ELD Lab 6
Exploring AXI Interface

Objective

- Understand the basics of AXI Interface
- Design floating point arithmetic using logarithmic and square root IPs available in Vivado

$$y = \frac{1}{\ln(x)}$$

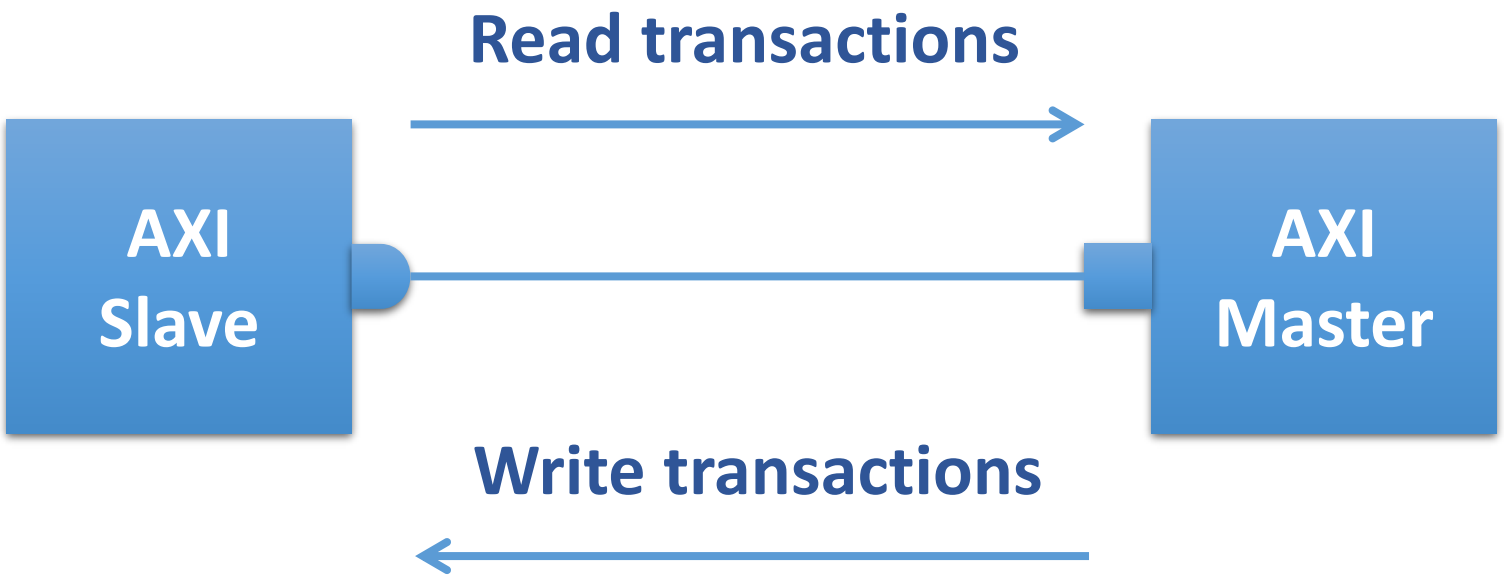
- **Lab Homework:** Design the floating point arithmetic circuit to implement following equation

