data to hardware. The third

<code>WriteBuf, WriteCount)</code>	parameter <code>WriteBuf</code> is data buffer to send and the data type is unit 16. The fourth parameter <code>WriteCount</code> is total data count to Send. If the operation is successful, this function will return true, false vice versa.
<code>smMatVEXV7(iBoardID, 'MEMOpen', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx')</code>	Initial the SMIMS Engine for memory interface. Return true if success, false vice versa. The first input parameter is the board ID. The second parameter is the serial number.
<code>[retvalue ReadBuf] = smMatVEXV7(iBoardID, 'MEMRead', Addr, ReadCount)</code>	Read data from hardware. The return value <code>ReadBuf</code> is buffer to store data. The third parameter <code>ReadCount</code> is total data count to read. If the operation is successful, the return value <code>retvalue</code> will be true, false vice versa. The 3 rd parameter <code>Addr</code> is the memory address.
<code>smMatVEXV7(iBoardID, 'MEMWrite', Addr, WriteBuf, WriteCount)</code>	Send data to hardware. The 3 rd parameter <code>Addr</code> is the memory address. The 4 th parameter <code>WriteBuf</code> is data buffer to send and the data type is unit 16. The 5 th parameter <code>WriteCount</code> is total data count to Send. If the operation is successful, this function will return true, false vice versa.
<code>smMatVEXV7(iBoardID, 'ChSel', Channel)</code>	Set 8 bits user-define data. Note: If the function is called without FIFO data empty, this function will be block and <code>SDK_CH[7:0]</code> will not changed immediately. <code>SDK_CH[7:0]</code> will be altered after all data in FIFO buffer is read out. The third parameter <code>Channel</code> is the setting value.
<code>smMatVEXV7(iBoardID, 'Program', 'BitFilePath')</code>	Program first FPGA. The third parameter <code>'BitFilePath'</code> is the FPGA programming binary file.
<code>smMatVEXV7(iBoardID, 'GetErrMsg')</code>	Retrieve SMIMS Engine Error Message, if an error occurs.
<code>smMatVEXV7(iBoardID, 'SetFreq', Freq)</code>	Set the FPGA clock frequency. The 3 rd parameter is the clock frequency in MHz. The range is 0.45MHz ~ 200MHz.
<code>result = smMatVEXV7(iBoardID, 'WaitReady', Mask, Timeout)</code>	Wait the FPGA interrupt signal, The function shall block until receiving the signal. The 3 rd parameter is mask. The 4 th parameter is timeout in millisecond. The return value <code>result</code> is the result.
<code>smMatVEXV7('H')</code>	Print the help message in the Matlab Command Window.

Hardware Interface

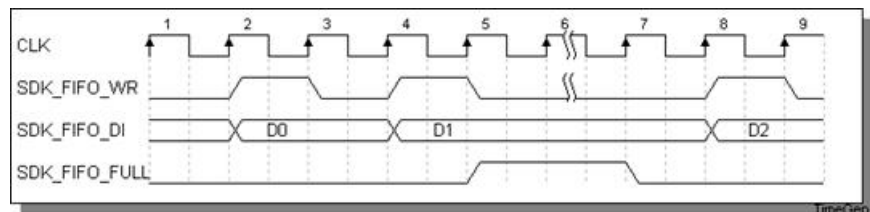
The hardware interface for user's design in FPGA is used the FIFO or memory interface. The signals and descriptions of hardware interface are listed as the following table.

Signal Name	Signal Type	Function Description
SDK_CLK	In → 	ClkMode =0, 48 MHz Clock Source for VEX platform.
SDK_RSTN	In → 	After programming a circuit into FPGA, the signal will be set to 0 . When 'Open' command is called, the signal will become 1
SDK_CH[7:0]	In → 	User can use this 8 bits channel for sending signal to the user Circuit.
SDK_FIFO_RD	In → 	When this signal is 1, SMIMS Engine start to read data.
SDK_FIFO_WR	In → 	When this signal is 1, SMIMS Engine start to write data.
SDK_FIFO_DI[15:0]	In → 	Data bus from SMIMS Engine.
SDK_FIFO_DO[15:0]	Out ← 	Data bus from user design to be written to SMIMS Engine buffer.
SDK_FIFO_Full	Out ← 	When this signal is 1, user design's FIFO id full.
SDK_FIFO_Empty	Out ← 	When this signal is 1, user design's FIFO id empty.
WRITE_READY	Out ← 	User design invokes this signal to unlock VEXV7_WaitReady.
READ_READY	Out ← 	User design invokes this signal to unlock VEXV7_WaitReady.
MEM_ADDR[23:0]	In → 	Memory address.
SDK_MEM_RD	In → 	When this signal is 1, SMIMS Engine start to read data.
SDK_MEM_DI[15:0]	In → 	Data bus from SMIMS Engine.
SDK_MEM_WR	In → 	When this signal is 1, SMIMS Engine start to write data.
SDK_MEM_DO[15:0]	Out ← 	Data bus from user design to be written to SMIMS Engine buffer.

Timing

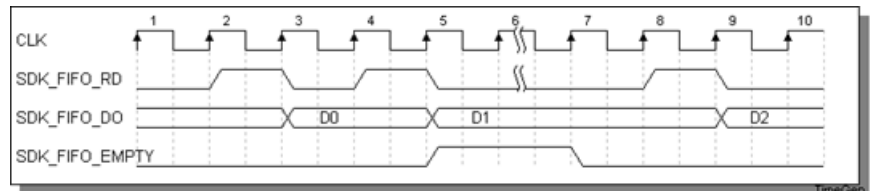
■ Write Multi Data through FIFO from PC to FPGA:

VeriLink Function : [smMatVEXV7 \(0, 'FIFOWrite', WriteBuf, 4\)](#)



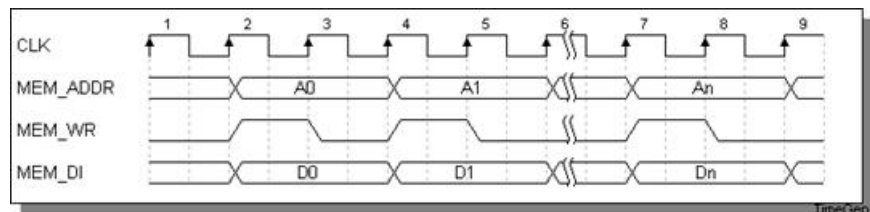
■ Read Multi Data through FIFO from FPGA to PC:

VeriLink Function : [smMatVEXV7 \(0, 'FIFORead', 5\)](#)



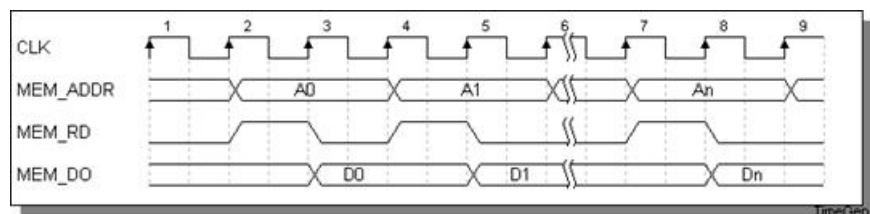
■ Write Multi Data through Memory from PC to FPGA:

VeriLink Function : [smMatVEXV7 \(0, 'MEMWrite', WriteBuf, 0, N\)](#)



■ Read Multi Data through Memory from FPGA to PC:

VeriLink Function : [smMatVEXV7 \(0, 'MEMRead', WriteBuf, 0, N\)](#)



PIN Map Information

FIFO Interface

(XC7V585T / XC7VC690T)

Signal Name	Pin Name		Signal Name	Pin Name	
SDK_CLK	W30	Input	SDK_CH[0]	AH32	Input
SDK_RSTN	Y28	Input	SDK_CH[1]	AH34	Input
SDK_WR	AM33	Input	SDK_CH[2]	AJ32	Input
SDK_RD	AP30	Input	SDK_CH[3]	AG33	Input
SDK_Full	W27	Input	SDK_CH[4]	AH33	Input
SDK_Empty	AB27	Input	SDK_CH[5]	AE32	Input
WRITE_READY	AF28	Output	SDK_CH[6]	AE33	Input
READ_READY	AE27	Output	SDK_CH[7]	AC30	Input
SDK_DI [0]	AE21	Input	SDK_DO [0]	U30	Output
SDK_DI [1]	AD19	Input	SDK_DO [1]	V30	Output
SDK_DI [2]	AE19	Input	SDK_DO [2]	AA30	Output
SDK_DI [3]	AF21	Input	SDK_DO [3]	AB30	Output
SDK_DI [4]	AG21	Input	SDK_DO [4]	W32	Output
SDK_DI [5]	AC18	Input	SDK_DO [5]	Y32	Output
SDK_DI [6]	AF25	Input	SDK_DO [6]	Y31	Output
SDK_DI [7]	AN33	Input	SDK_DO [7]	AA31	Output
SDK_DI [8]	AK34	Input	SDK_DO [8]	AJ34	Output
SDK_DI [9]	AL34	Input	SDK_DO [9]	AJ31	Output
SDK_DI [10]	AP33	Input	SDK_DO [10]	Y29	Output
SDK_DI [11]	AK32	Input	SDK_DO [11]	V27	Output
SDK_DI [12]	AK33	Input	SDK_DO [12]	V28	Output
SDK_DI [13]	AM32	Input	SDK_DO [13]	V29	Output
SDK_DI [14]	AN32	Input	SDK_DO [14]	W29	Output
SDK_DI [15]	AL33	Input	SDK_DO [15]	W26	Output

Note : Please change the setting of Pin I/O Std. to be LVCMOS18.

