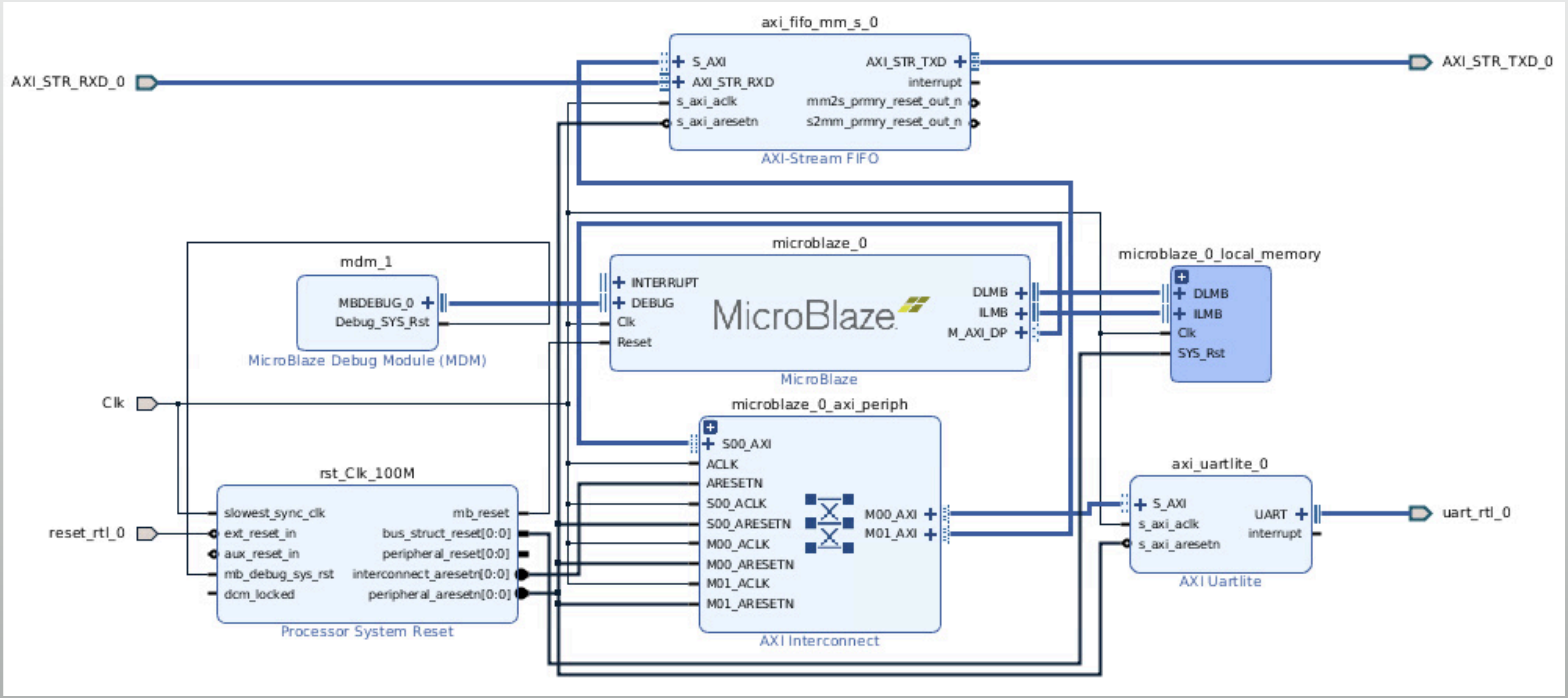
Reconfigurable Architecture (11)

High-Level Synthesis (2 of 2: Microblaze Integration)

Previously in this class…

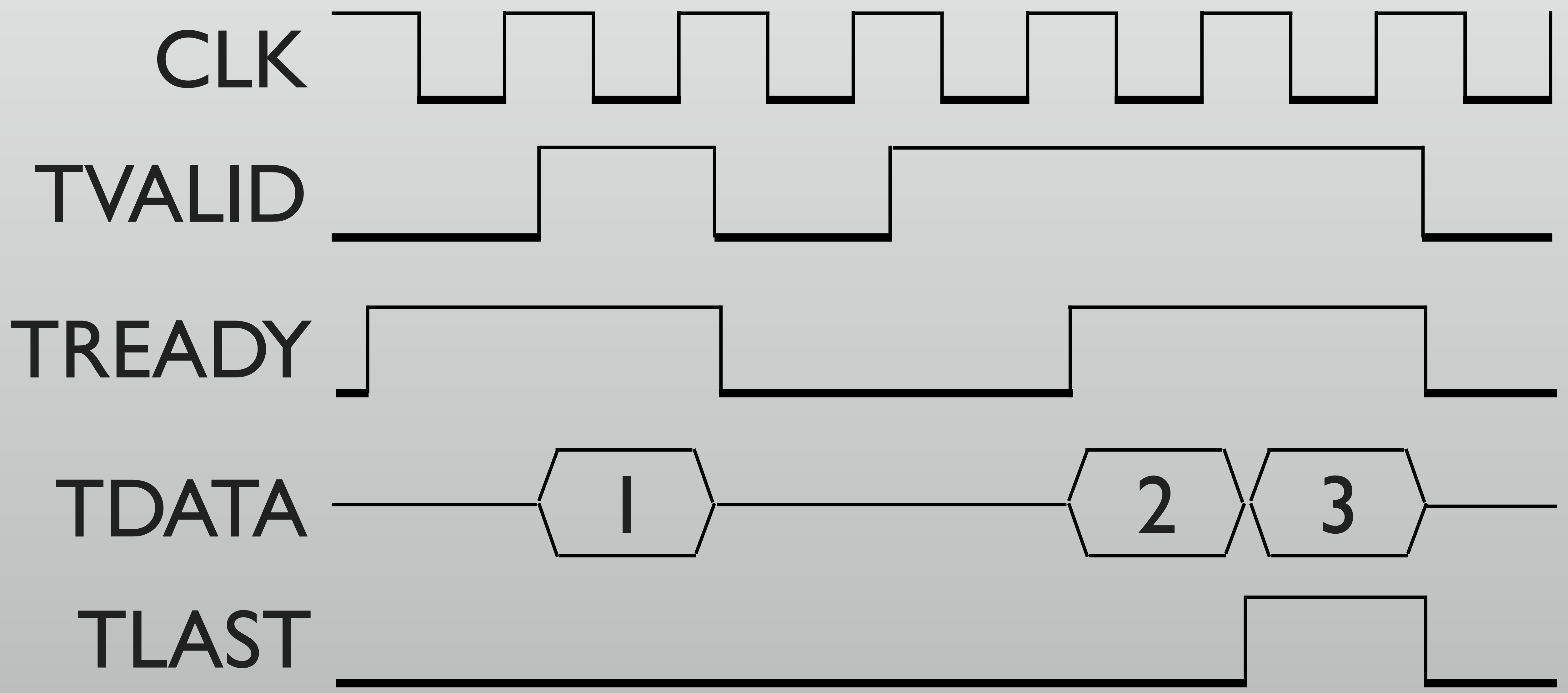
- Microblaze + AXI Stream FIFO
  - Access RTL designed AXI Stream module from MicroBlaze software
- High-Level Synthesis
  - Generate RTL from algorithms described in C/C++
- Today: HLS generated module + MicroBlaze software

MicroBlaze + AXI Stream FIFO



Handshake on AXI Stream

- Transfer on TVALID & TREADY
  - TDATA: Data (src→dest)
  - TVALID: Data valid (src→dest)
  - TREADY: Dest ready (dest→src)
  - TLAST: Last word of stream (src→dest)



Example: 3 word transfer

fifo-example.c

- init() for API initialization
- send() to send data (with length to send)
- recv() to receive (API detects the length of received data frame)
    - Just modify main() for your own AXI stream logic

AXI Stream in Vivado HLS

- hls::stream class provides FIFO object
  - In function, FIFO is synthesized: convenient for streaming buffer
  - On interface, FIFO interface or AXI Stream interface is synthesized
    - Specified by INTERFACE directive
    - Read-only or Write-only port (no bidirectional port is possible)

hls::stream example

- Reads 10 integer from stream a
  - Writes Sum, minimum and maximum value to stream b
- Access by read() and write()
- hls::stream must be passed by reference (with &)