$$z = \sqrt{x} + \frac{1}{\ln y} + 1.5$$

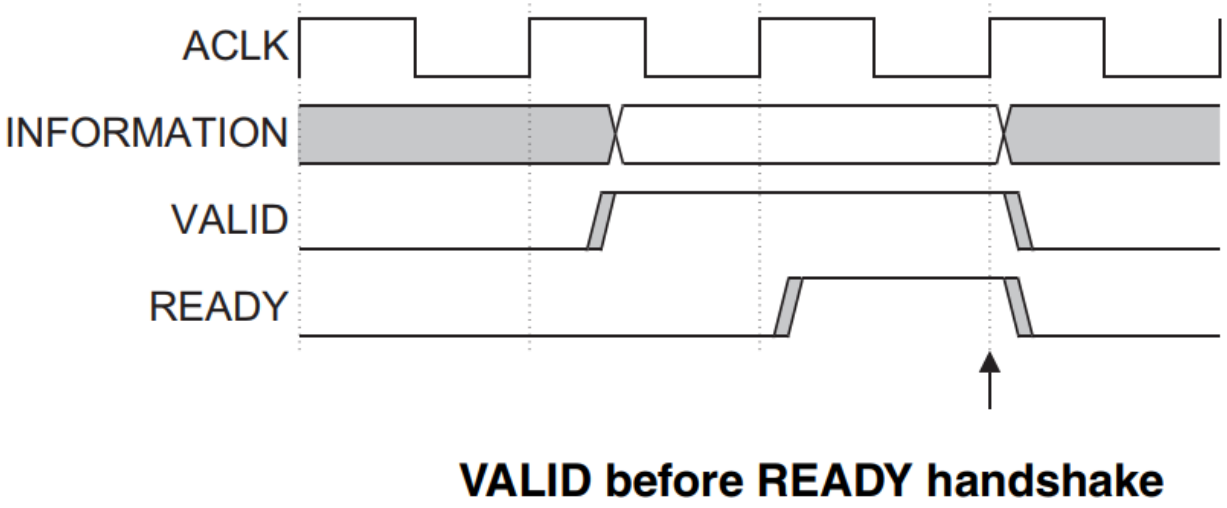
Theory

AXI

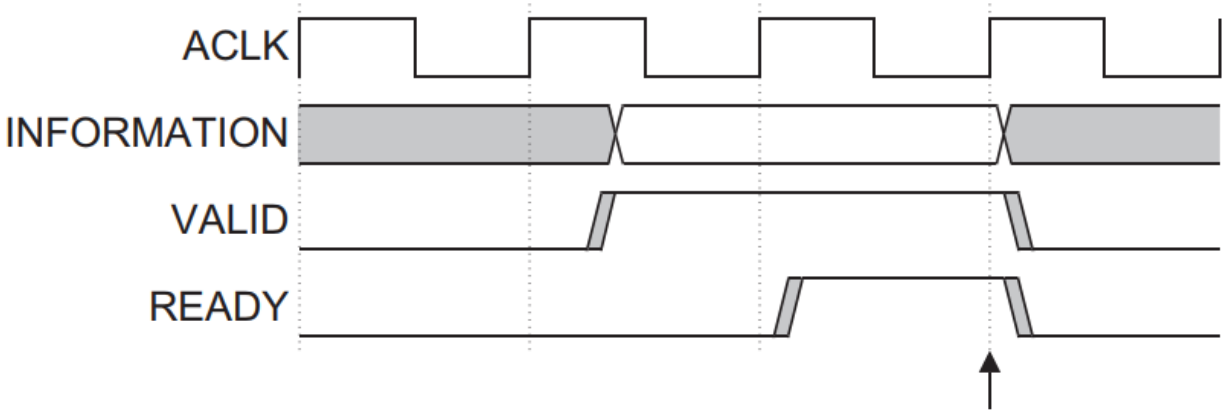
- Every AXI link contains two part: AXI master and AXI slave.
- AXI master initializes the transactions such as read and write. AXI slave is the one who responds to AXI master transactions.
- **Transaction:** Transfer of data from one point to another point



