XILINX.

Port-Level I/O Protocols

2021.1

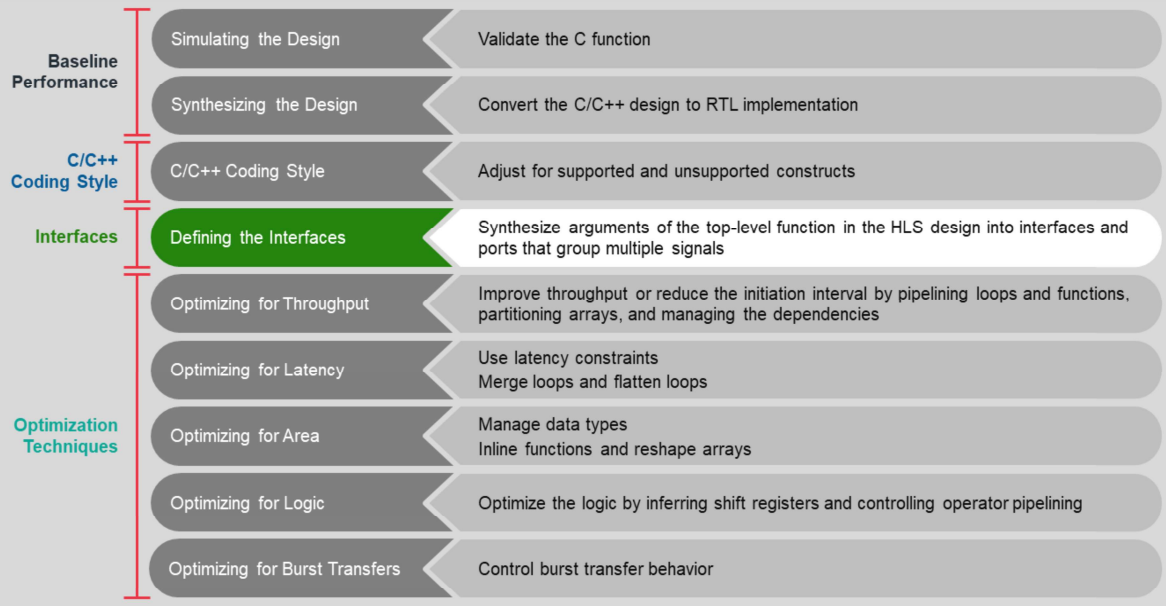
Objectives

After completing this module, you will be able to:

- List the types of port-level protocols abstracted in the Vitis™ HLS tool

HLS Design Methodology

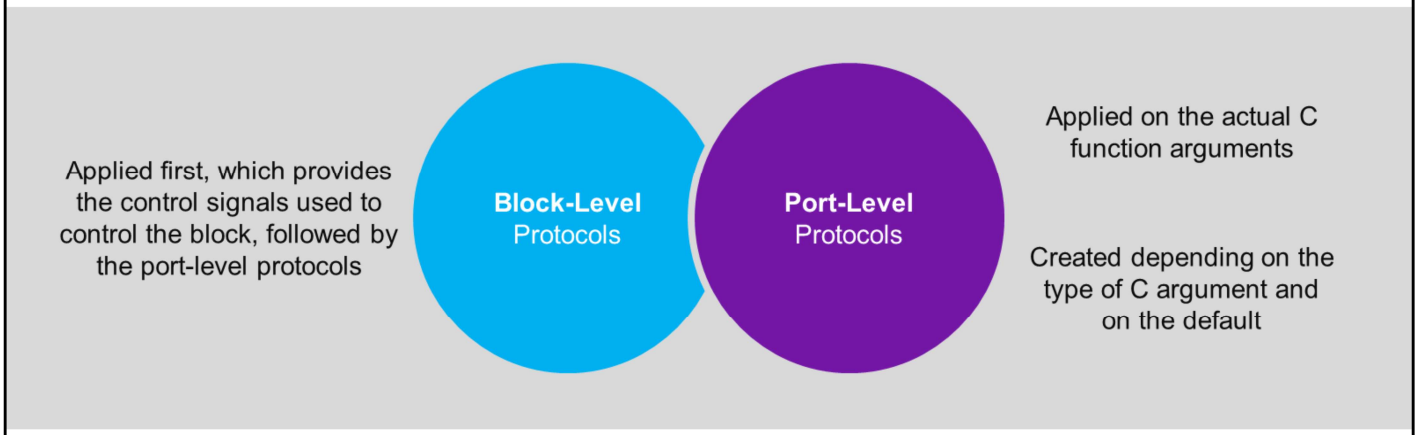
Once simulation and synthesis of a design are performed successfully, the first step is defining the interfaces



XILINX.

The first step is to define the interfaces.