Memory Interface

(XC7V585T / XC7VC690T)

Signal Name	Pin Name		Signal Name	Pin Name	
SDK_CLK	W30	Input	WRITE_READY	AF28	Output
SDK_RSTN	Y28	Input	READ_READY	AE27	Output
MEM_WR	AM33	Input			
MEM_RD	AP30	Input			
MEM_DI [0]	AE21	Input	MEM_DO [0]	U30	Output
MEM_DI [1]	AD19	Input	MEM_DO [1]	V30	Output
MEM_DI [2]	AE19	Input	MEM_DO [2]	AA30	Output

MEM_DI [3]	AF21	Input	MEM_DO [3]	AB30	Output
MEM_DI [4]	AG21	Input	MEM_DO [4]	W32	Output
MEM_DI [5]	AC18	Input	MEM_DO [5]	Y32	Output
MEM_DI [6]	AF25	Input	MEM_DO [6]	Y31	Output
MEM_DI [7]	AN33	Input	MEM_DO [7]	AA31	Output
MEM_DI [8]	AK34	Input	MEM_DO [8]	AJ34	Output
MEM_DI [9]	AL34	Input	MEM_DO [9]	AJ31	Output
MEM_DI [10]	AP33	Input	MEM_DO [10]	Y29	Output
MEM_DI [11]	AK32	Input	MEM_DO [11]	V27	Output
MEM_DI [12]	AK33	Input	MEM_DO [12]	V28	Output
MEM_DI [13]	AM32	Input	MEM_DO [13]	V29	Output
MEM_DI [14]	AN32	Input	MEM_DO [14]	W29	Output
MEM_DI [15]	AL33	Input	MEM_DO [15]	W26	Output
MEM_ADDR[0]	AK27	Input	MEM_ADDR[12]	AP29	Input
MEM_ADDR[1]	AJ26	Input	MEM_ADDR[13]	AM25	Input
MEM_ADDR[2]	AJ27	Input	MEM_ADDR[14]	AN25	Input
MEM_ADDR[3]	AG25	Input	MEM_ADDR[15]	AM28	Input
MEM_ADDR[4]	AH25	Input	MEM_ADDR[16]	AN28	Input
MEM_ADDR[5]	AH24	Input	MEM_ADDR[17]	AP25	Input
MEM_ADDR[6]	AJ25	Input	MEM_ADDR[18]	AP26	Input
MEM_ADDR[7]	AL25	Input	MEM_ADDR[19]	AN27	Input
MEM_ADDR[8]	AL26	Input	MEM_ADDR[20]	AP27	Input
MEM_ADDR[9]	AM26	Input	MEM_ADDR[21]	AF24	Input
MEM_ADDR[10]	AM27	Input	MEM_ADDR[22]	AC27	Input
MEM_ADDR[11]	AN29	Input	MEM_ADDR[23]	AG32	Input

Note : Please change the setting of Pin I/O Std. to be LVCMOS18.

VeriEnterprise-VEXA7 Xilinx Artix-7 USB

VEXA7 USB in Matlab command window

Support Models:
VEX-A7-USB-STD
VEX-A7-USB-EDU-STD
Target FPGA : Xilinx ARTIX-7 XC7A200T

Requirement

- 1. SMIMS VEXA7 FPGA hardware device.
- 2. Matlab®/Simulink® of Mathwork®
- 3. Xilinx ISE design software.

VeriLink Function Introduction

This section introduces the way of the Matlab program to communicate with the user circuit in the FPGA by veriLink functions. After installed VeriLink in the PC, Matlab program is able to use VeriLink function.

















SMIMS Engine VeriLink function descriptions:
Since the VeriLink function supports up to 2 same type SMIMS FPGA boards in a single system, the 1st parameter of those functions indicates the selected board to operate. The board ID of the first plug-in board is 0.

Function Name	Function description
smMatVEXA200(iBoardID, 'FIFOOpen', 'xxxx-xxxx-xxxx-xxxx-xxxx')	Initial the SMIMS Engine for FIFO interface. Return true if success, false vice versa. The first input parameter is the board ID. The second parameter is the serial number.
smMatVEXA200 (iBoardID, 'Close')	Close the SMIMS Engine, Return true if success, false vice versa.
[retvalue ReadBuf] = smMatVEXA200 (iBoardID, 'FIFORead', ReadCount)	Read data from hardware. The return value ReadBuf is buffer to store data. The third parameter ReadCount is total data count to read. If the operation is successful, the return value retvalue will be true, false vice versa.

<code>smMatVEXA200 (iBoardID,'FIFOWrite', WriteBuf, WriteCount)</code>	Send data to hardware. The third parameter <code>WriteBuf</code> is data buffer to send and the data type is unit 16. The fourth parameter <code>WriteCount</code> is total data count to Send. If the operation is successful, this function will return true, false vice versa.
<code>smMatVEXA200 (iBoardID, 'MEMOpen','xxxx-xxxx-xxxx-xxxx-xxxx')</code>	Initial the SMIMS Engine for memory interface. Return true if success, false vice versa. The first input parameter is the board ID. The second parameter is the serial number.
<code>[retval ReadBuf] = smMatVEXA200 (iBoardID,'MEMRead', Addr, ReadCount)</code>	Read data from hardware. The return value <code>ReadBuf</code> is buffer to store data. The third parameter <code>ReadCount</code> is total data count to read. If the operation is successful, the return value <code>retval</code> will be true, false vice versa. The 3 rd parameter <code>Addr</code> is the memory address.
<code>smMatVEXA200 (iBoardID,'MEMWrite', Addr, WriteBuf, WriteCount)</code>	Send data to hardware. The 3 rd parameter <code>Addr</code> is the memory address. The 4 th parameter <code>WriteBuf</code> is data buffer to send and the data type is unit 16. The 5 th parameter <code>WriteCount</code> is total data count to Send. If the operation is successful, this function will return true, false vice versa.
<code>smMatVEXA200 (iBoardID,'ChSel', Channel)</code>	Set 8 bits user-define data. Note: If the function is called without FIFO data empty, this function will be block and <code>SDK_CH[7:0]</code> will not changed immediately. <code>SDK_CH[7:0]</code> will be altered after all data in FIFO buffer is read out. The third parameter <code>Channel</code> is the setting value.
<code>smMatVEXA200(iBoardID,'Program', 'BitFilePath')</code>	Program first FPGA. The third parameter <code>'BitFilePath'</code> is the FPGA programming binary file.
<code>smMatVEXA200(iBoardID,'GetErrMsg')</code>	Retrieve SMIMS Engine Error Message, if an error occurs.
<code>smMatVEXA200(iBoardID,'SetFreq', Freq)</code>	Set the FPGA clock frequency. The 3 rd parameter is the clock frequency in MHz. The range is 0.45MHz ~ 200MHz.
<code>result = smMatVEXA200(iBoardID,'WaitReady', Mask, Timeout)</code>	Wait the FPGA interrupt signal, The function shall block until receiving the signal. The 3 rd parameter is mask. The 4 th parameter is timeout in millisecond. The return value <code>result</code> is the result.
<code>smMatVEXA200('H')</code>	Print the help message in the Matlab Command Window.

Hardware Interface

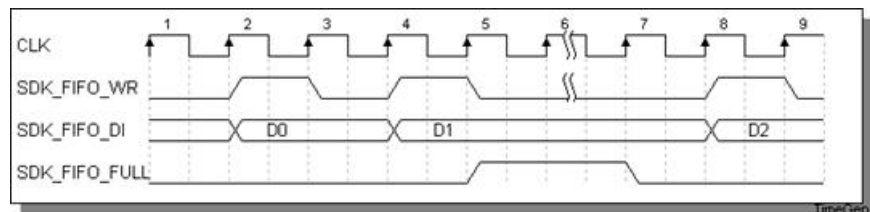
The hardware interface for user's design in FPGA is used the FIFO or memory interface. The signals and descriptions of hardware interface are listed as the following table.

Signal Name	Signal Type	Function Description
SDK__CLK	In → 	ClkMode =0, 48 MHz Clock Source for VEX platform.
SDK_RSTN	In → 	After programming a circuit into FPGA, the signal will be set to 0 . When 'Open' command is called, the signal will become 1
SDK_CH[7:0]	In → 	User can use this 8 bits channel for sending signal to the user Circuit.
SDK_FIFO_RD	In → 	When this signal is 1, SMIMS Engine start to read data.
SDK_FIFO_WR	In → 	When this signal is 1, SMIMS Engine start to write data.
SDK_FIFO_DI[15:0]	In → 	Data bus from SMIMS Engine.
SDK_FIFO_DO[15:0]	Out ← 	Data bus from user design to be written to SMIMS Engine buffer.
SDK_FIFO_Full	Out ← 	When this signal is 1, user design's FIFO id full.
SDK_FIFO_Empty	Out ← 	When this signal is 1, user design's FIFO id empty.
WRITE_READY	Out ← 	User design invokes this signal to unlock VEXV7_WaitReady.
READ_READY	Out ← 	User design invokes this signal to unlock VEXV7_WaitReady.
MEM_ADDR[23:0]	In → 	Memory address.
SDK_MEM_RD	In → 	When this signal is 1, SMIMS Engine start to read data.
SDK_MEM_DI[15:0]	In → 	Data bus from SMIMS Engine.
SDK_MEM_WR	In → 	When this signal is 1, SMIMS Engine start to write data.
SDK_MEM_DO[15:0]	Out ← 	Data bus from user design to be written to SMIMS Engine buffer.