AXI Memory Mapped



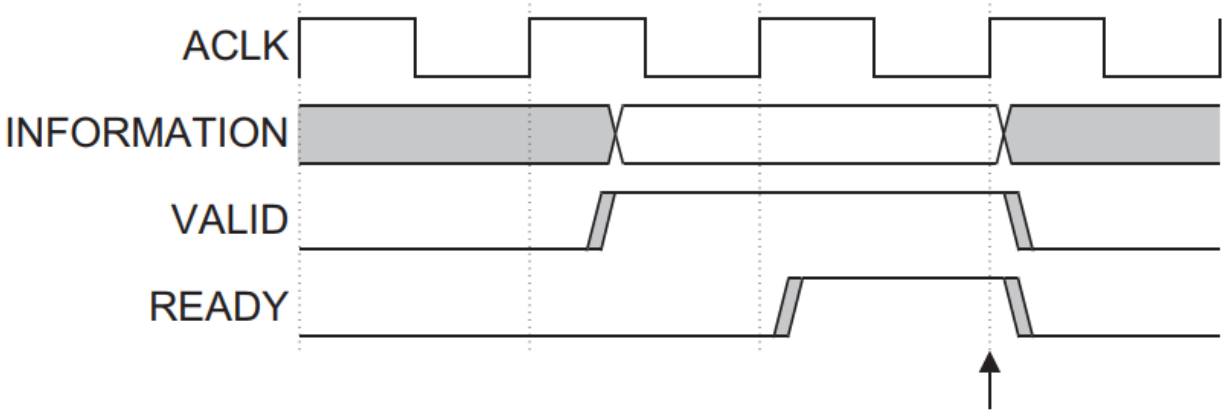
AXI Memory Mapped



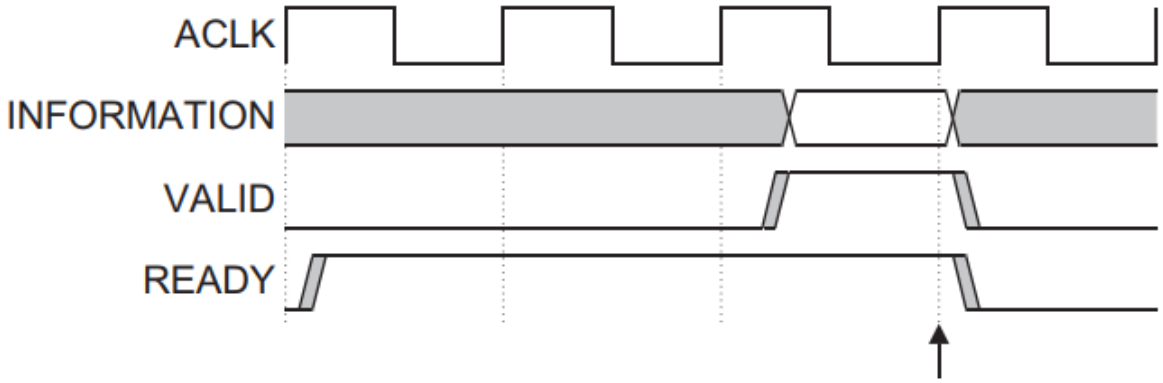
VALID before READY handshake

- The *source* generates the **VALID** signal to indicate when the address, data or control information is available.
- The *destination* generates the **READY** signal to indicate that it can accept the information.
- Transfer occurs only when *both* the **VALID** and **READY** signals are HIGH