HLS Design Methodology



4

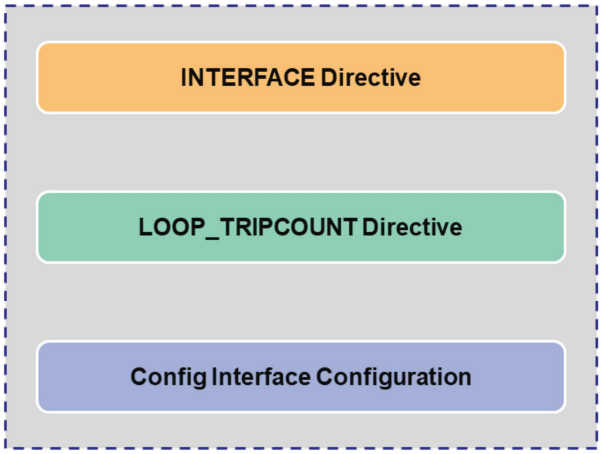
XILINX.

The interfaces are defined as the block-level and port-level protocols for the design. The block-level protocols are applied first, which provides the control signals used to control the block, followed by the port-level protocols.

A port-level I/O protocol is applied on the actual C function arguments. This protocol is created depending on the type of C argument and on the default.

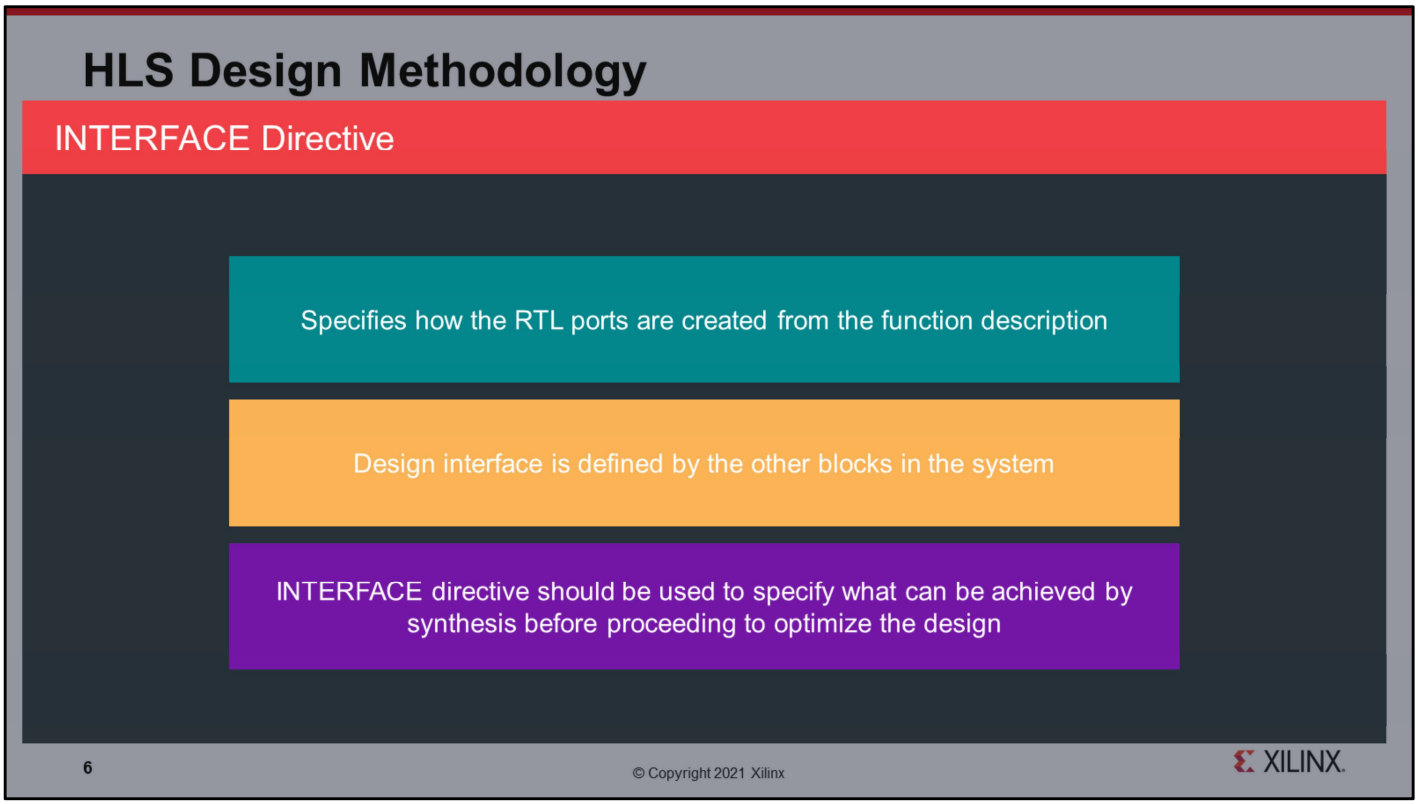
HLS Design Methodology

Directives or configurations used in the “Define Interfaces” step



The directives or configuration used in this step are:

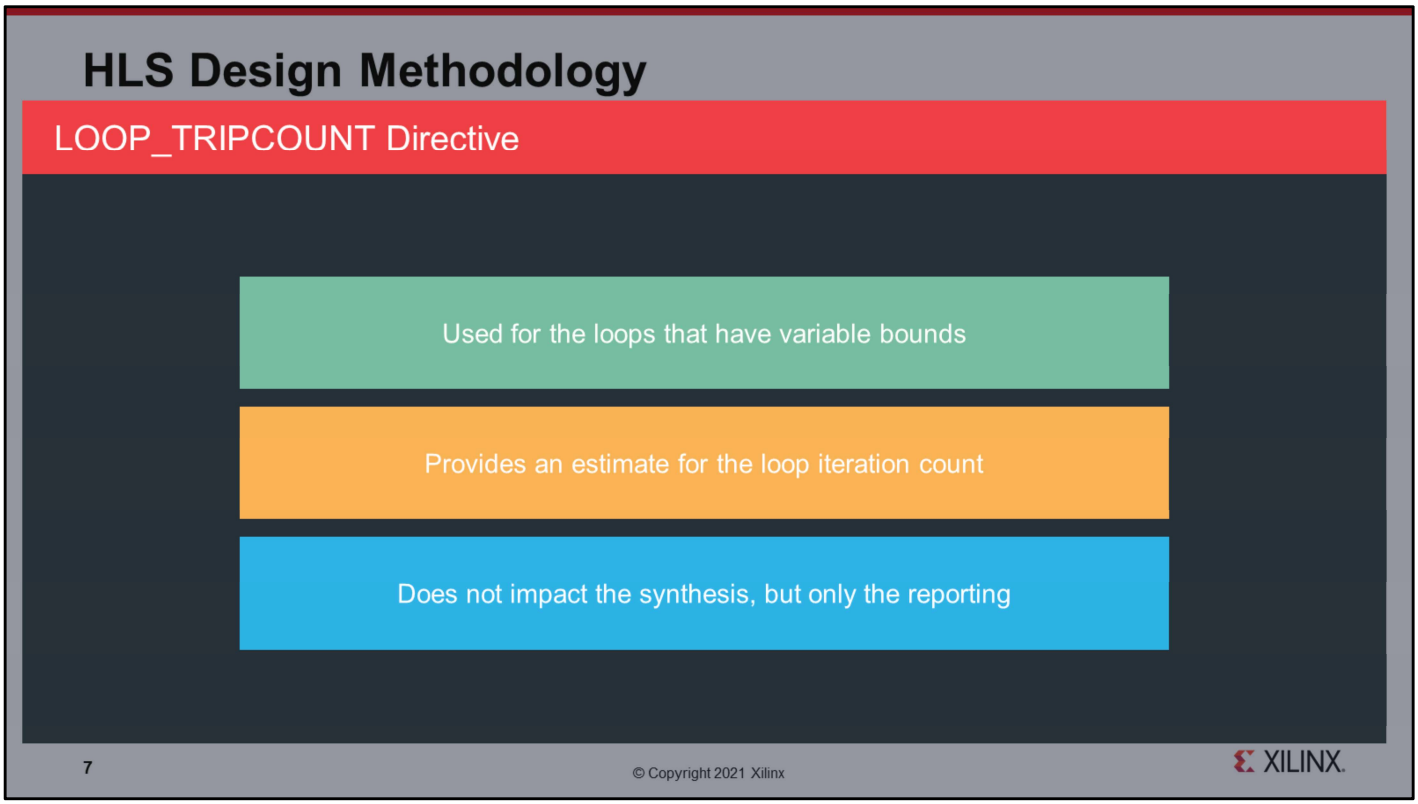
- INTERFACE directive
- LOOP_TRIPCOUNT directive, and
- Config Interface configuration