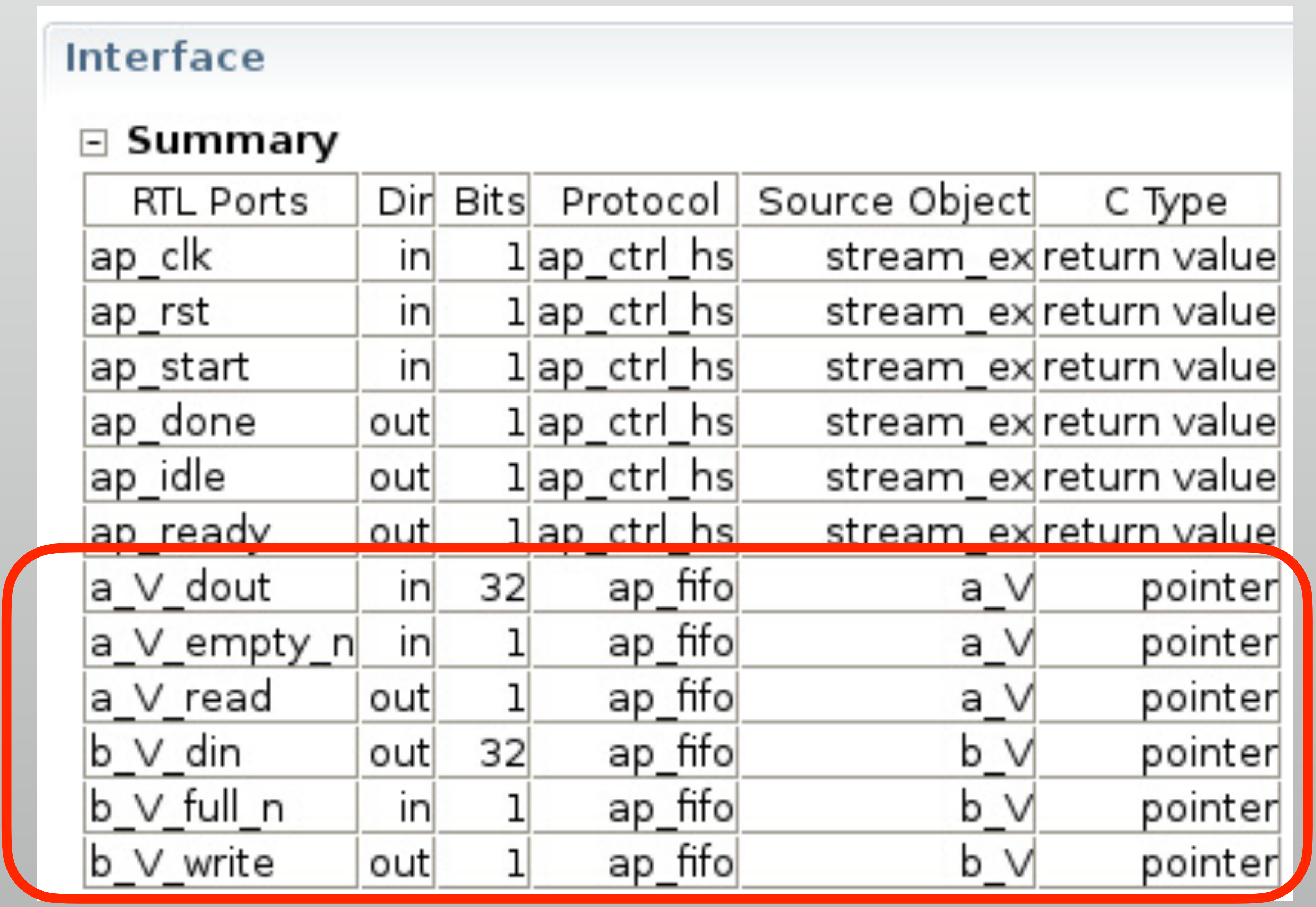
```
#include “hls_stream.h”

void stream_ex( hls::stream<int>& a,
                hls::stream<int>& b ){
  int sum=0, ma, mi;

  for (int i=0; i<10; i++){
    int x = a.read();
    if (i==0){ ma=x; mi=x; }
    if (i!=0 && ma<x) ma=x;
    if (i!=0 && mi>x) mi=x;
    sum += x;
  }

  b.write(sum); b.write(ma); b.write(mi);
}
```

Synthesis result

* Port a and b will be "ap_fifo"
  * Simple FIFO port, not AXI Stream
  * Add pragma to fix:

    (insert just above int sum=0, …)

```
#pragma HLS INTERFACE axis port=a
#pragma HLS INTERFACE axis port=b
```



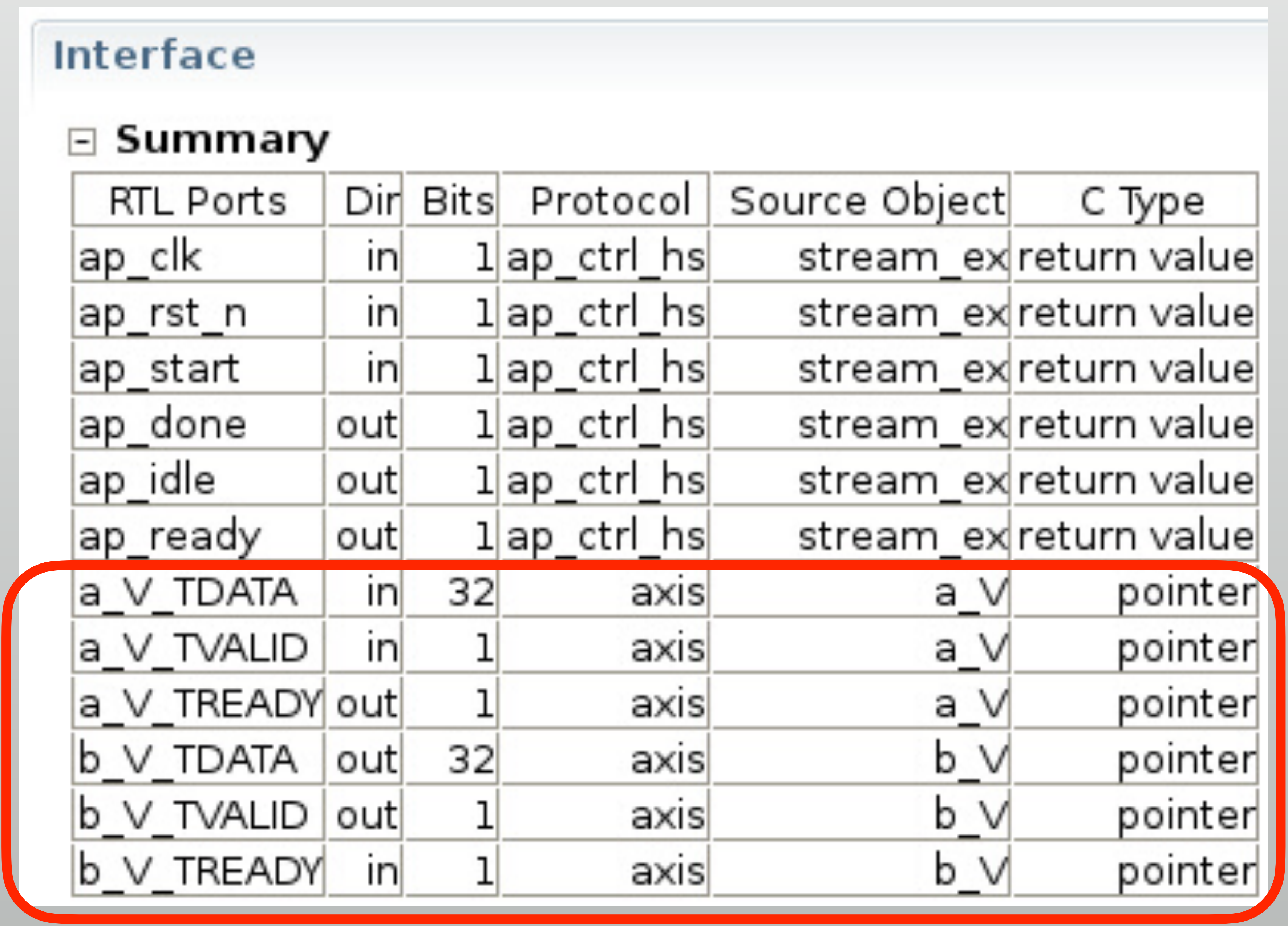
Interface

⊟ Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	stream_ex	return value
ap_rst	in	1	ap_ctrl_hs	stream_ex	return value
ap_start	in	1	ap_ctrl_hs	stream_ex	return value
ap_done	out	1	ap_ctrl_hs	stream_ex	return value
ap_idle	out	1	ap_ctrl_hs	stream_ex	return value
ap_ready	out	1	ap_ctrl_hs	stream_ex	return value
a_V_dout	in	32	ap_fifo	a_V	pointer
a_V_empty_n	in	1	ap_fifo	a_V	pointer
a_V_read	out	1	ap_fifo	a_V	pointer
b_V_din	out	32	ap_fifo	b_V	pointer
b_V_full_n	in	1	ap_fifo	b_V	pointer
b_V_write	out	1	ap_fifo	b_V	pointer