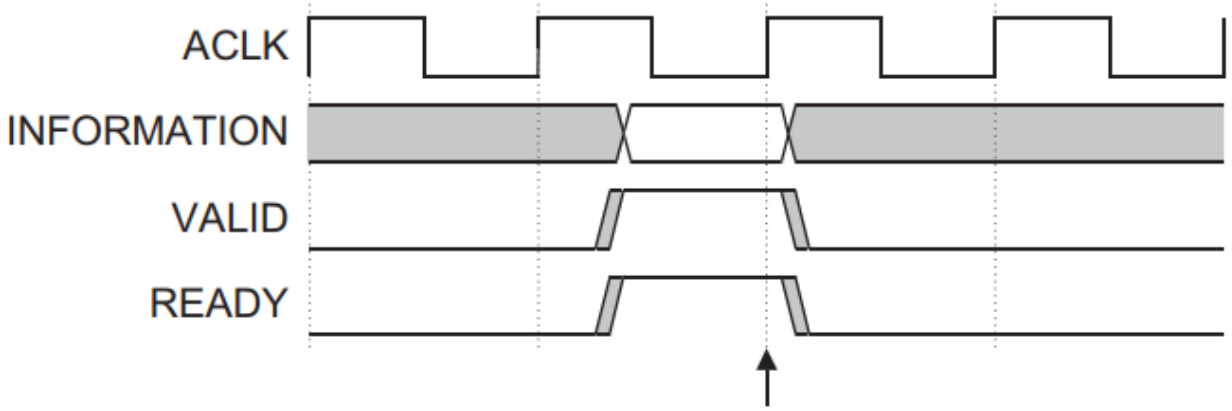
AXI Memory Mapped



VALID before READY handshake

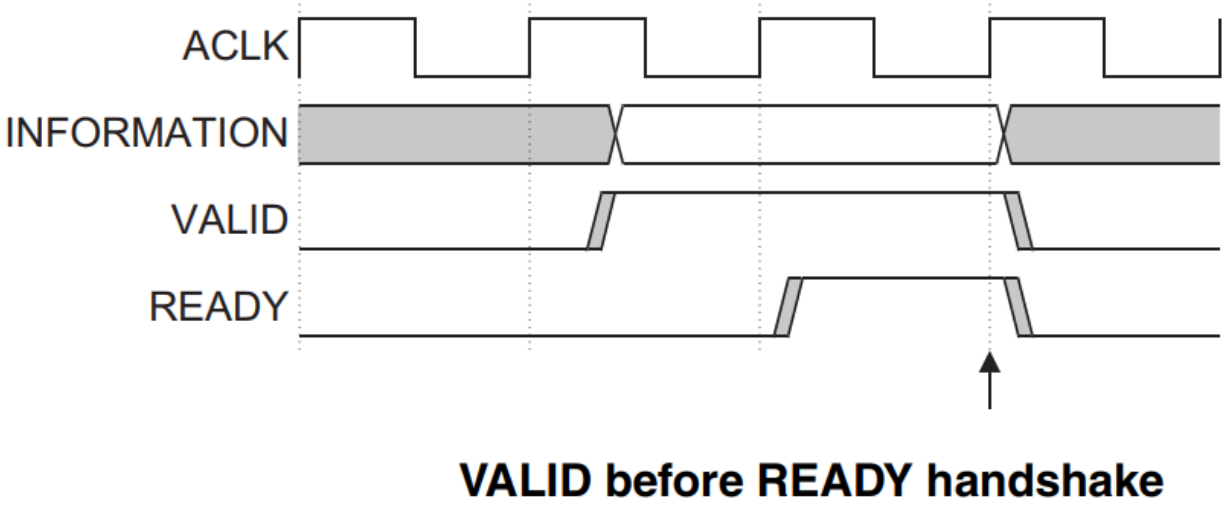


READY before VALID handshake



VALID with READY handshake

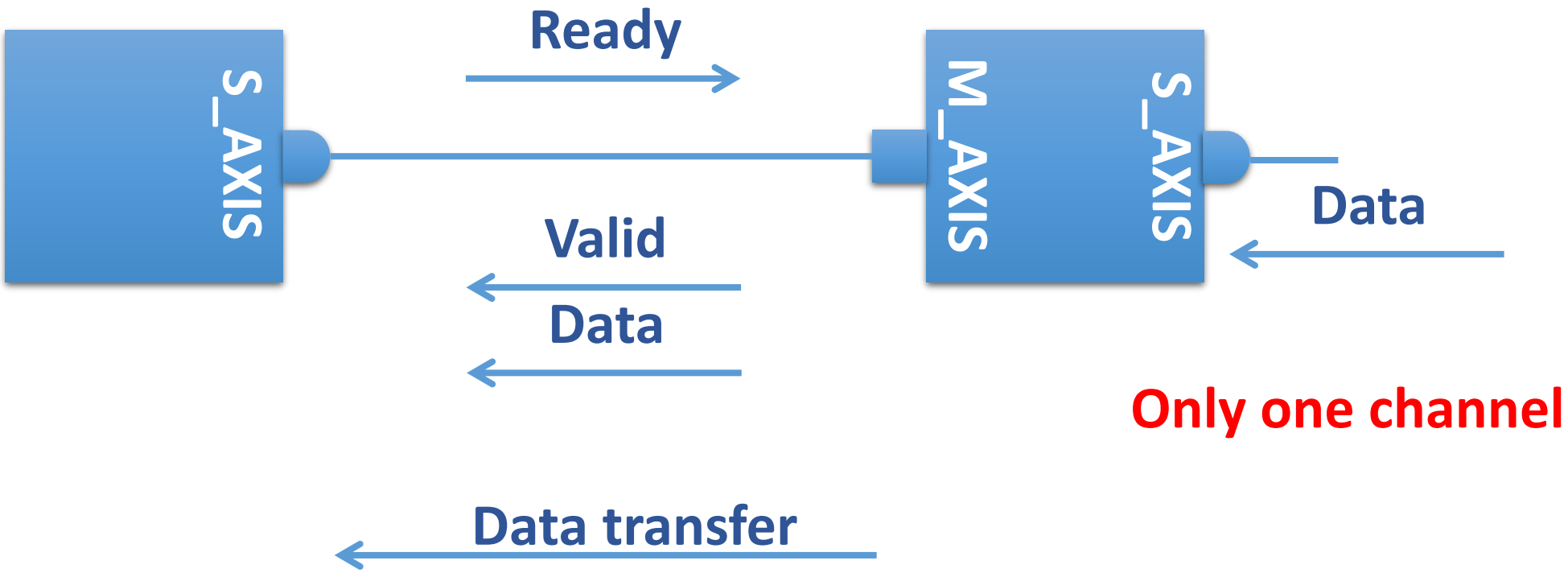
AXI Memory Mapped



- A source is NOT permitted to wait until READY is asserted before asserting VALID
- Once VALID is asserted it must remain asserted until the handshake occurs, at a rising clock edge at which VALID and READY are both asserted.
- A destination is permitted to wait for **VALID** to be asserted before asserting the corresponding **READY**.
- If **READY** is asserted, it is permitted to deassert **READY** before **VALID** is asserted.