Timing

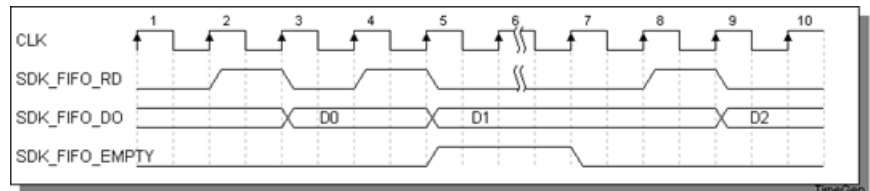
■ Write Multi Data through FIFO from PC to FPGA:

VeriLink Function : [smMatVEXA200 \(0, 'FIFOWrite', WriteBuf, 4\)](#)



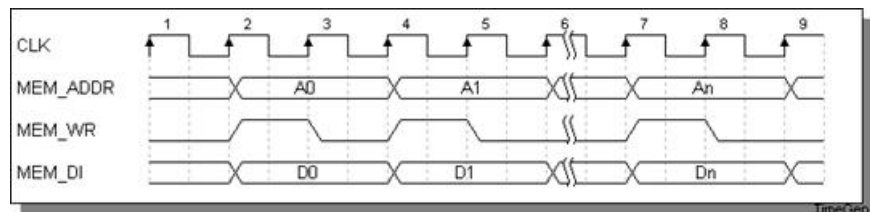
■ Read Multi Data through FIFO from FPGA to PC:

VeriLink Function : [smMatVEXA200 \(0, 'FIFORead', 5\)](#)



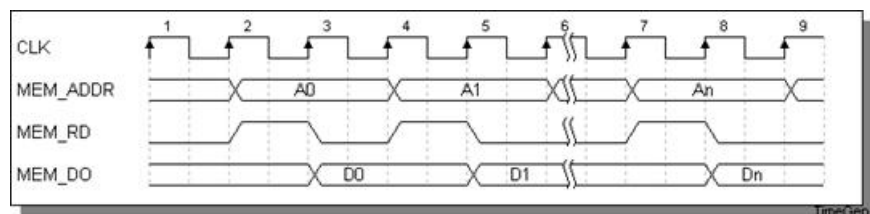
■ Write Multi Data through Memory from PC to FPGA:

VeriLink Function : [smMatVEXA200 \(0, 'MEMWrite', WriteBuf, 0, N\)](#)



■ Read Multi Data through Memory from FPGA to PC:

VeriLink Function : [smMatVEXA200 \(0, 'MEMRead', WriteBuf, 0, N\)](#)



PIN Map Information

FIFO Interface

(XC7A200T)

Signal Name	Pin Name		Signal Name	Pin Name	
SDK_CLK	K21	Input	SDK_CH[0]	N24	Input
SDK_RSTN	T17	Input	SDK_CH[1]	P23	Input
SDK_WR	L25	Input	SDK_CH[2]	R22	Input
SDK_RD	M24	Input	SDK_CH[3]	R23	Input
SDK_Full	W19	Output	SDK_CH[4]	T22	Input
SDK_Empty	V17	Output	SDK_CH[5]	T23	Input
			SDK_CH[6]	U22	Input
			SDK_CH[7]	V23	Input
SDK_DI [0]	D25	Input	SDK_DO [0]	J19	Output
SDK_DI [1]	D26	Input	SDK_DO [1]	K18	Output
SDK_DI [2]	E25	Input	SDK_DO [2]	L18	Output
SDK_DI [3]	E26	Input	SDK_DO [3]	L19	Output
SDK_DI [4]	F24	Input	SDK_DO [4]	M19	Output
SDK_DI [5]	F25	Input	SDK_DO [5]	N18	Output
SDK_DI [6]	G24	Input	SDK_DO [6]	N19	Output
SDK_DI [7]	G25	Input	SDK_DO [7]	P18	Output
SDK_DI [8]	G26	Input	SDK_DO [8]	P19	Output
SDK_DI [9]	H26	Input	SDK_DO [9]	R18	Output
SDK_DI [10]	J24	Input	SDK_DO [10]	T18	Output
SDK_DI [11]	J25	Input	SDK_DO [11]	T19	Output
SDK_DI [12]	J26	Input	SDK_DO [12]	U19	Output
SDK_DI [13]	K25	Input	SDK_DO [13]	V18	Output
SDK_DI [14]	K26	Input	SDK_DO [14]	V19	Output
SDK_DI [15]	L24	Input	SDK_DO [15]	W18	Output

Note : Please change the setting of Pin I/O Std. to be LVCMOS33.

Memory Interface

(XC7A200T)

Signal Name	Pin Name		Signal Name	Pin Name	
SDK_CLK	K21	Input	SDK_RSTN	T17	Input
MEM_WR	L25	Input	WRITE_READY	K17	Output
MEM_RD	M24	Input	READ_READY	K26	Output
MEM_DI[0]	D25	Input	MEM_DO[0]	J19	Output
MEM_DI[1]	D26	Input	MEM_DO [1]	K18	Output
MEM_DI[2]	E25	Input	MEM_DO [2]	L18	Output
MEM_DI[3]	E26	Input	MEM_DO [3]	L19	Output
MEM_DI[4]	F24	Input	MEM_DO [4]	M19	Output

MEM_DI[5]	F25	Input	MEM_DO [5]	N18	Output
MEM_DI[6]	G24	Input	MEM_DO [6]	N19	Output
MEM_DI[7]	G25	Input	MEM_DO [7]	P18	Output
MEM_DI[8]	G26	Input	MEM_DO [8]	P19	Output
MEM_DI[9]	H26	Input	MEM_DO [9]	R18	Output
MEM_DI[10]	J24	Input	MEM_DO [10]	T18	Output
MEM_DI[11]	J25	Input	MEM_DO [11]	T19	Output
MEM_DI[12]	J26	Input	MEM_DO [12]	U19	Output
MEM_DI[13]	K25	Input	MEM_DO [13]	V18	Output
MEM_DI[14]	K26	Input	MEM_DO [14]	V19	Output
MEM_DI[15]	L24	Input	MEM_DO [15]	W18	Output
MEM_ADDR[0]	W24	Input	MEM_ADDR[12]	F22	Input
MEM_ADDR[1]	W25	Input	MEM_ADDR[13]	F23	Input
MEM_ADDR[2]	W26	Input	MEM_ADDR[14]	G22	Input
MEM_ADDR[3]	Y25	Input	MEM_ADDR[15]	H23	Input
MEM_ADDR[4]	Y26	Input	MEM_ADDR[16]	H24	Input
MEM_ADDR[5]	AA24	Input	MEM_ADDR[17]	J23	Input
MEM_ADDR[6]	AA25	Input	MEM_ADDR[18]	K22	Input
MEM_ADDR[7]	AB24	Input	MEM_ADDR[19]	K23	Input
MEM_ADDR[8]	AB25	Input	MEM_ADDR[20]	L22	Input
MEM_ADDR[9]	AB26	Input	MEM_ADDR[21]	L23	Input
MEM_ADDR[10]	AC26	Input	MEM_ADDR[22]	M22	Input
MEM_ADDR[11]	E23	Input	MEM_ADDR[23]	N23	Input

Note : Please change the setting of Pin I/O Std. to be LVCMOS33.

Pin Table of Peripheral Device

[Refer to the VEX-A200_manual_xxxxQx.pdf](#)

VeriLite Altera Cyclone IV USB

VeriLite Altera Cyclone IV USB in Matlab command window

Support Models:

VLAC4-USB-STD

VL AC4-USB-EDU-STD

Target FPGA : Altera Cyclone IV EP4CE30F23C8

Requirement

1. SMIMS VeriLite Altera Cyclone IV USB FPGA hardware device.
2. Matlab®/Simulink® of Mathwork®
3. Altera Quartus II 10.0 and above

VeriLink Function

This section introduces the way of the Matlab program to communicate with the user circuit in the FPGA by VeriLink functions. After installed VeriLink in the PC, Matlab program is able to use VeriLink function.

SMIMS Engine VeriLink function descriptions:






Since the VeriLink function supports up to 2 same type SMIMS FPGA boards in a single system, the 1st parameter of those functions indicates the selected board to operate. The board ID of the first plug-in board is 0.