Got AXI Stream ports!

```
void stream_ex( hls::stream<int>& a,
                hls::stream<int>& b ){
#pragma HLS INTERFACE axis port=a
#pragma HLS INTERFACE axis port=b

  int sum=0, ma, mi;
  for (int i=0; i<10; i++){
    int x = a.read();
    if (i==0){ ma=x; mi=x; }
    if (i!=0 && ma<x) ma=x;
    if (i!=0 && mi>x) mi=x;
    sum += x;
  }

  b.write(sum); b.write(ma); b.write(mi);
}
```



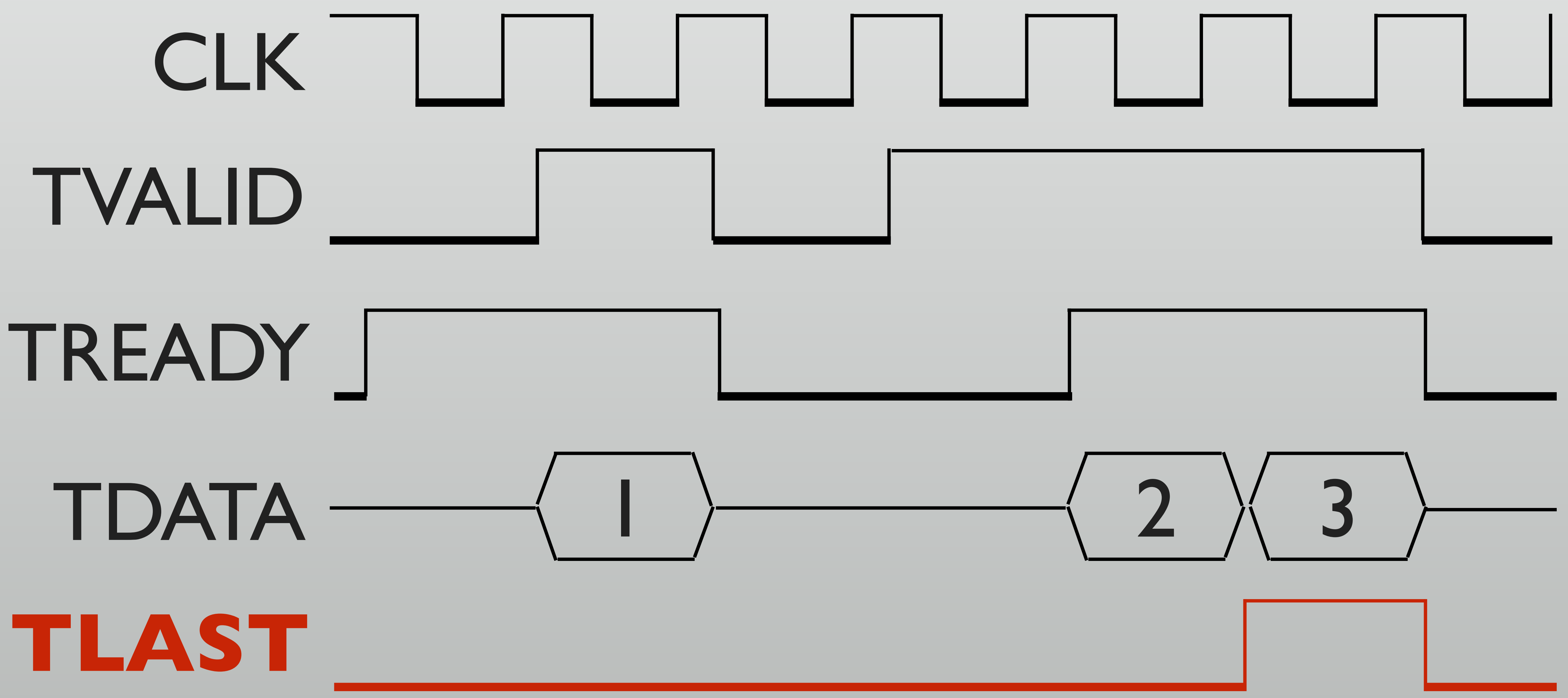
Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	stream_ex	return value
ap_rst_n	in	1	ap_ctrl_hs	stream_ex	return value
ap_start	in	1	ap_ctrl_hs	stream_ex	return value
ap_done	out	1	ap_ctrl_hs	stream_ex	return value
ap_idle	out	1	ap_ctrl_hs	stream_ex	return value
ap_ready	out	1	ap_ctrl_hs	stream_ex	return value
a_V_TDATA	in	32	axis	a_V	pointer
a_V_TVALID	in	1	axis	a_V	pointer
a_V_TREADY	out	1	axis	a_V	pointer
b_V_TDATA	out	32	axis	b_V	pointer
b_V_TVALID	out	1	axis	b_V	pointer
b_V_TREADY	in	1	axis	b_V	pointer

AXI Stream signals again

- Mandatory signals:
  - TDATA, TVALID and TREADY
  - Already in hls::stream design
- Optional sideband signals:
  - **TLAST**, TUSER, TDEST, ….
  - TLAST is required with AXI Stream FIFO



AXI Stream + sideband (official form)

- With `ap_axis` template struct:
  - <Data, User, ID, Dest> for
    - TDATA width
    - TUSER width
    - TID width
    - TDEST width

```
#include "ap_int.h"

template<int D,int U,int TI,int TD>
struct ap_axis{
     ap_int<D>    data;
     ap_uint<D/8> keep;
     ap_uint<D/8> strb;
     ap_uint<U>   user;
     ap_uint<1>   last;
     ap_uint<TI>  id;
     ap_uint<TD>  dest;
};
```

See Xilinx UG902 for detail

ap_axis example

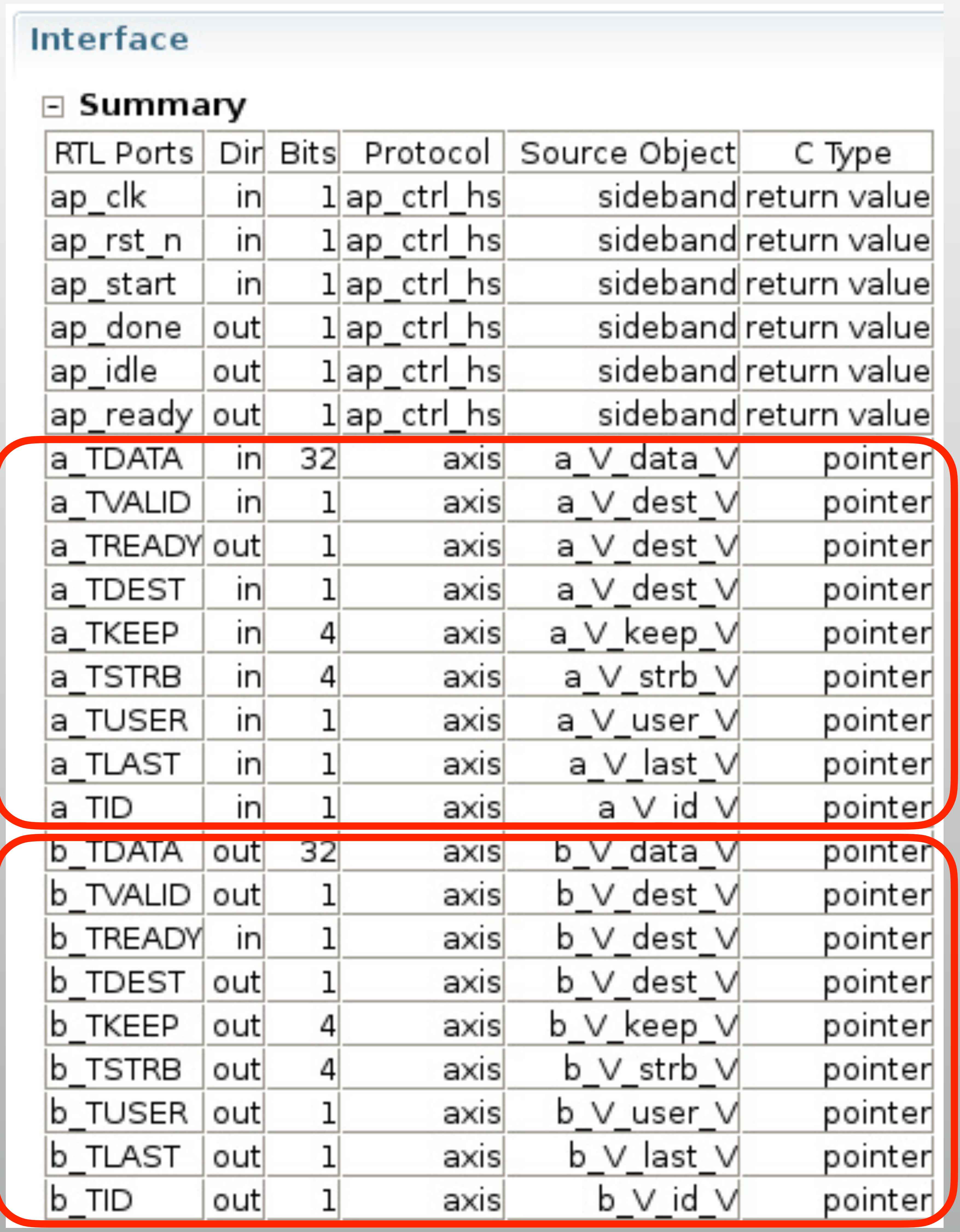
```
#include "hls_stream.h"
#include "ap_axi_sdata.h"

void sideband
  (hls::stream<ap_axis<32,1,1,1> >& a,
   hls::stream<ap_axis<32,1,1,1> >& b ){
#pragma HLS INTERFACE axis port=a
#pragma HLS INTERFACE axis port=b

  ap_axis<32,1,1,1> aa, bb;
  int sum=0, i=0, ma, mi;
```

```
  do {
    aa = a.read();
    int x = aa.data.to_int();
    if (i==0){ ma=x; mi=x; }
    if (i!=0 && ma<x) ma=x;
    if (i!=0 && mi>x) mi=x;
    sum += x;
    i ++;
  } while(aa.last == 0);

  bb.data=sum; bb.last=0;  b.write(bb);
  bb.data=ma;  bb.last=0;  b.write(bb);
  bb.data=mi;  bb.last=1;  b.write(bb);
}
```