INTERFACE Directive

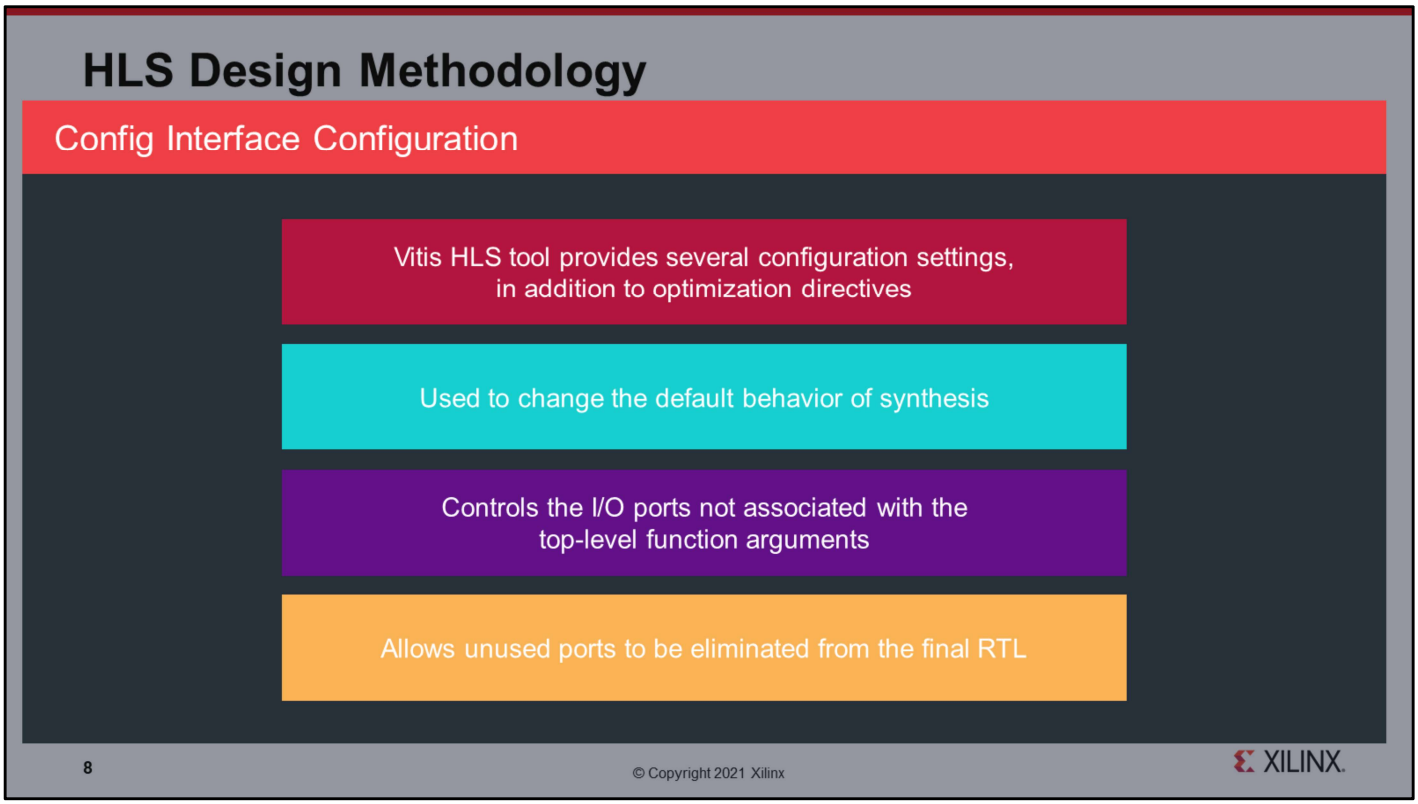
It specifies how the RTL ports are created from the function description. The design interface is typically defined by the other blocks in the system.

Since the type of I/O protocol helps determine what can be achieved by synthesis, it is recommended that the INTERFACE directive be used to specify this before proceeding to optimize the design.



LOOP_TRIPCOUNT Directive

It is used for the loops that have variable bounds. It provides an estimate for the loop iteration count. It has no impact on the synthesis, but only on the reporting.

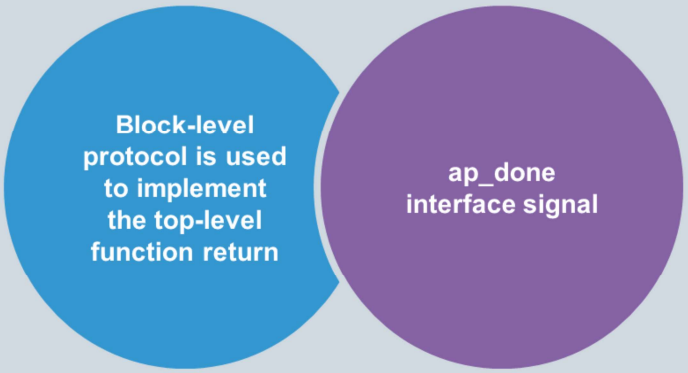


Config Interface Configuration

In addition to the optimization directives, the Vitis HLS tool provides several configuration settings, which are used to change the default behavior of synthesis.

The Config Interface configuration controls the I/O ports, which are not associated with the top-level function arguments and allows unused ports to be eliminated from the final RTL.