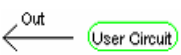
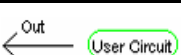


Function Name	Function description
<code>smMatVLAC4(iBoardID, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx')</code>	Initial the SMIMS Engine. Return true if success, false vice versa. The first input parameter is the board ID. The other is the serial number.
<code>smMatVLAC4 (iBoardID, 'Close')</code>	Close the SMIMS Engine, Return true if success, false vice versa.
<code>[retvalue ReadBuf] = smMatVLAC4 (iBoardID, 'Read', ReadCount)</code>	Read Data From Hardware. The return value <code>ReadBuf</code> is buffer to store data. The third parameter <code>ReadCount</code> is total data count to read. If the operation is successful, the return value <code>retvalue</code> will be true, false vice versa.

smMatVLAC4 (iBoardID,'Write', WriteBuf, WriteCount)	Send Data to Hardware. The third parameter WriteBuf is data buffer to send and the data type is unit 16. The fourth parameter WriteCount is total data count to Send. If the operation is successful, this function will return true, false vice versa.
smMatVLAC4 (iBoardID,'ChSel', Channel)	Set 8 bits user-define data. . Note: If the function is called without FIFO data empty, this function will be block and SDK_CH[7:0] will not changed immediately. SDK_CH[7:0] will be altered after all data in FIFO buffer is read out. The third parameter Channel is the setting value.
smMatVLAC4 (iBoardID,'Program', 'BitFilePath')	Program FPGA. The third parameter BitFilePath is the FPGA programming binary file.
smMatVLAC4 (iBoardID,'GetErrMsg')	Retrieve SMIMS Engine Error Message, if an error occurs.
smMatVLAC4 ('H')	Print the help message in the Matlab Command Window.

Hardware Interface

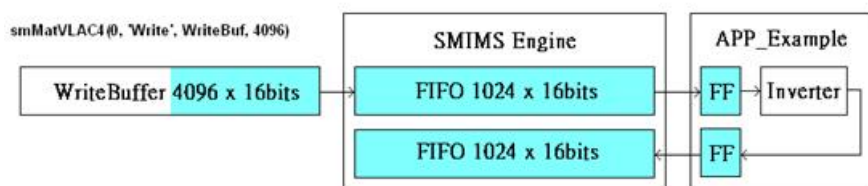
The hardware interface for user's design in FPGA is used the FIFO interface. The signals and descriptions of hardware interface are listed as the following table.

Signal Name	Signal Type	Signal Description
SDK_CLK	In → 	48 MHz Clock Source from SMIMS engine to VLAC4 platform
SDK_RSTN	In → 	After programming a circuit into FPGA, the signal will be set to 0. When SMIMS_VLAC4_AppOpen is called, the signal will be set to 1. When SMIMS_VLAC4_AppReset is called, a 1-to-0-to-1 pulse will be invoked to reset user's circuit.
SDK_CH[7:0]	In → 	User defined 8-bits bus.
SDK_RD	← Out 	User design invokes this signal to read FIFO data from SMIMS engine.
SDK_DI[15:0]	In → 	Data bus from SMIMS Engine FIFO.

SDK_Empty		When this signal is 1, SMIMS Engine FIFO buffer for user design is empty. There is no data to be read.
SDK_AlmostEmpty		When this signal is 1, SMIMS Engine FIFO buffer for user design is only one data available.
SDK_WR		User design invokes this signal to write data to SMIMS engine FIFO buffer.
SDK_DO[15:0]		Data bus from user design to be written to SMIMS Engine FIFO buffer.
SDK_Full		When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is full and it can not be written any data more.
SDK_AlmostFull		When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is only one space left.

VeriLite Xilinx Spartan-6 USB in VeriLink function mode provides the 16bits data bus. Because the data transmit as FIFO, if users want to use a SRAM, they should generate their own address for the SRAM.

There are two 1024 x 16 bits FIFO inside the SMIMS Engine. In the example “Inverter”, if write a data size bigger than the FIFO buffer size of the SMIMS Engine, the program will be blocked in the API `smMatVLAC4(0, 'Write', WriteBuf, WriteCount)`. The picture shows this situation below.

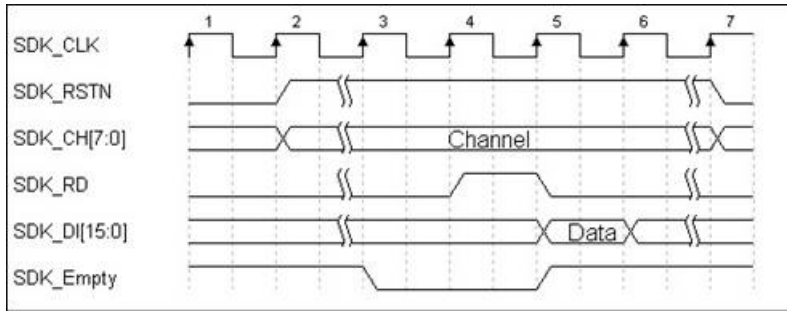


Timing

■ Write Single Data from PC to FPGA:

VeriLink Function : `smMatVLAC4(0, 'Write', WriteBuf, 1)`

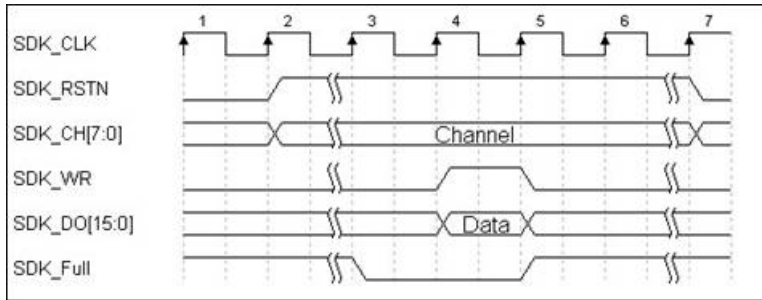
The timing diagram is shown as follows. When SDK_Empty signal is low and SDK_RD activates for one clock period, the data will be read in the next clock period.



■ Read Single Data from FPGA to PC:

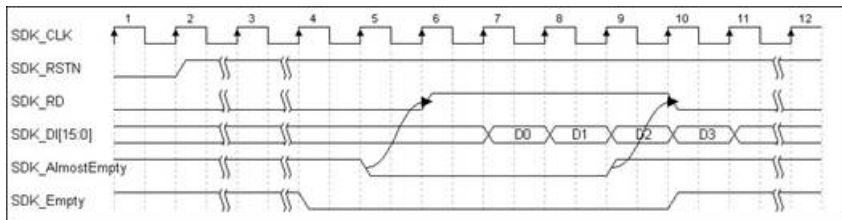
VeriLink Function : [smMatVLAC4 \(0, 'Read', 1\)](#)

The timing diagram is shown as follows. When SDK_Full signal is low, sends out the SDK_WR and SDK_DO data in the same time.



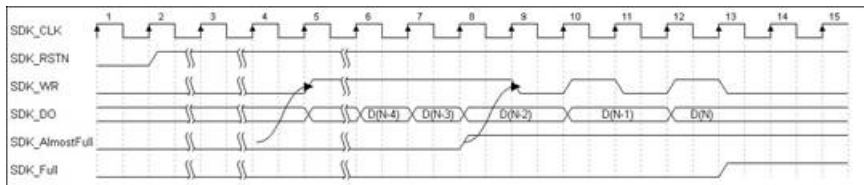
■ Write Multi Data from PC to FPGA:

VeriLink Function : [smMatVLAC4\(0, 'Write', WriteBuf, 4\)](#)



■ Read Multi Data from FPGA to PC:

VeriLink Function : [smMatVLAC4 \(0, 'Read', 5\)](#)



PIN Map Information

Note : Please set those pins IO standards to be **LVCMOS33** for better performance. You may refer to ucf file in the SDK/example directory

(EP4CE30)

Signal Name	Pin Name		Signal Name	Pin Name	
SDK_CLK	PIN_T2	Input	SDK_CH[0]	PIN_AB5	Input
SDK_RSTN	PIN_T5	Input	SDK_CH[1]	PIN_V5	Input
SDK_WR	PIN_AA14	Output	SDK_CH[2]	PIN_V6	Input
SDK_RD	PIN_AB14	Output	SDK_CH[3]	PIN_V7	Input
SDK_Full	PIN_V2	Input	SDK_CH[4]	PIN_Y6	Input
SDK_Empty	PIN_W2	Input	SDK_CH[5]	PIN_W7	Input
SDK_AlmostFull	PIN_V1	Input	SDK_CH[6]	PIN_AB7	Input
SDK_AlmostEmpty	PIN_Y2	Input	SDK_CH[7]	PIN_W8	Input
SDK_Interrupt	PIN_M19	Output			
SDK_DI [0]	PIN_M3	Input	SDK_DO [0]	PIN_W1	Output
SDK_DI [1]	PIN_M2	Input	SDK_DO [1]	PIN_AA4	Output
SDK_DI [2]	PIN_M1	Input	SDK_DO [2]	PIN_W6	Output
SDK_DI [3]	PIN_N1	Input	SDK_DO [3]	PIN_Y7	Output
SDK_DI [4]	PIN_P4	Input	SDK_DO [4]	PIN_AA7	Output
SDK_DI [5]	PIN_P3	Input	SDK_DO [5]	PIN_AB8	Output
SDK_DI [6]	PIN_P1	Input	SDK_DO [6]	PIN_U7	Output
SDK_DI [7]	PIN_R2	Input	SDK_DO [7]	PIN_AB9	Output
SDK_DI [8]	PIN_R1	Input	SDK_DO [8]	PIN_U13	Output
SDK_DI [9]	PIN_P6	Input	SDK_DO [9]	PIN_V13	Output
SDK_DI [10]	PIN_R6	Input	SDK_DO [10]	PIN_W13	Output
SDK_DI [11]	PIN_T4	Input	SDK_DO [11]	PIN_Y13	Output
SDK_DI [12]	PIN_M4	Input	SDK_DO [12]	PIN_AA13	Output
SDK_DI [13]	PIN_N2	Input	SDK_DO [13]	PIN_AB13	Output
SDK_DI [14]	PIN_U2	Input	SDK_DO [14]	PIN_U14	Output
SDK_DI [15]	PIN_U1	Input	SDK_DO [15]	PIN_V14	Output