Note: TUSER, TID, TDEST width must be non-zero, TLAST is always 1 bit width

Synthesized ports

- Ports a and b has sideband signals
  - Although we only need TLAST with MicroBlaze, `ap_axis` always gives all sideband signals
  - That's overkill…



Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	sideband	return value
ap_rst_n	in	1	ap_ctrl_hs	sideband	return value
ap_start	in	1	ap_ctrl_hs	sideband	return value
ap_done	out	1	ap_ctrl_hs	sideband	return value
ap_idle	out	1	ap_ctrl_hs	sideband	return value
ap_ready	out	1	ap_ctrl_hs	sideband	return value
a_TDATA	in	32	axis	a_V_data_V	pointer
a_TVALID	in	1	axis	a_V_dest_V	pointer
a_TREADY	out	1	axis	a_V_dest_V	pointer
a_TDEST	in	1	axis	a_V_dest_V	pointer
a_TKEEP	in	4	axis	a_V_keep_V	pointer
a_TSTRB	in	4	axis	a_V_strb_V	pointer
a_TUSER	in	1	axis	a_V_user_V	pointer
a_TLAST	in	1	axis	a_V_last_V	pointer
a_TID	in	1	axis	a_V_id_V	pointer
b_TDATA	out	32	axis	b_V_data_V	pointer
b_TVALID	out	1	axis	b_V_dest_V	pointer
b_TREADY	in	1	axis	b_V_dest_V	pointer
b_TDEST	out	1	axis	b_V_dest_V	pointer
b_TKEEP	out	4	axis	b_V_keep_V	pointer
b_TSTRB	out	4	axis	b_V_strb_V	pointer
b_TUSER	out	1	axis	b_V_user_V	pointer
b_TLAST	out	1	axis	b_V_last_V	pointer
b_TID	out	1	axis	b_V_id_V	pointer

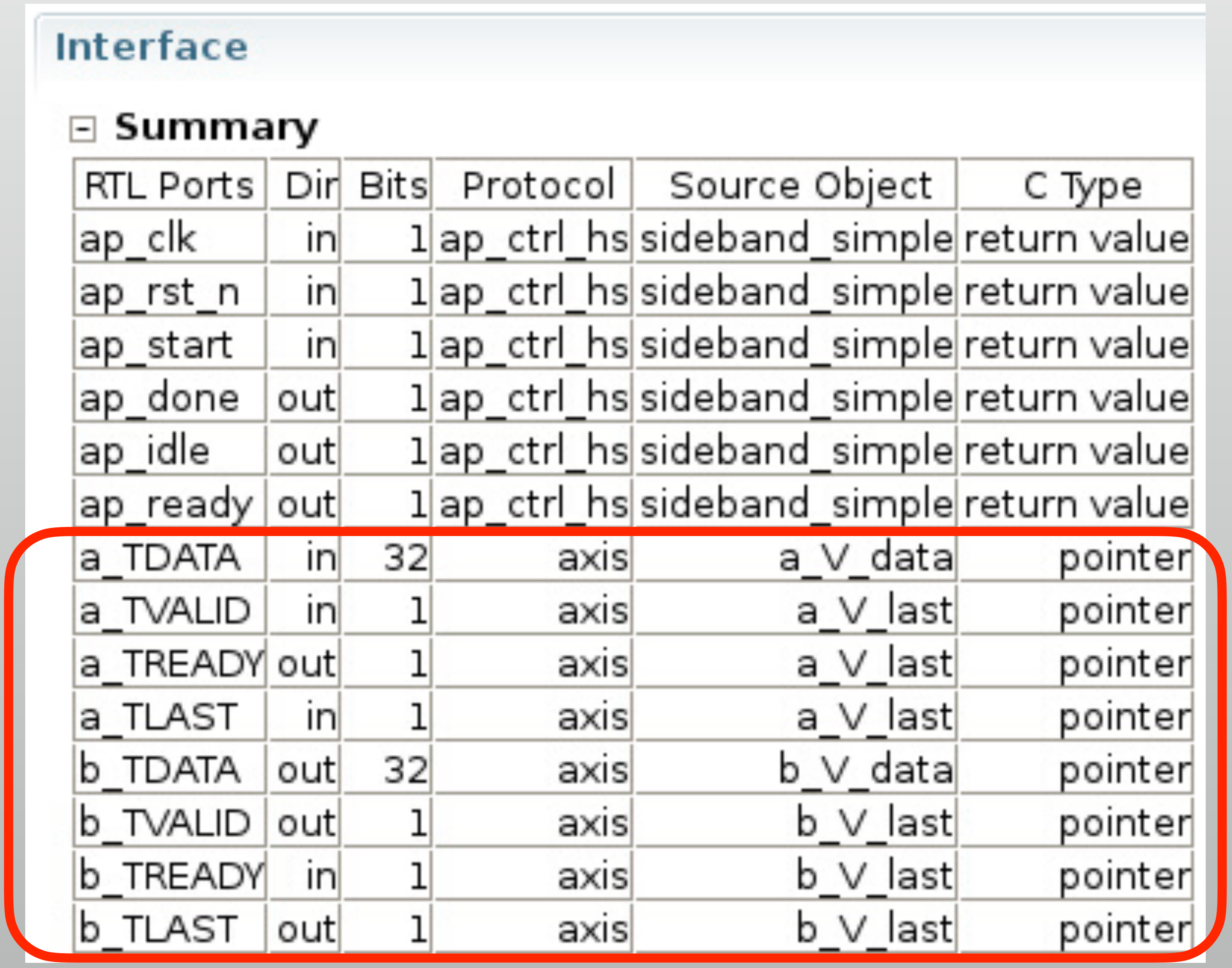
AXI Stream + sideband (simpler, non-official)

```
struct int_s{
  int data;
  bool last;
};

void sideband_simple
  (hls::stream<int_s>& a,
   hls::stream<int_s>& b){
#pragma HLS INTERFACE axis port=a
#pragma HLS INTERFACE axis port=b

  int_s aa, bb;
  int sum=0, i=0, ma, mi;
  do {
    aa = a.read();
    int x = aa.data;
    if (i==0){ ma=x; mi=x; }
    if (i!=0 && ma<x) ma=x;
    if (i!=0 && mi>x) mi=x;
    sum += x;
    i ++;
  } while(aa.last == 0);

  bb.data=sum; bb.last=0;  b.write(bb);
  bb.data=ma;  bb.last=0;  b.write(bb);
  bb.data=mi;  bb.last=1;  b.write(bb);
}
```

Custom struct with a member named `last`, instead of using `ap_axis`

Synthesized ports

- Simple AXI Stream port with TLAST only!