Port-Level I/O Protocols



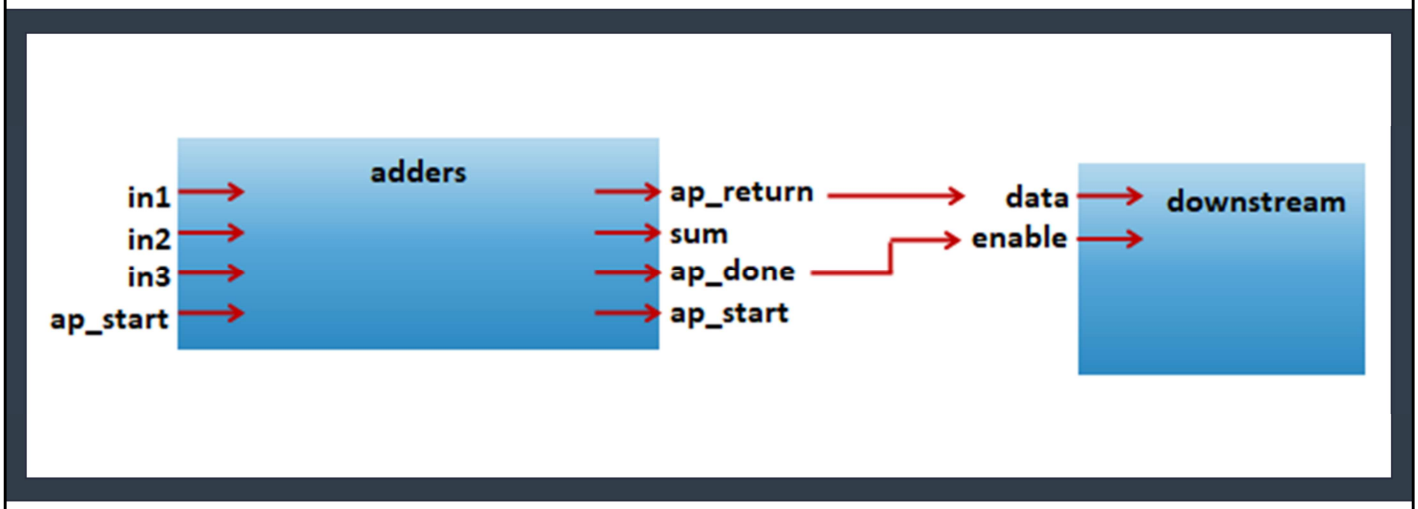
Goes high to indicate that function return is valid
Allows the downstream blocks to correctly sample the output port

XILINX.

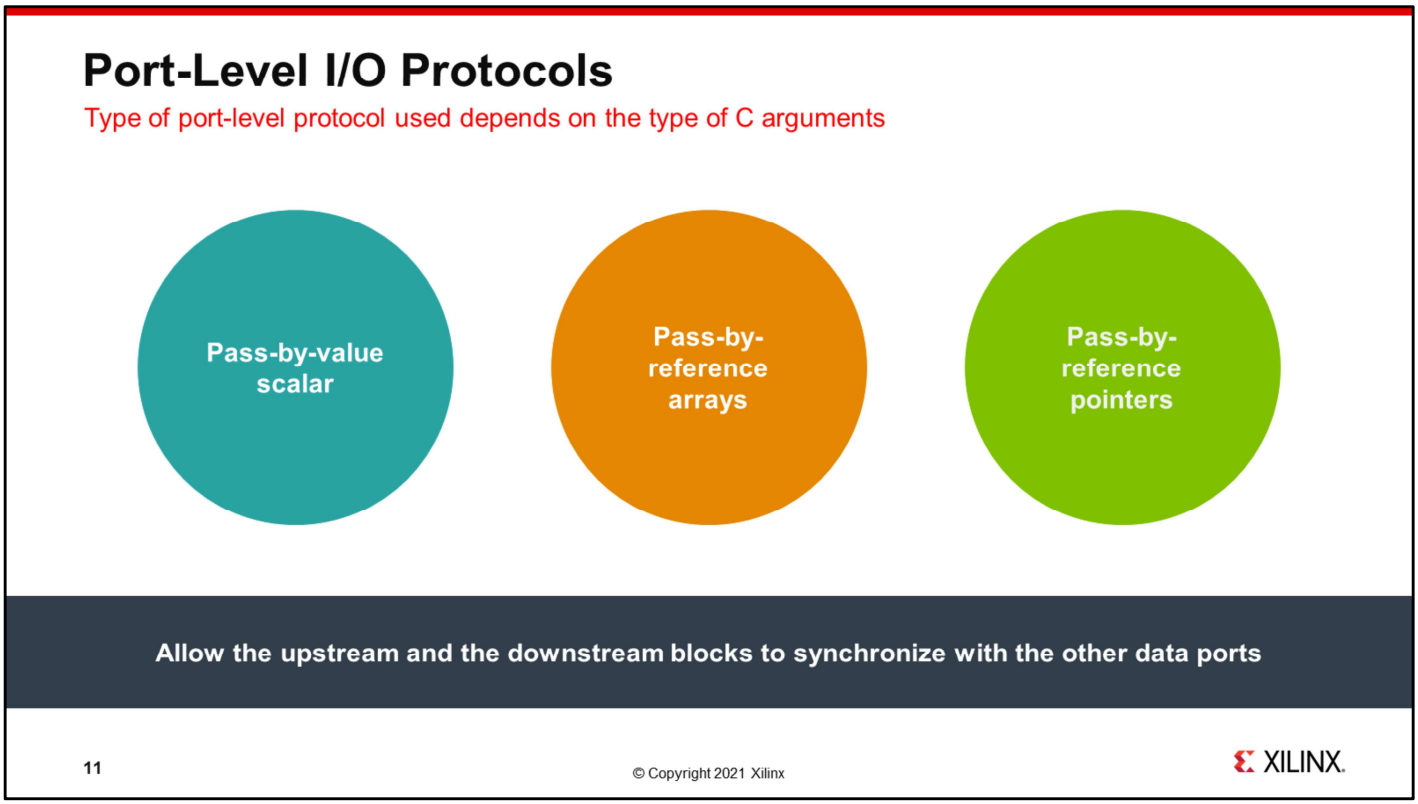
The block-level protocol is used to implement the top-level function ‘return’ through the ap_done interface signal.