Example

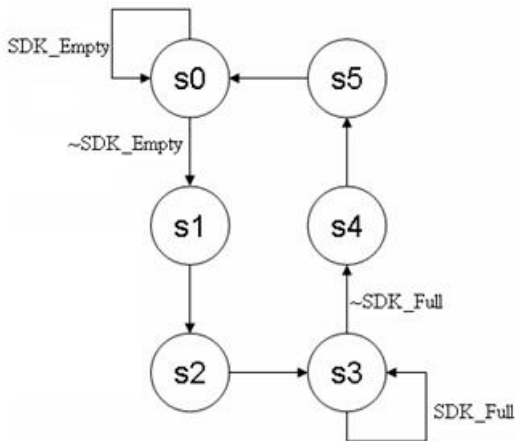
Example Inverter

Example Description: (example\Inverter)

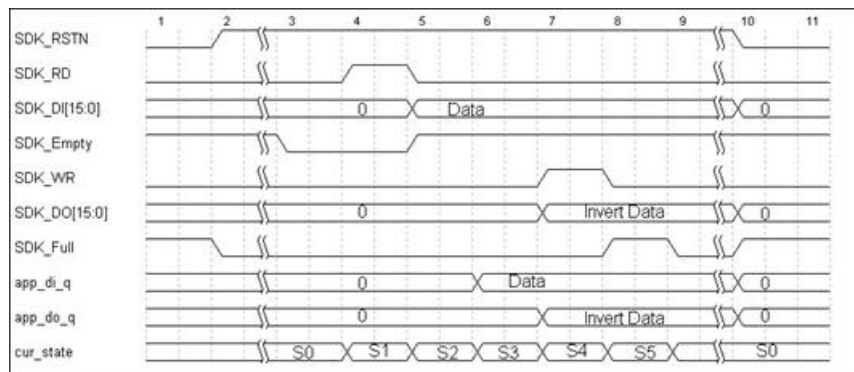
This example, Inverter, show how to use the VeriLink Function to design the Matlab program working with user's circuit. There are hardware and software two parts.

Hardware

The hardware design (single read/write) is to read data from Input FIFO, pass through an inverter, and then output the data to the Output FIFO. Here use a FSM (Finite state machine) to implement it. The state diagram is shown as follows.



The detailed timing diagram is shown in the following figure.



Software

The software part runs the script file in Matlab. This example uses the least SMIMS function to explain how to use the function. It is a simple and quick introduction. The software source code Matlab script file is TestVLAC4SDK.m.

Software program explanation (TestVLAC4SDK.m)

[% Test run SMIMS VLAC4 SDK](#)

Clear

```
%Program .bit file tp FPGA
```

```
smMatVLAC4 (0, 'Program', 'c:\app_example.rbf');
```

```
% Open FPGA Board and set serial number.
```

```
smMatVLAC4 (0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx');
```

```
for i=1:10
```

```
    writebuf(i)=uint16(i);
```

```
end
```

```
%Write to FPGA Board
```

```
smMatVLAC4 (0, 'Write', writebuf, 10);
```

```
%Read From FPGA Board
```

```
[ret readbuf] = smMatVLAC4 (0, 'Read', 10);
```

```
disp(readbuf);
```

```
%close FPGA Board
```

```
smMatVLAC4(0, 'Close');
```

Run and see the result as follows.

This circuit does invert, so the input data 1 to 10 will get the result 65534 to 65525.

```
>> TestVLAC4SDK
```

```
65534
```

```
65533
```

```
65532
```

```
65531
```

```
65530
```

```
65529
```

```
65528
```

```
65527
```

```
65526
```

```
65525
```

```
>>
```

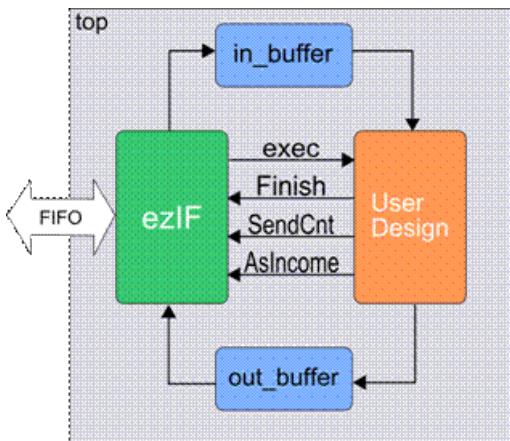
ezIF Interface

Introduce

Interface Description: (example\ Matlab\ezIF)

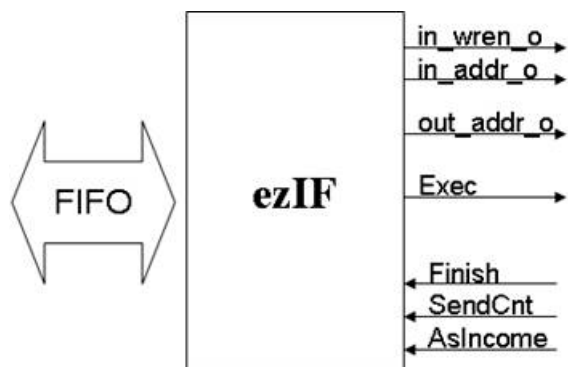
The ezIF is an interface to isolate users' design and FIFO interface. It can simplify user's design flow. The data received from software will be written in the memory by the ezIF. After ezIF writes all data in the memory, it will send a signal to info the User Design starts to work. After the User Design finished the process, it needs to send a signal to info the ezIF and a signal to tell whether to send data back to software side or not. The ezIF will automatically send user data back to the software side. The integration of ezIF and user design is shown as follow picture.

The ezIF is just an example to help user the combine the design with SMIMS development interface. If user needs more efficient or flexible operation mode, please design another mechanism to replace the ezIF.



Interface Operation

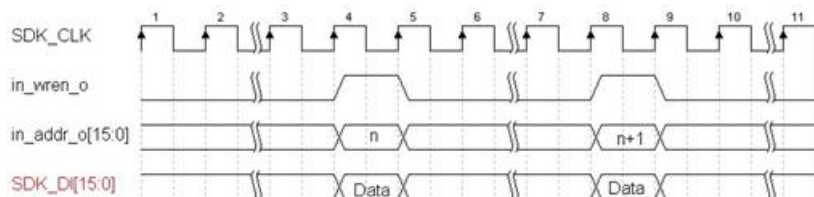
The interface operation of ezIF shows as follow diagram. The FIFO side is skipped. Here just introduces the right side signals.



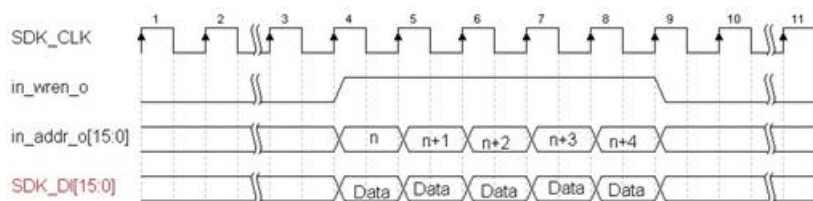
Signal Name	IO Type	Functional description
in_wren_o	output	Memory writing signal. 1 means to write data to the memory.
in_addr_o	output	Memory writing address
out_addr_o	output	Memory reading address for sending data to PC side.
Exec	output	Starts the user design
Finish	input	Info the user design finished the process.
SendCnt	input	The data count of sending data back to PC.
AsIncome	input	If the sending back data has the same counts as received data, then set this signal be 1.

Timing

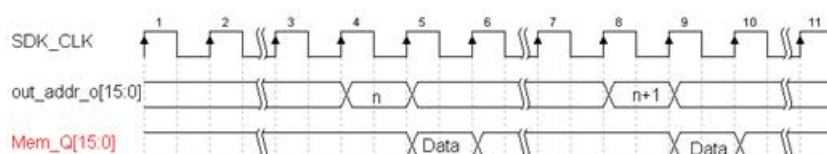
- ezIF writes single data to the memory (the data source is SDK_DI of the FIFO).



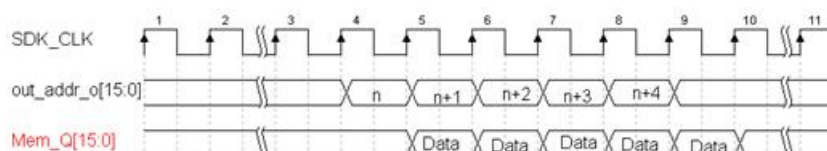
- ezIF writes burst data to the memory (the data source is SDK_DI of the FIFO).