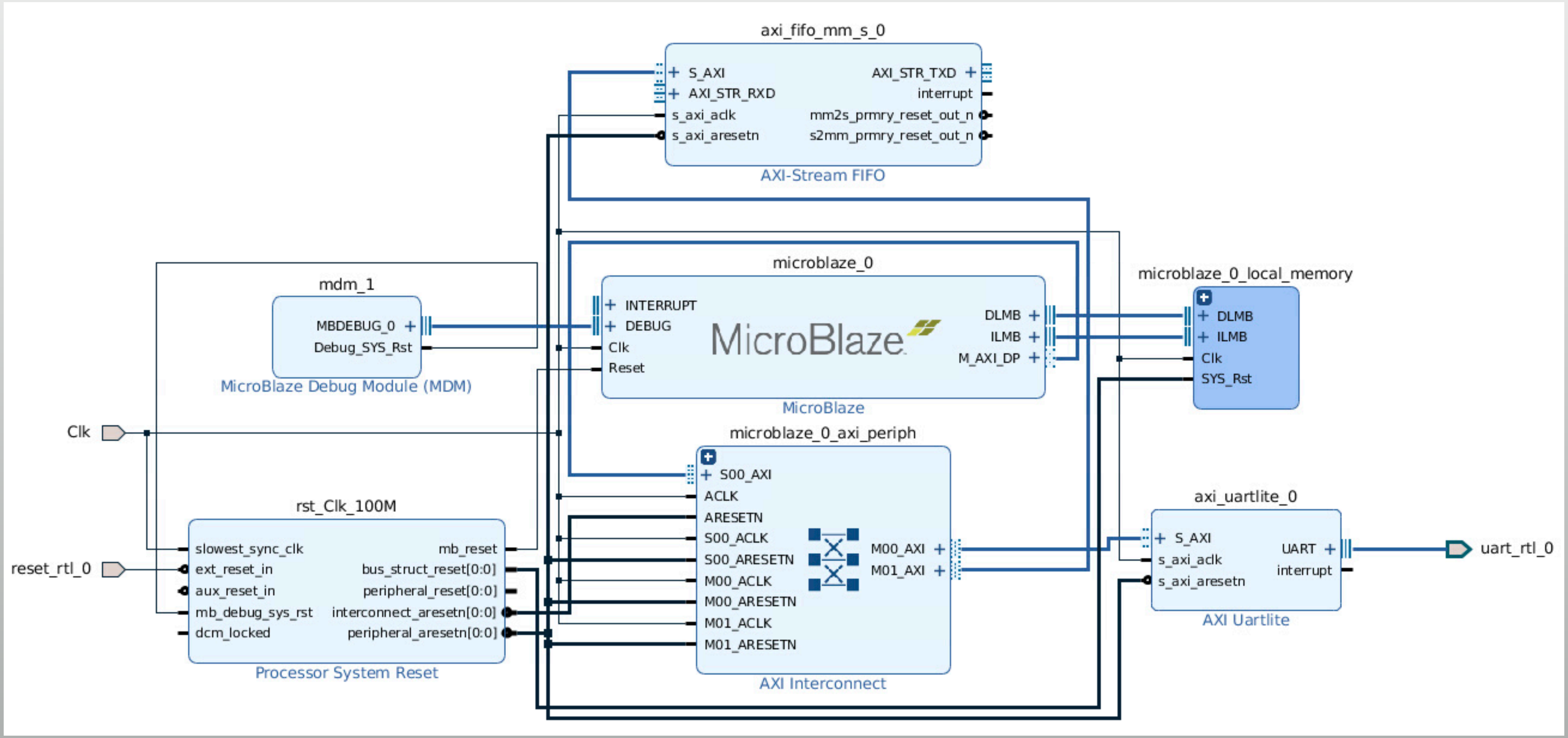


Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	sideband_simple	return value
ap_rst_n	in	1	ap_ctrl_hs	sideband_simple	return value
ap_start	in	1	ap_ctrl_hs	sideband_simple	return value
ap_done	out	1	ap_ctrl_hs	sideband_simple	return value
ap_idle	out	1	ap_ctrl_hs	sideband_simple	return value
ap_ready	out	1	ap_ctrl_hs	sideband_simple	return value
a_TDATA	in	32	axis	a_V_data	pointer
a_TVALID	in	1	axis	a_V_last	pointer
a_TREADY	out	1	axis	a_V_last	pointer
a_TLAST	in	1	axis	a_V_last	pointer
b_TDATA	out	32	axis	b_V_data	pointer
b_TVALID	out	1	axis	b_V_last	pointer
b_TREADY	in	1	axis	b_V_last	pointer
b_TLAST	out	1	axis	b_V_last	pointer