This signal goes high to indicate that the function return is valid and allows the downstream blocks to correctly sample the output port.

Port-Level I/O Protocols



In the adders design module shown here, the block-level protocols such as ap_start, ap_return etc. has been implemented. The ap_done signal goes high to indicate the output is valid. Now, let's see how the data ports will be implemented using the port-level protocols.



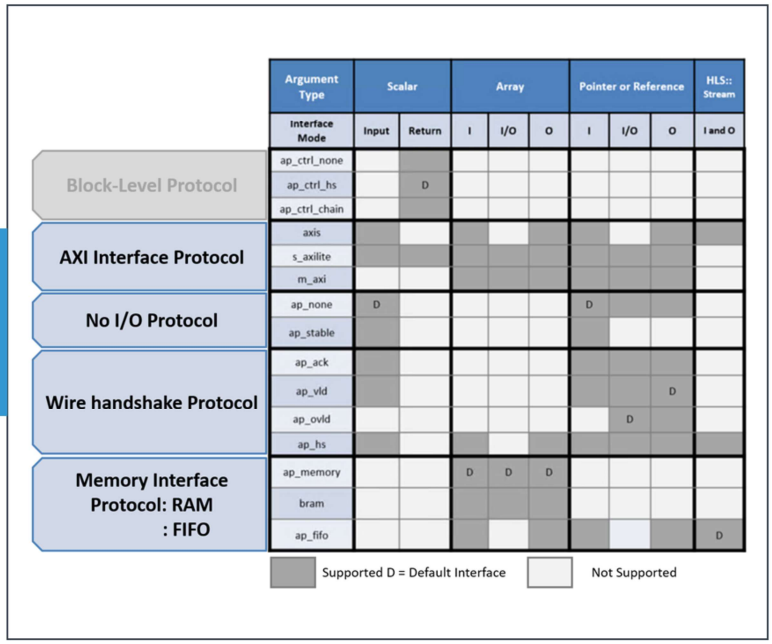
The type of port-level protocol used depends on the type of C arguments as follows:

- Pass-by-value scalar
- Pass-by-reference arrays
- Pass-by-reference pointers

These protocols allow the upstream and the downstream blocks to synchronize with the other data ports.

Interface Types

- **Every combination** of C argument and port protocol is ***not*** supported
- **Code modification** may be required to implement a specific I/O protocol



XILINX.

Here is the complete list of all possible port-level I/O protocols:

- AXI4 Interface protocols
- No I/O protocol
- Wire handshake protocol
- Memory interface protocol

In all these port-level I/O protocols, there are multiple interface protocols that are available.

Every combination of C argument and port protocol is not supported.

Code modification may be required to implement a specific I/O protocol.

Interface Types

Adder example:

Sum is a pointer which can read and write data

I/O protocol created for this pointer depends on the type of C argument and on the default interface type chosen by the HLS tool

```
# include “adders.h”

int adders(int in1, int in2, int *sum) {
      int temp;

     *sum = in1 + in2 + *sum;
      temp = in1 + in2;

      return temp;
}
```


XILINX.

The example code shown is a simple adder where the sum is a pointer which can read and write data.

The I/O protocol created for this pointer depends on the type of C argument and on the default interface type chosen by HLS tool.

Interface Types

Default interface type for the sum port will be type ap_ovld

```
# include "adders.h"
int adders(int in1, int in2, int *sum) {
    int temp;

    *sum = in1 + in2 + *sum;
    temp = in1 + in2;

    return temp;
}
```

AXI Interface Protocol

No I/O Protocol

Wire Handshake Protocol

Argument	Scalar		Array			Pointer or Reference		
	pass-by-value		pass-by-reference			pass-by-reference		
Interface Mode	Input	Returns	I	IO	O	I	IO	O
ap_ctrl_none								
ap_ctrl_hs		D						
ap_ctrl_chain								
axis								
s_axilite								
m_axi								
ap_none	D					D		
ap_stable								
ap_ack								
ap_vld								D
ap_ovld							D	
ap_hs								D
ap_memory			D	D	D			
bram								
ap_fifo								

Supported D = Default

NOT Supported

XILINX.