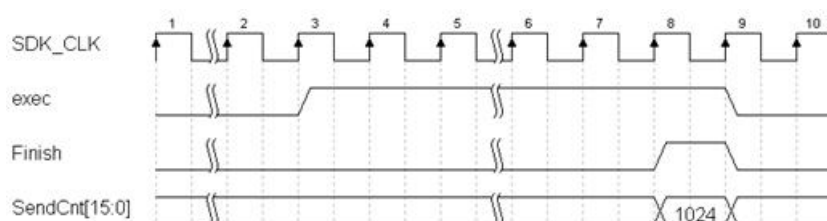
- ezIF reads single data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_DO).



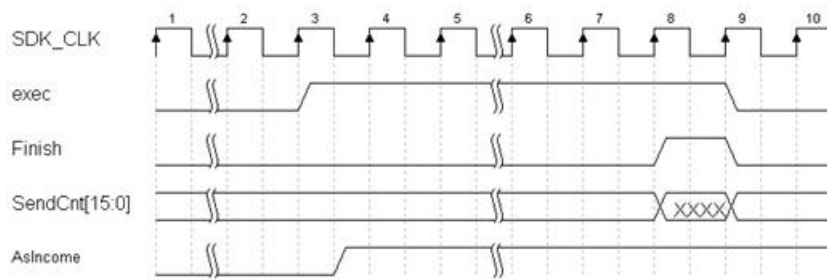
- ezIF reads burst data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_DO).



After the ezIF finished the receiving date, it will keep the Exec signal to be 1. The user design receives this signal will start to operate until the circuit finished the process, it will send out a Finish signal. Thus the Exec signal will be 0. When the user design sends out the Finish signal, the SendCnt should be set at the same time with how many data counts needs to send back to the software side. Therefore, the ezIF will follow the signals to send the data back.



If the sending back data has the same counts as received data, then set this signal be 1. In the mean time, the SendCnt will be ignored.



Software Control Interface

In the software side, user needs to set up the data counts before sending data. Before setting the data counts, the SDK_CH of FIFO needs to be 1. After setting the data counts, set the SDK_CH back to 0. The source code shown as follows. The important is that the settings needs a lot of time. Only if every time sends different data counts, otherwise please do not set it before every sending. The ideal way is setting once and then sends data continuously, until the data count is different then setup the new data count.

```
smMatVLAC4(0, 'ChSel', 1);
Cmd = data counts
smMatVLAC4(0, 'Write', Cmd, 1);
smMatVLAC4(0, 'ChSel', 0);
```

Example Code

```
% Test run SMIMS VLAC4 SDK
close all;
clear all;
```

```
rand('seed',0);
```

```
% Open FPGA Board and set serial number.
```

```
smMatVLAC4(0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx');
```

```
for i=1:4095
```

```
    smMatVLAC4(0, 'ChSel', 1);
```

```
    Cmd = i;
```

```
    smMatVLAC4(0, 'Write', Cmd,1);
```

```
    smMatVLAC4(0, 'ChSel', 0);
```

```
    fprintf('Send to HW Data Count : %d    ',i);
```

```
    flag = 1;
```

```
    // generate the random data
```

```
    WriteBuffer=rem(round(4096*rand(1,i)),4096);
```

```
    // send data to the circuit
```

```
    smMatVLAC4(0, 'Write', WriteBuffer,i);
```

```
    // receive data from the circuit
```

```
    [ret ReadBuffer] = smMatVLAC4(0, 'Read', i);
```

```
    // check the operation result
```

```
    for j=1:i
```

```
        if ReadBuffer(j) ~= (4096-WriteBuffer(j))
```

```
            flag = 0;
```

```
        end
```

```
    end
```

```
    if(flag == 1)
```

```
        fprintf('Check ok...\n');
```

```
    else
```

```
        fprintf('Error !\n');
```

```
        break;
```

```
    end
```

```
end
```

```
smMatVLAC4(0, 'Close');
```

VeriLite Xilinx Spartan-6 USB

VeriLite Xilinx Spartan-6 USB in Matlab command window

Support Models:

VLXS6-USB-STD

VLXS6-USB-EDU-STD

Target FPGA : Xilinx Spartan-6 XC6SLX16 、XC6SLX25

Requirement

1. SMIMS VeriLite Xilinx Spartan-6 USB FPGA hardware device.
2. Matlab®/Simulink® of Mathwork®
3. Xilinx ISE design software 11.2 and above

VeriLink Function

This section introduces the way of the Matlab program to communicate with the user circuit in the FPGA by VeriLink functions. After installed VeriLink in the PC, Matlab program is able to use VeriLink function.

SMIMS Engine VeriLink function descriptions:







Since the VeriLink function supports up to 2 same type SMIMS FPGA boards in a single system, the 1st parameter of those functions indicates the selected board to operate. The board ID of the first plug-in board is 0.






Function Name	Function description
<code>smMatVLXSP6(iBoardID, 'Open', 'xxx-xxx-xxx-xxx-xxx-xxx')</code>	Initial the SMIMS Engine. Return true if success, false vice versa. The first input parameter is the board ID. The other is the serial number.
<code>smMatVLXSP6 (iBoardID,'Close')</code>	Close the SMIMS Engine, Return true if success, false vice versa.
<code>[retvalue ReadBuf] = smMatVLXSP6 (iBoardID,'Read', ReadCount)</code>	Read Data From Hardware. The return value <code>ReadBuf</code> is buffer to store data. The third parameter <code>ReadCount</code> is total data count to read. If the operation is successful, the return value <code>retvalue</code> will be true, false vice versa.
<code>smMatVLXSP6 (iBoardID,'Write',</code>	Send Data to Hardware. The third parameter

WriteBuf , WriteCount)	WriteBuf is data buffer to send and the data type is unit 16. The fourth parameter WriteCount is total data count to Send. If the operation is successful, this function will return true, false vice versa.
smMatVLXSP6 (iBoardID , 'ChSel' , Channel)	Set 8 bits user-define data. . Note: If the function is called without FIFO data empty, this function will be block and SDK_CH[7:0] will not changed immediately. SDK_CH[7:0] will be altered after all data in FIFO buffer is read out. The third parameter Channel is the setting value.
smMatVLXSP6 (iBoardID , 'Program' , 'BitFilePath')	Program FPGA. The third parameter BitFilePath is the FPGA programming binary file.
smMatVLXSP6 (iBoardID , 'GetErrMsg')	Retrieve SMIMS Engine Error Message, if an error occurs.
smMatVLXSP6 ('H')	Print the help message in the Matlab Command Window.

Hardware Interface

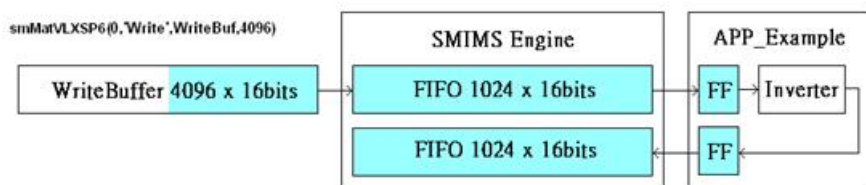
The hardware interface for user's design in FPGA is used the FIFO interface. The signals and descriptions of hardware interface are listed as the following table.

Signal Name	Signal Type	Signal Description
SDK_CLK	In → 	48 MHz Clock Source from SMIMS engine to VLXSP6 platform
SDK_RSTN	In → 	After programming a circuit into FPGA, the signal will be set to 0. When SMIMS_VLXSP6_AppOpen is called, the signal will be set to 1. When SMIMS_VLXSP6_AppReset is called, a 1-to-0-to-1 pulse will be invoked to reset user's circuit.
SDK_CH[7:0]	In → 	User defined 8-bits bus.
SDK_RD	← Out 	User design invokes this signal to read FIFO data from SMIMS engine.
SDK_DI[15:0]	In → 	Data bus from SMIMS Engine FIFO.
SDK_Empty	In → 	When this signal is 1, SMIMS Engine FIFO buffer for user design

		is empty. There is no data to be read.
SDK_AlmostEmpty	In → 	When this signal is 1, SMIMS Engine FIFO buffer for user design is only one data available.
SDK_WR	← Out 	User design invokes this signal to write data to SMIMS engine FIFO buffer.
SDK_DO[15:0]	← Out 	Data bus from user design to be written to SMIMS Engine FIFO buffer.
SDK_Full	In → 	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is full and it can not be written any data more.
SDK_AlmostFull	In → 	When this signal is 1, SMIMS Engine FIFO buffer for storing user design's output data is only one space left.

VeriLite Xilinx Spartan-6 USB in VeriLink function mode provides the 16bits data bus. Because the data transmit as FIFO, if users want to use a SRAM, they should generate their own address for the SRAM.

There are two 1024 x 16 bits FIFO inside the SMIMS Engine. In the example “Inverter”, if write a data size bigger than the FIFO buffer size of the SMIMS Engine, the program will be blocked in the API `smMatVLXSP6(0, 'Write', WriteBuf, WriteCount)`. The picture shows this situation below.