Connect to MicroBlaze

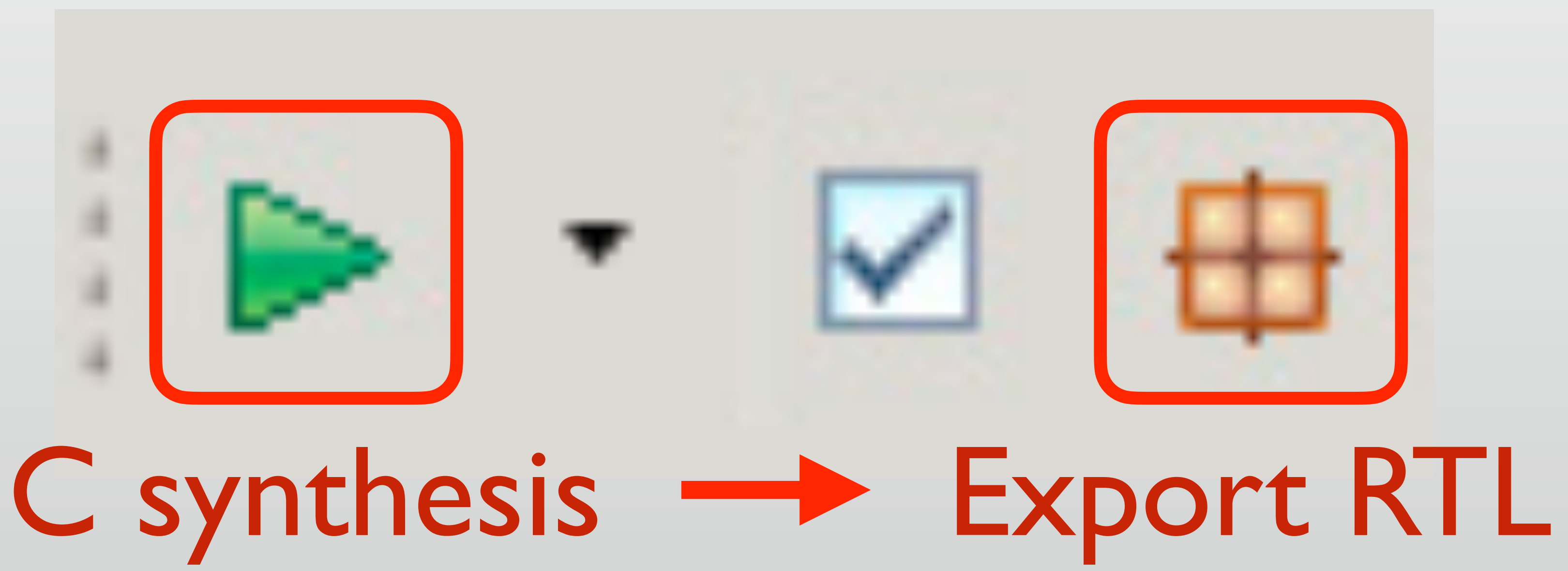


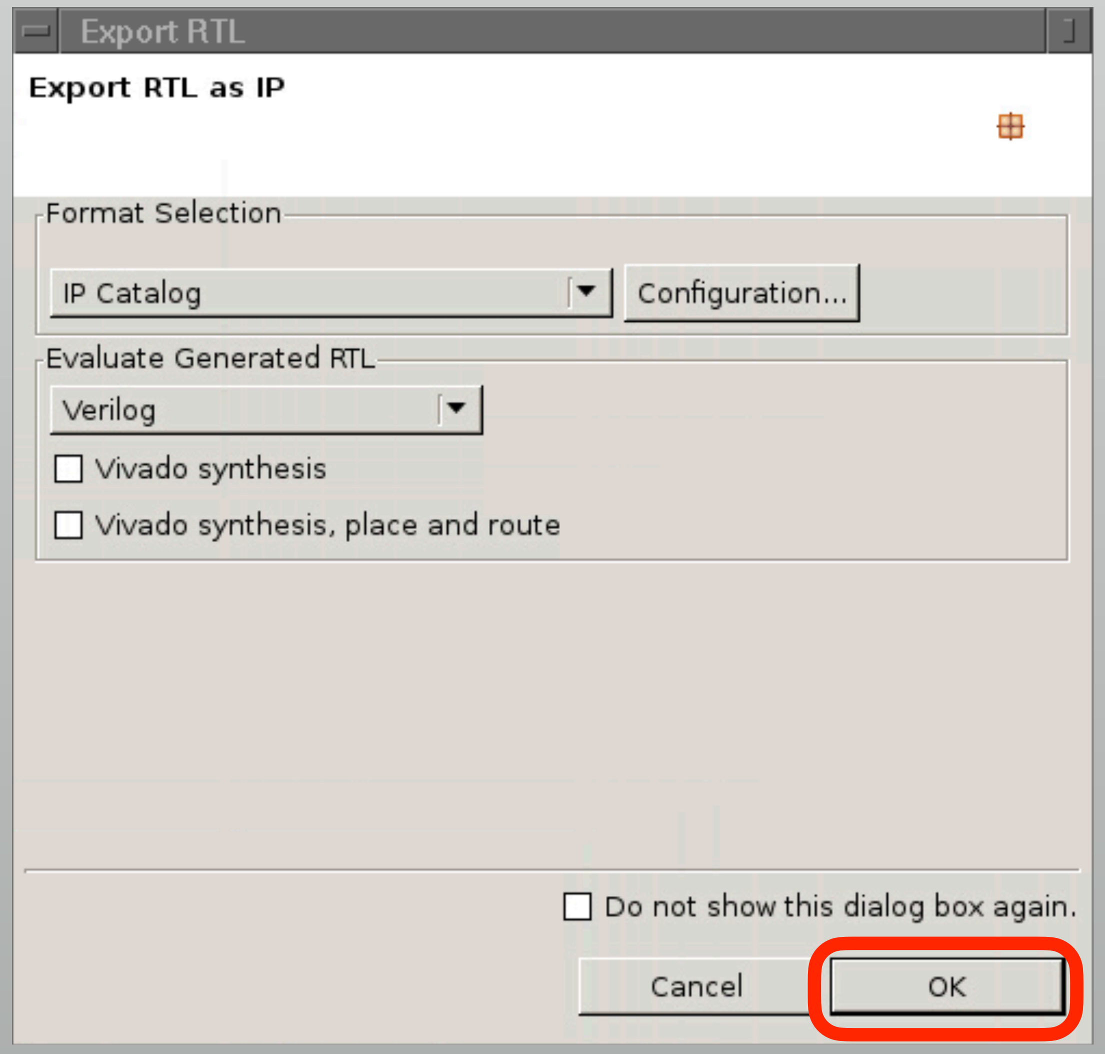
Create the base system in Vivado

- Create block design
  - Add MicroBlaze
    - Block automation (64kB RAM, external clock port)
    - Connection automation (Active high reset)
- Add UartLite
  - Make UART port external
- Add AXI Stream FIFO
  - Disable TX control port
- Connection Automation: all

Do HLS and export RTL

- Load axis-sc.cc in Vivado HLS
  - Choose sideband_simple as the top level module
  - Run C synthesis and Export RTL




Export RTL

Export RTL as IP

Format Selection

IP Catalog ▾ Configuration...

Evaluate Generated RTL

Verilog ▾

☐ Vivado synthesis

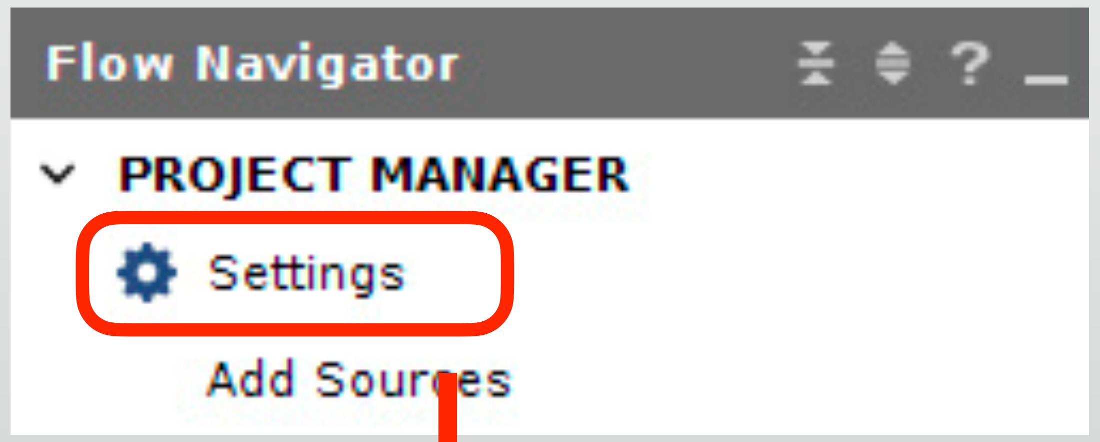
☐ Vivado synthesis, place and route

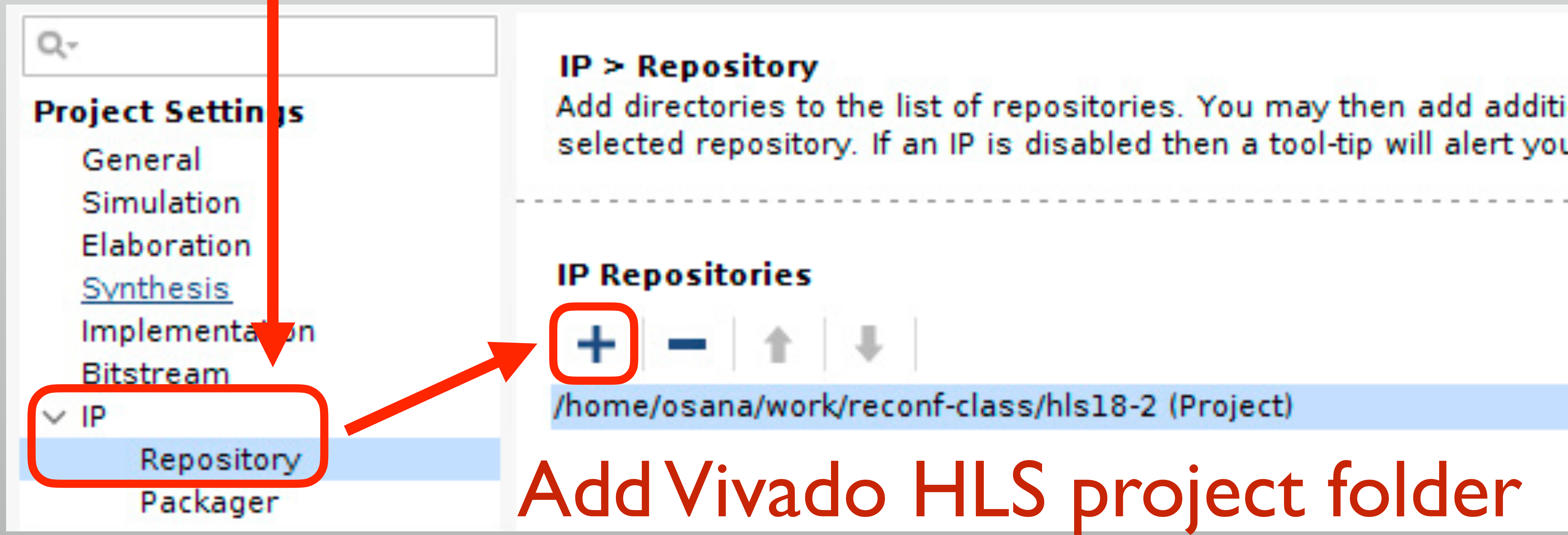
☐ Do not show this dialog box again.