This sum port in the code can be any of these interface types:

- AXI interface protocol
    - s_axilite
    - m_axi
- No I/O protocol
    - ap_none
- Wire handshake protocol
    - ap_ack
    - ap_vld
    - ap_ovld (default)
    - ap_hs

The default interface type for the sum port will be type ap_ovld.

Default I/O Protocols

For every type of C argument in the function code, there is a default I/O protocol associated with it

C Argument Type	Default I/O Protocol
Input	ap_done
Output	ap_vld
Inout	ap_ovld
In port of inout Out port of inout	ap_none ap_vld
Arrays	ap_memory

For every type of C argument in the function code, there is a default I/O protocol associated with it, as shown in the table.

Default I/O Protocols

By default

Input pass-by-value arguments and pointers are implemented as simple wire ports with no associated handshaking signal

- If the port has no I/O protocol, the input data must be held stable until it is read

Output pointers are implemented with an associated output valid signal to indicate when the output data is valid

- Output writes have an accompanying output valid signal that can be used to validate them

Arrays will default to RAM interfaces (two-port RAM is the default RAM)

16

XILINX.

By default, the input pass-by-value arguments and pointers are implemented as simple wire ports with no associated handshaking signal. If the port has no I/O protocol, the input data must be held stable until it is read.

By default, the output pointers are implemented with an associated output valid signal to indicate when the output data is valid. If there is no I/O protocol associated with the output port, it is difficult to know when to read the data.

The output writes have an accompanying output valid signal that can be used to validate them.

Arrays will default to RAM interfaces.

Default I/O Protocols

HLS tool shell/console always shows the results of interface synthesis

```
@I [RTGEN-500] Setting IO mode on port 'adders|in1' to 'ap_none'.
@I [RTGEN-500] Setting IO mode on port 'adders|in2' to 'ap_none'.
@I [RTGEN-500] Setting IO mode on port 'adders|in3' to 'ap_none'.
@I [RTGEN-500] Setting IO mode on port 'adders|in1' to 'ap_ctrl_hs'.
```


XILINX.

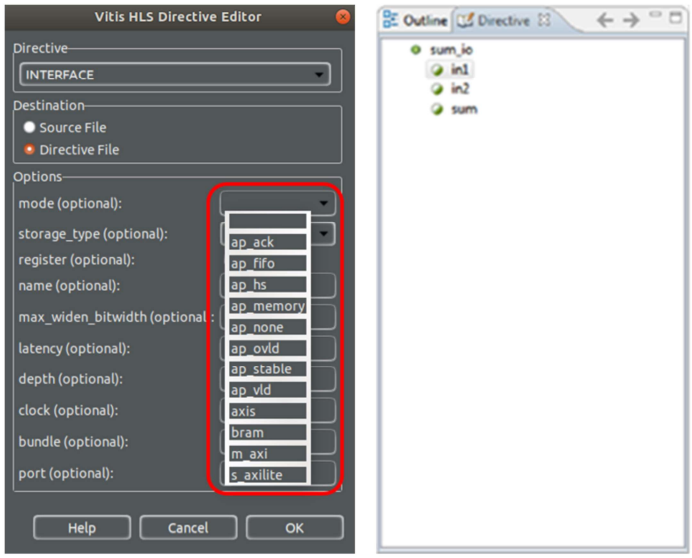
The HLS tool shell/console always shows the results of interface synthesis.

Specifying I/O Protocols: INTERFACE Directive

User can specify the interface type of their choice. The tool can then overwrite the interface type which was selected by default

Steps to specify the I/O protocol:

- Select the port in the Directives pane to specify a protocol
- Right-click and select Add Directives e
- Select INTERFACE as the directive and select the desired mode from the drop-down list



XILINX.

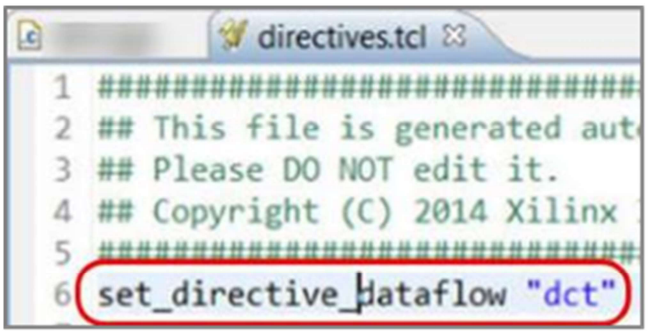
The HLS tool allows the user to specify the interface type of their choice. The tool can then overwrite the interface type selected by default.