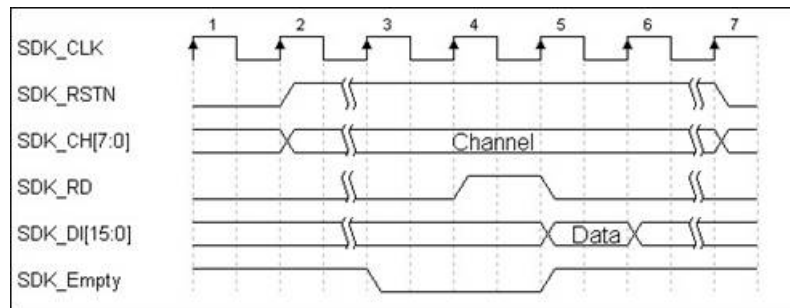


Timing

■ Write Single Data from PC to FPGA:

VeriLink Function : [smMatVLXSP6\(0, 'Write', WriteBuf, 1\)](#)

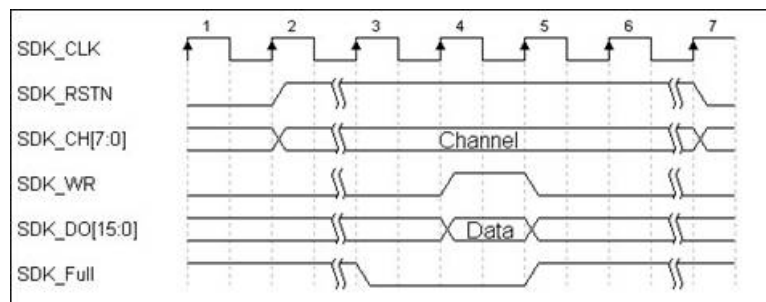
The timing diagram is shown as follows. When SDK_Empty signal is low and SDK_RD activates for one clock period, the data will be read in the next clock period.



■ Read Single Date from FPGA to PC:

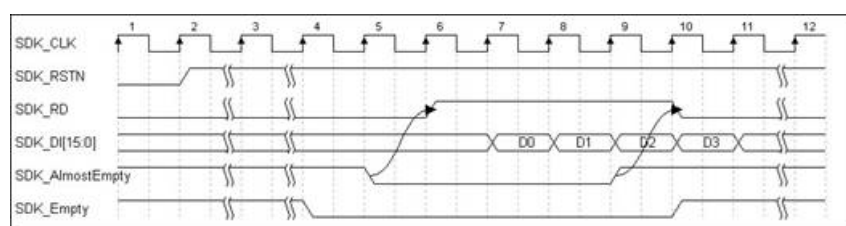
VeriLink Function : [smMatVLXSP6 \(0, 'Read', 1\)](#)

The timing diagram is shown as follows. When SDK_Full signal is low, sends out the SDK_WR and SDK_DO data in the same time.



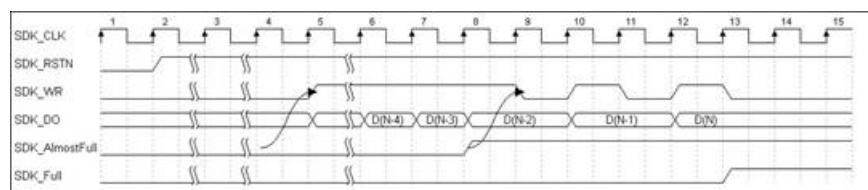
■ Write Multi Data from PC to FPGA:

VeriLink Function : [smMatVLXSP6\(0, 'Write', WriteBuf, 4\)](#)



■ Read Multi Date from FPGA to PC:

VeriLink Function : [smMatVLXSP6 \(0, 'Read', 5\)](#)



PIN Map Information

Note : Please set those pins IO standards to be **LVC MOS33** for better performance. You may refer to ucf file in the SDK/example directory

(XC6SLX16/XC6SLX25)

Signal Name	Pin Name		Signal Name	Pin Name	
SDK_CLK	F2	Input	SDK_CH[0]	P2	Input
SDK_RSTN	K3	Input	SDK_CH[1]	P1	Input
SDK_WR	P15	Output	SDK_CH[2]	R2	Input
SDK_RD	P16	Output	SDK_CH[3]	P4	Input
SDK_Full	L3	Input	SDK_CH[4]	T4	Input
SDK_Empty	M5	Input	SDK_CH[5]	N5	Input
SDK_AlmostFull	L1	Input	SDK_CH[6]	T5	Input
SDK_AlmostEmpty	M4	Input	SDK_CH[7]	M6	Input
SDK_Interrupt	C15	Output			
SDK_DI [0]	F1	Input	SDK_DO [0]	N4	Output
SDK_DI [1]	G3	Input	SDK_DO [1]	N3	Output
SDK_DI [2]	G1	Input	SDK_DO [2]	R1	Output
SDK_DI [3]	H3	Input	SDK_DO [3]	P5	Output
SDK_DI [4]	H2	Input	SDK_DO [4]	R5	Output
SDK_DI [5]	H1	Input	SDK_DO [5]	T6	Output
SDK_DI [6]	H5	Input	SDK_DO [6]	L8	Output
SDK_DI [7]	J3	Input	SDK_DO [7]	T8	Output
SDK_DI [8]	J1	Input	SDK_DO [8]	R12	Output
SDK_DI [9]	K5	Input	SDK_DO [9]	T12	Output
SDK_DI [10]	K6	Input	SDK_DO [10]	T14	Output
SDK_DI [11]	J4	Input	SDK_DO [11]	T13	Output
SDK_DI [12]	K2	Input	SDK_DO [12]	R14	Output
SDK_DI [13]	K1	Input	SDK_DO [13]	T15	Output
SDK_DI [14]	L4	Input	SDK_DO [14]	R15	Output
SDK_DI [15]	L5	Input	SDK_DO [15]	R16	Output

Example

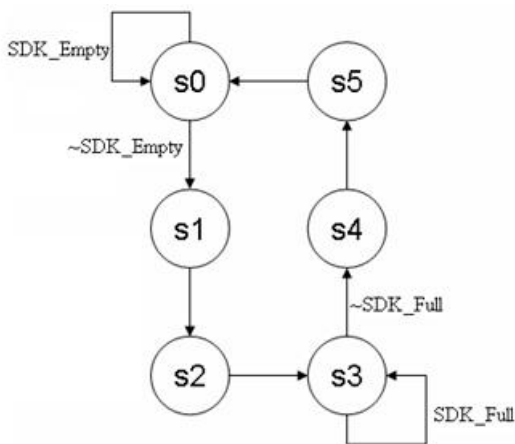
Example Inverter

Example Description: (example\Inverter)

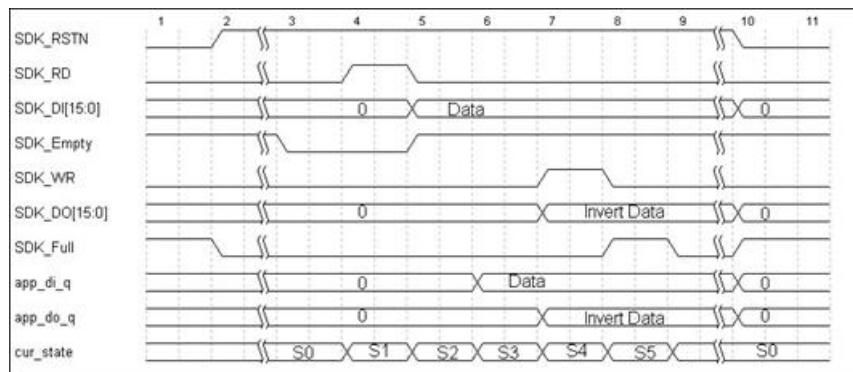
This example, Inverter, show how to use the VeriLink Function to design the Matlab program working with user's circuit. There are hardware and software two parts.

Hardware

The hardware design (single read/write) is to read data from Input FIFO, pass through an inverter, and then output the data to the Output FIFO. Here use a FSM (Finite state machine) to implement it. The state diagram is shown as follows.



The detailed timing diagram is shown in the following figure.



Software

The software part runs the script file in Matlab. This example uses the least SMIMS function to explain how to use the function. It is a simple and quick introduction. The software source code Matlab script file is TestVLXSP6SDK.m.

Software program explanation (TestVLXSP6SDK.m)

```
% Test run SMIMS VLXSP6 SDK
```

```
Clear
```

```
%Program .bit file to FPGA
```

```
smMatVLXSP6 (0, 'Program','c:\app_example.bit');
```

```
% Open FPGA Board and set serial number.
```

```
smMatVLXSP6 (0, 'Open', 'xxx-xxx-xxx-xxx-xxx-xxx');
```

```
for i=1:10
```

```
    writebuf(i)=uint16(i);
```

```
end
```

```
%Write to FPGA Board
```

```
smMatVLXSP6 (0, 'Write', writebuf, 10);
```

```
%Read From FPGA Board
```

```
[ret readbuf] = smMatVLXSP6 (0, 'Read', 10);
```

```
disp(readbuf);
```

```
%close FPGA Board
```

```
smMatVLXSP6(0, 'Close');
```

Run and see the result as follows.