Cancel OK

Find and Generate IP in Vivado

- Add IP repository
  - Vivado HLS project directory (or any upper directory)
  - HLS core will appear in IP catalog





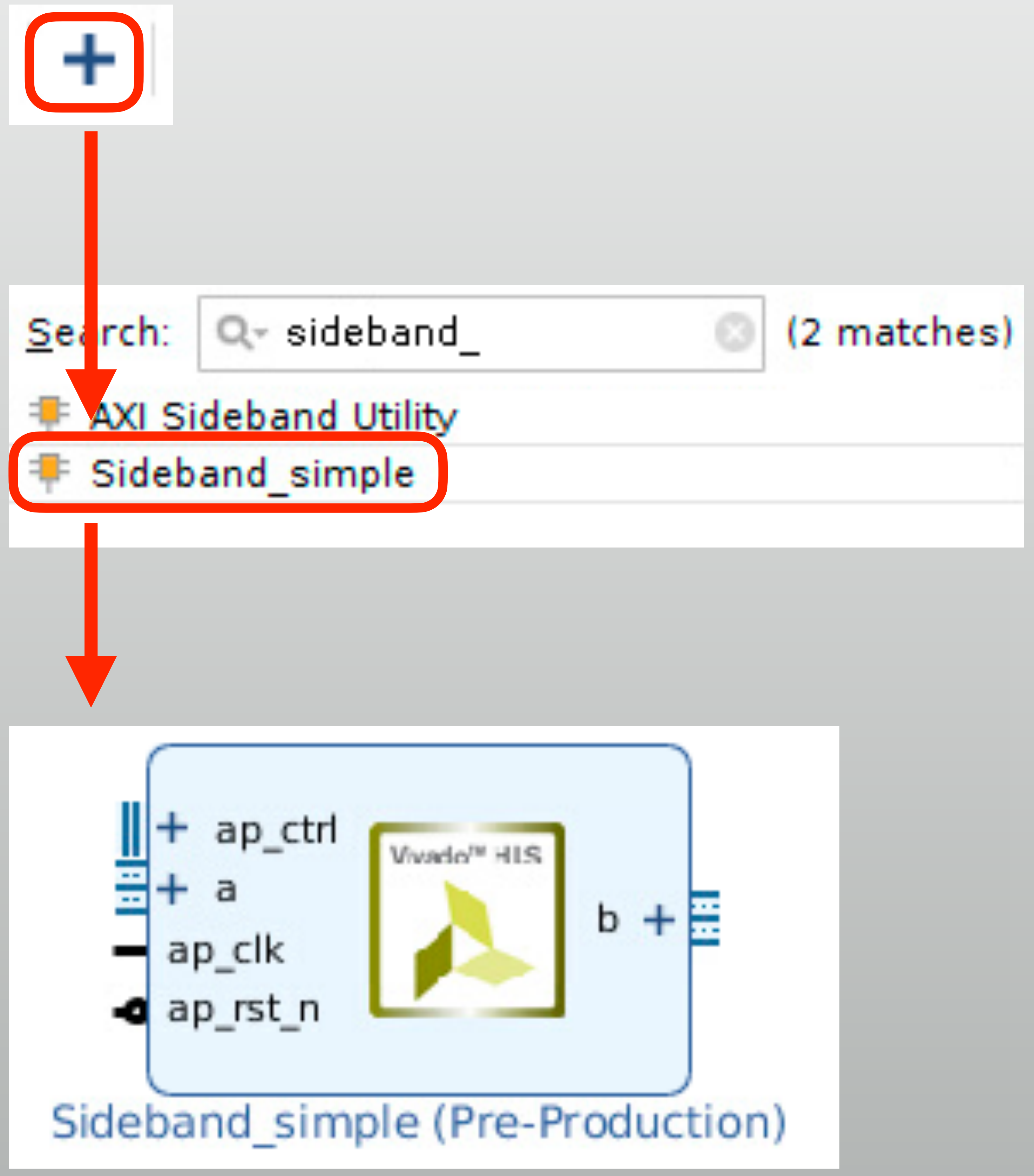
Add directories to the list of repositories. You may then add additi
selected repository. If an IP is disabled then a tool-tip will alert you

/home/osana/work/reconf-class/hls18-2 (Project)

Add Vivado HLS project folder

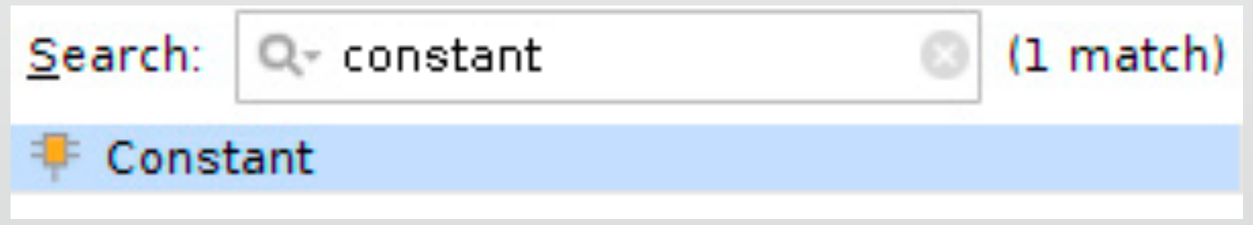
Add sideband_simple in block design

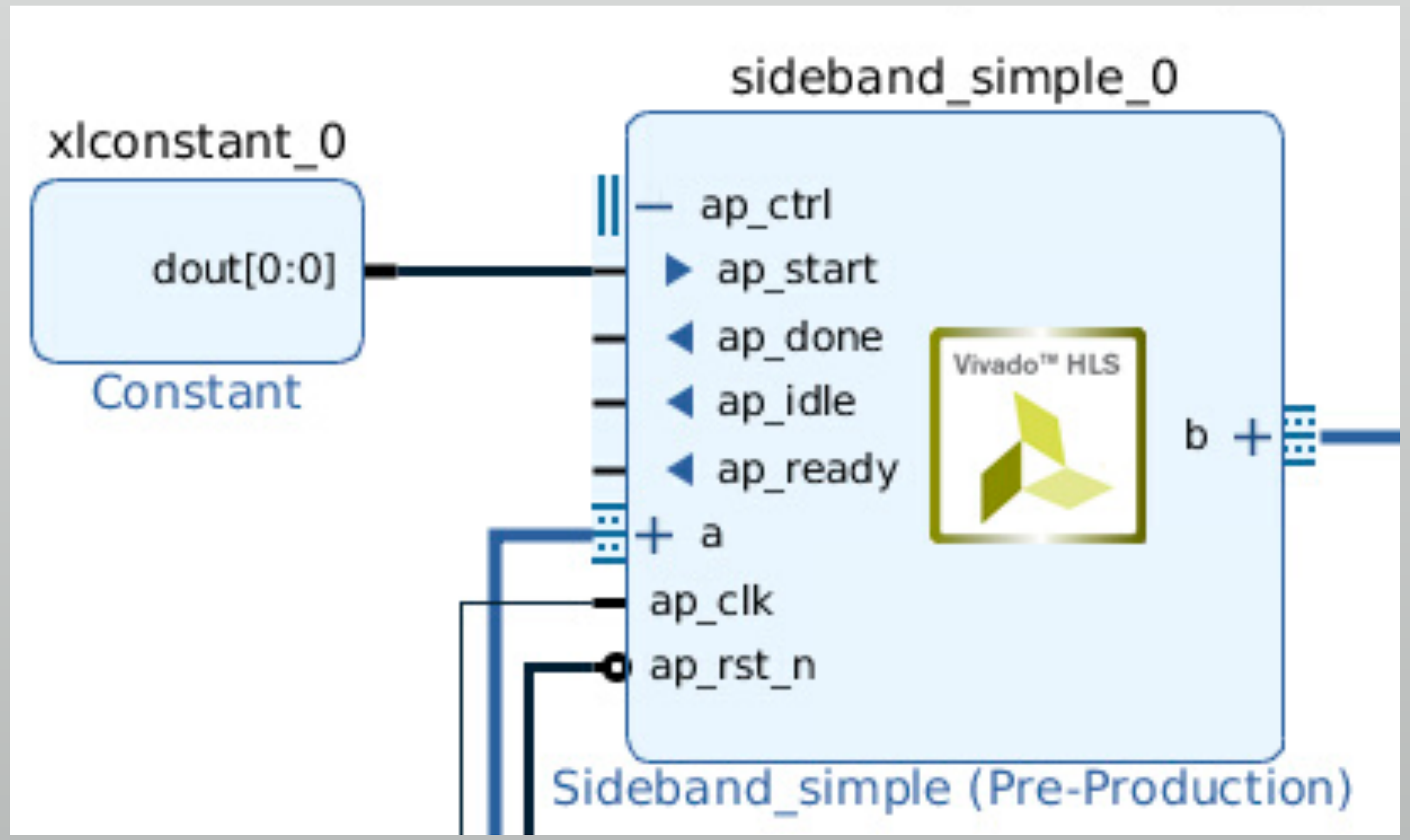
* HLS core to AXI-Stream FIFO:
  * ap_clk: to s_axi_aclk
  * ap_rst_n: to s_axi_areset
  * a (input): to AXI_STR_TXD
  * b (output): to AXI_STR_RXD