Here are the steps to specify the I/O protocol:

- Select the port in the Directives pane to specify a protocol
- Right-click and select Add Directives
- Select INTERFACE as the directive and select the desired mode ~~protocol~~ from the drop-down list

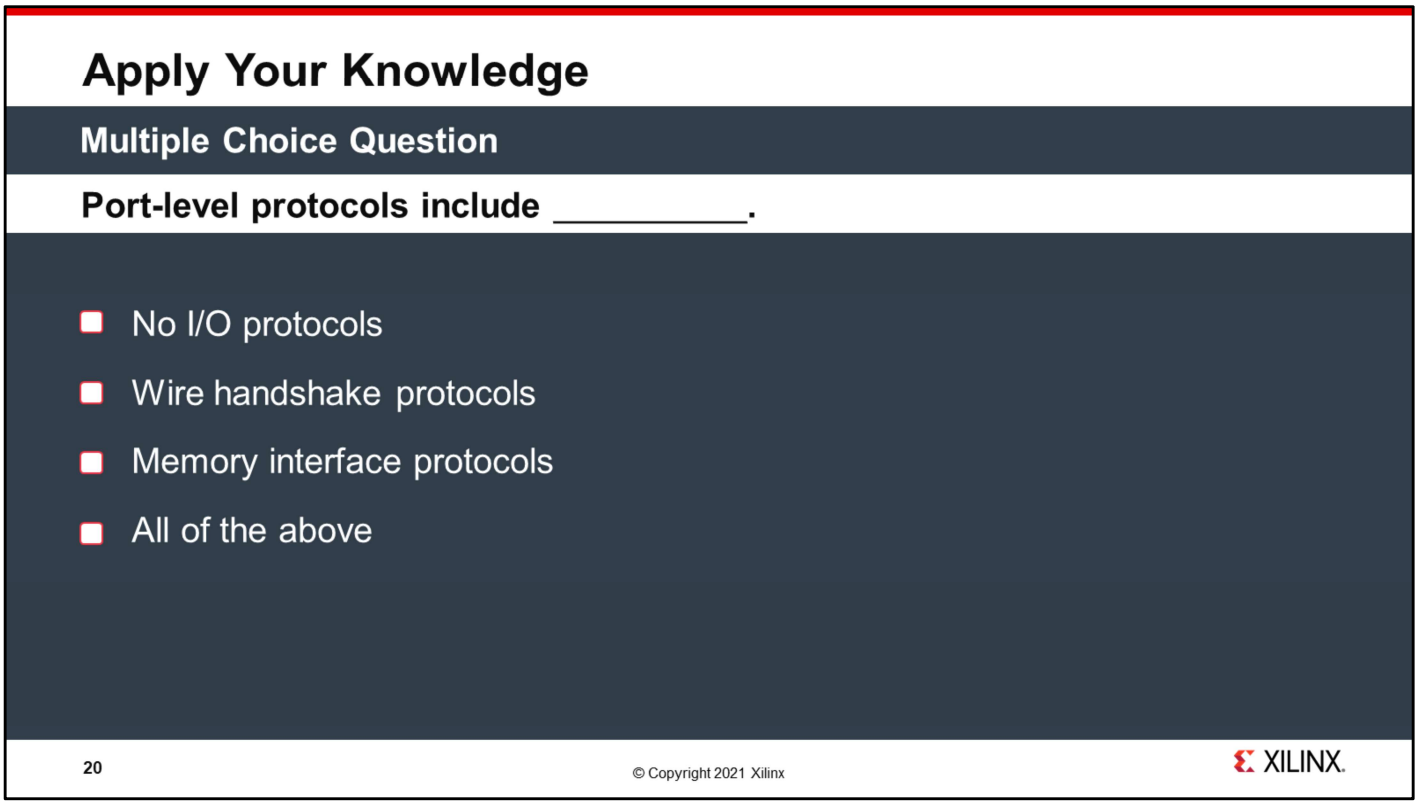
Specifying I/O Protocols: INTERFACE Directive

Specify a protocol using pragma or Tcl command



XILINX.

OR specify a protocol using a pragma or Tcl command.



Correct answer:
All of the above

Correct feedback:
You selected the correct answer

Incorrect feedback:
The correct answer is “All of the above”.

Try again feedback:
Hint: Recall all the types of port-level protocols.

Apply Your Knowledge

True or False Question

Every combination of C argument and port-level protocol is supported.

- ☐ True
- ☐ False

XILINX.

Correct answer:
False

Correct feedback:
You selected the correct answer.

Incorrect feedback:
The correct answer is “False”.

Summary

- Port-level I/O protocols are used to sequence data into and out of the block
- The I/O protocol created depends on the type of C argument and the INTERFACE directive
- Different types of port-level protocols:
  - AXI interface protocols
  - No I/O protocol
  - Memory interface protocols (RAM, FIFO)
  - Wire handshake protocols