This circuit does invert, so the input data 1 to 10 will get the result 65534 to 65525.

```
>> TestVLXSP6SDK
```

```
65534
```

```
65533
```

```
65532
```

```
65531
```

```
65530
```

```
65529
```

```
65528
```

```
65527
```

```
65526
```

```
65525
```

```
>>
```

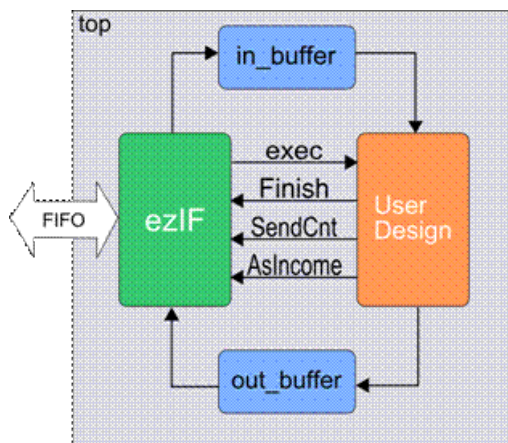
ezIF Interface

Introduce

Interface Description: (example\ Matlab\ezIF)

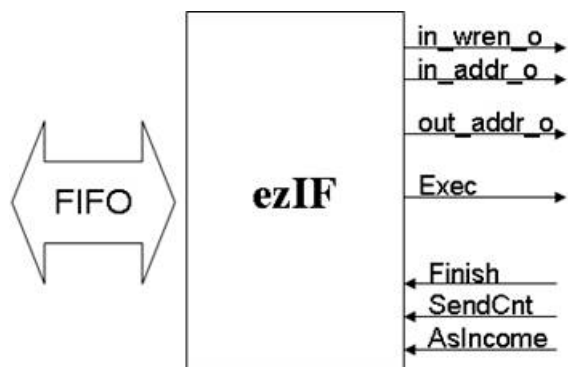
The ezIF is an interface to isolate users' design and FIFO interface. It can simplify user's design flow. The data received from software will be written in the memory by the ezIF. After ezIF writes all data in the memory, it will send a signal to info the User Design starts to work. After the User Design finished the process, it needs to send a signal to info the ezIF and a signal to tell whether to send data back to software side or not. The ezIF will automatically send user data back to the software side. The integration of ezIF and user design is shown as follow picture.

The ezIF is just an example to help user the combine the design with SMIMS development interface. If user needs more efficient or flexible operation mode, please design another mechanism to replace the ezIF.



Interface Operation

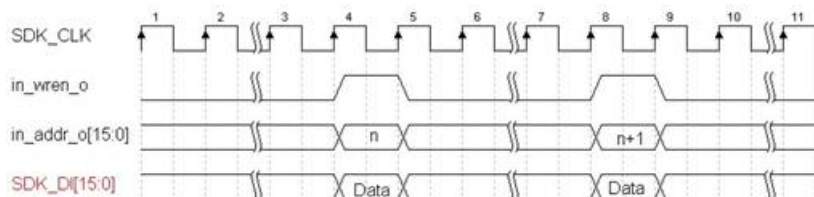
The interface operation of ezIF shows as follow diagram. The FIFO side is skipped. Here just introduces the right side signals.



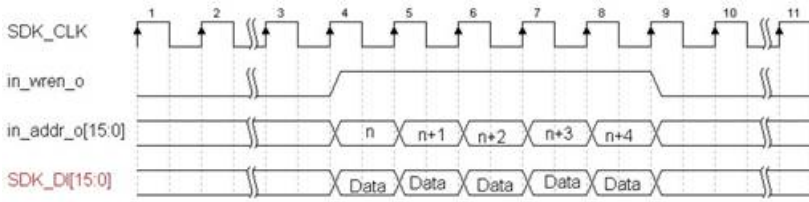
Signal Name	IO Type	Functional description
in_wren_o	output	Memory writing signal. 1 means to write data to the memory.
in_addr_o	output	Memory writing address
out_addr_o	output	Memory reading address for sending data to PC side.
Exec	output	Starts the user design
Finish	input	Info the user design finished the process.
SendCnt	input	The data count of sending data back to PC.
AsIncome	input	If the sending back data has the same counts as received data, then set this signal be 1.

Timing

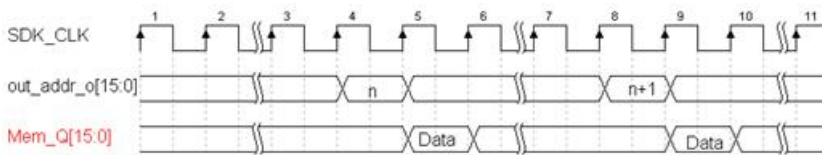
- ezIF writes single data to the memory (the data source is SDK_DI of the FIFO).



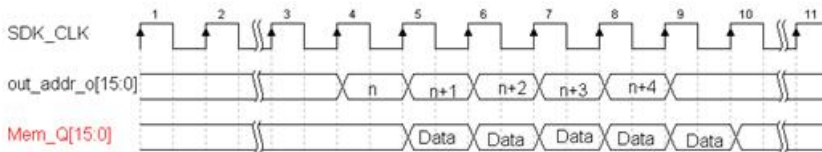
- ezIF writes burst data to the memory (the data source is SDK_DI of the FIFO).



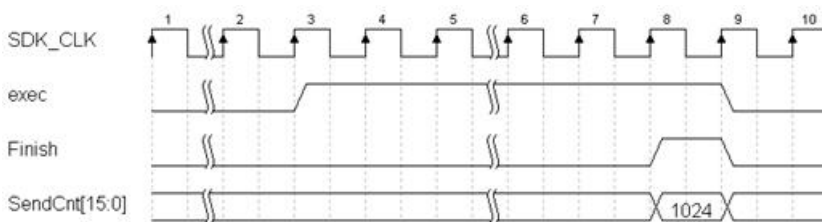
- ezIF reads single data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_DO).



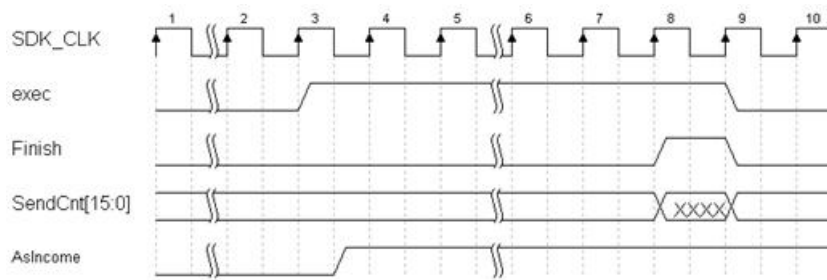
- ezIF reads burst data from the memory and sends back to software side. (Mem_Q is the out put data from memory, user needs to connect it with SDK_DO).



After the ezIF finished the receiving date, it will keep the Exec signal to be 1. The user design receives this signal will start to operate until the circuit finished the process, it will send out a Finish signal. Thus the Exec signal will be 0. When the user design sends out the Finish signal, the SendCnt should be set at the same time with how many data counts needs to send back to the software side. Therefore, the ezIF will follow the signals to send the data back.



If the sending back data has the same counts as received data, then set this signal be 1. In the mean time, the SendCnt will be ignored.



Software Control Interface

In the software side, user needs to set up the data counts before sending data. Before setting the data counts, the SDK_CH of FIFO needs to be 1. After setting the data counts, set the SDK_CH back to 0. The source code shown as follows. The important is that the settings needs a lot of time. Only if every time sends different data counts, otherwise please do not set it before every sending. The ideal way is setting once and then sends data continuously, until the data count is different then setup the new data count.

```
smMatVLXSP6(0, 'ChSel', 1);
Cmd = data counts
smMatVLXSP6(0, 'Write', Cmd,1);
smMatVLXSP6(0, 'ChSel', 0);
```

Example Code

```
% Test run SMIMS VLXSP6 SDK
close all;
clear all;
```

```
rand('seed',0);
```

```
% Open FPGA Board and set serial number.
```

```
smMatVLXSP6(0, 'Open', 'xxxx-xxxx-xxxx-xxxx-xxxx-xxxx');
```

```
for i=1:4095
```

```
    smMatVLXSP6(0, 'ChSel', 1);
```

```
    Cmd = i;
```

```
    smMatVLXSP6(0, 'Write', Cmd,1);
```

```
    smMatVLXSP6(0, 'ChSel', 0);
```

```
    fprintf('Send to HW Data Count : %d    ',i);
```

```
    flag = 1;
```

```
    // generate the random data
```

```
    WriteBuffer=rem(round(4096*rand(1,i)),4096);
```

```
    // send data to the circuit
```

```
    smMatVLXSP6(0, 'Write', WriteBuffer,i);
```

```
    // receive data from the circuit
```

```
    [ret ReadBuffer] = smMatVLXSP6(0, 'Read', i);
```

```
    // check the operation result
```

```
    for j=1:i
```

```
        if ReadBuffer(j) ~= (4096-WriteBuffer(j))
```

```
            flag = 0;
```

```
        end
```

```
    end
```

```
    if(flag == 1)
```

```
        fprintf('Check ok...\n');
```

```
    else
```

```
        fprintf('Error !\n');
```

```
        break;
```

```
    end
```

```
end
```

```
smMatVLXSP6(0, 'Close');
```


Contact Information

Please contact our technical support through one of the methods below.

■ Telephone

Call our support hotline at 886-6-2091821 from 9:00 am to 6:00 pm (Taipei time GMT+8:00), Monday to Friday.

■ Instant Messaging

For real-time technical assistance, contact us through MSN. Our support account name is service@smims.com. Official support hours are from 9:00 am to 6:00 pm (Taipei time GMT+8:00), Monday to Friday.

■ E-mail

You may also send any queries to our email address sevice@smims.com