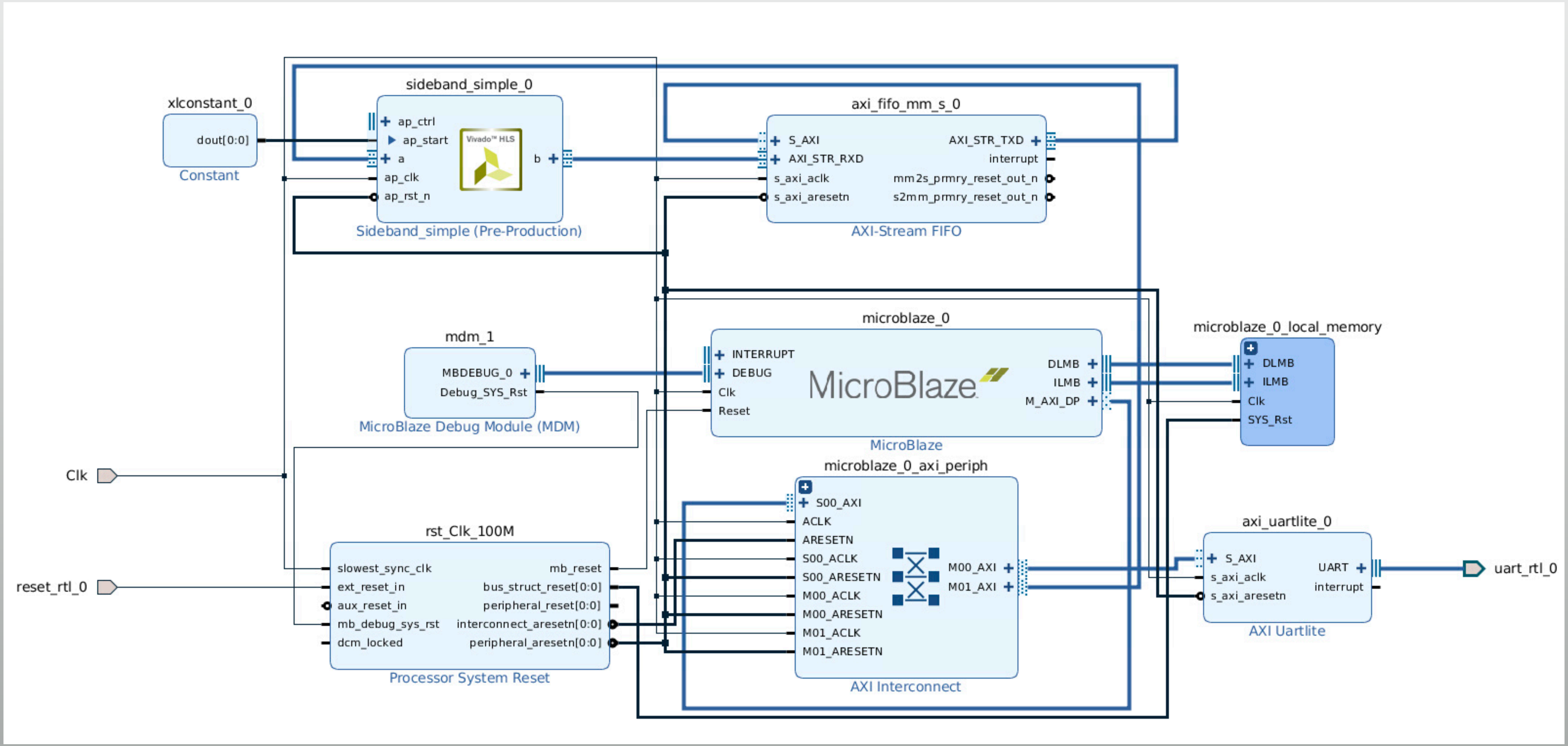
Keep HLS core running

- Add "constant" block to tie ap_start
  - default value is 1
  - connect to ap_start
- That's all for HW design
  - create HDL wrapper and load top.xdc as constraint



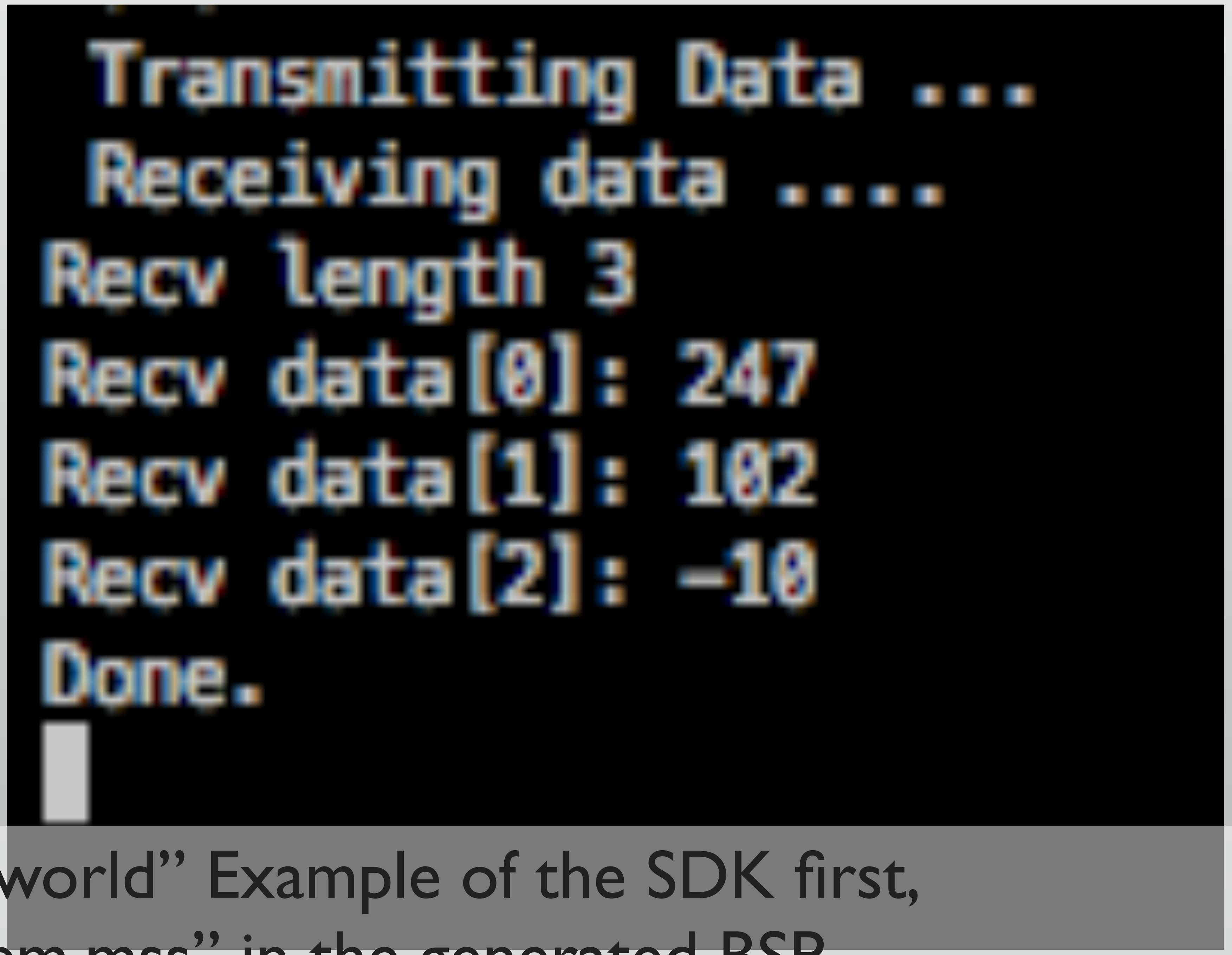


MicroBlaze + HLS core is ready



Implement and launch SDK

- Export Hardware with bitstream
  - then, launch SDK
- Create an application project, replace source code with mblaze.c
  - Very similar to “fifo-example.c” in 8th class
  - and it should work!!



```
 Transmitting Data ...
 Receiving data ....
Recv length 3
Recv data[0]: 247
Recv data[1]: 102
Recv data[2]: -10
Done.
```

1. Create “Hello world” Example of the SDK first,
2. then find “system.mss” in the generated BSP.
3. You can import the axi_fifo_mm_s polling example from “Peripheral drivers” in system.mss.
4. Just remove the C source in the generated polling example and replace by fifo-example.c .

The final assignment (Previously announced)

* Sample code to drive peripheral devices on Nexys4 is on the class web
  * Microphone, Accelerometer, Temperature sensor
  * Slide switches to control
  * See top.v for instructions, free to use in the assignment