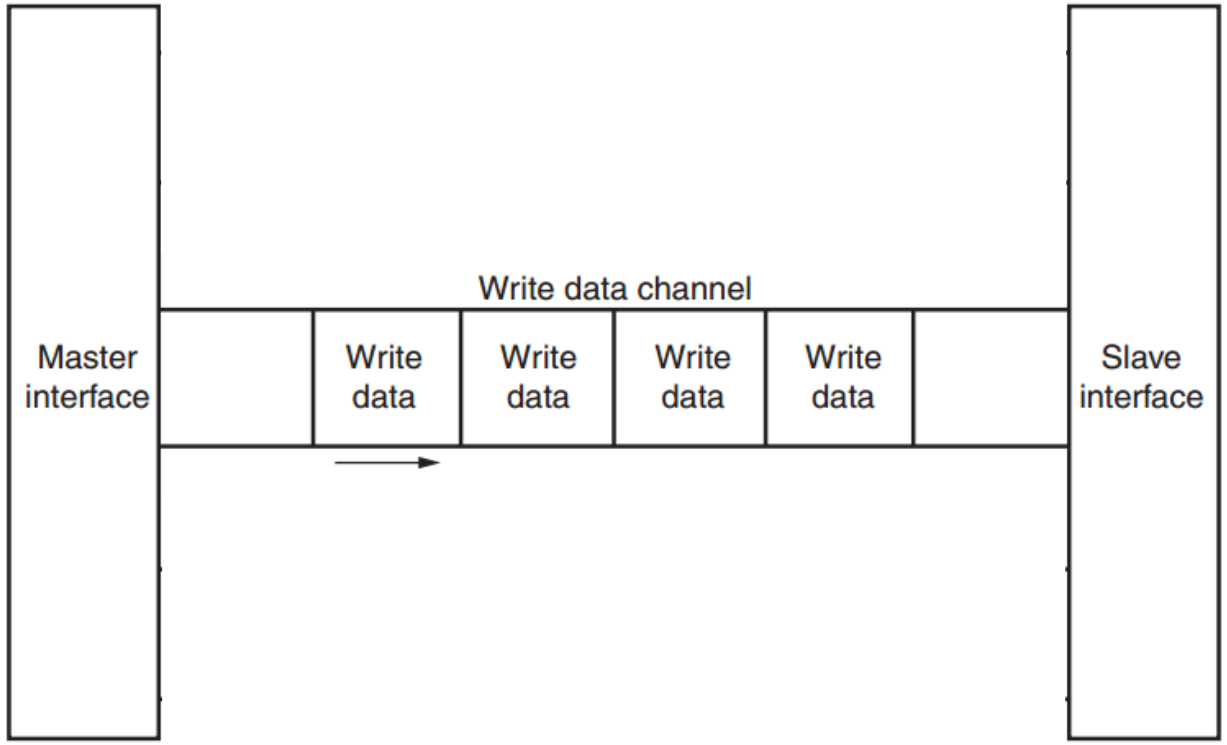
AXI Stream

- The AXI4-Stream protocol defines a single channel for transmission of streaming data (unlimited burst).



Only one channel

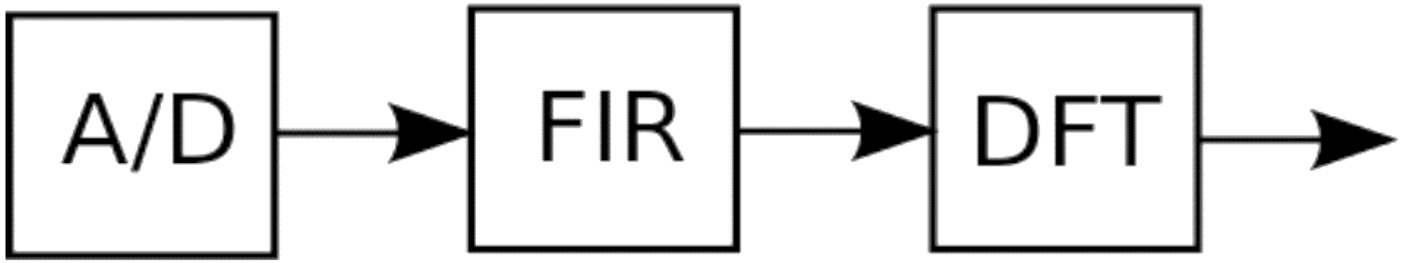
AXI Stream



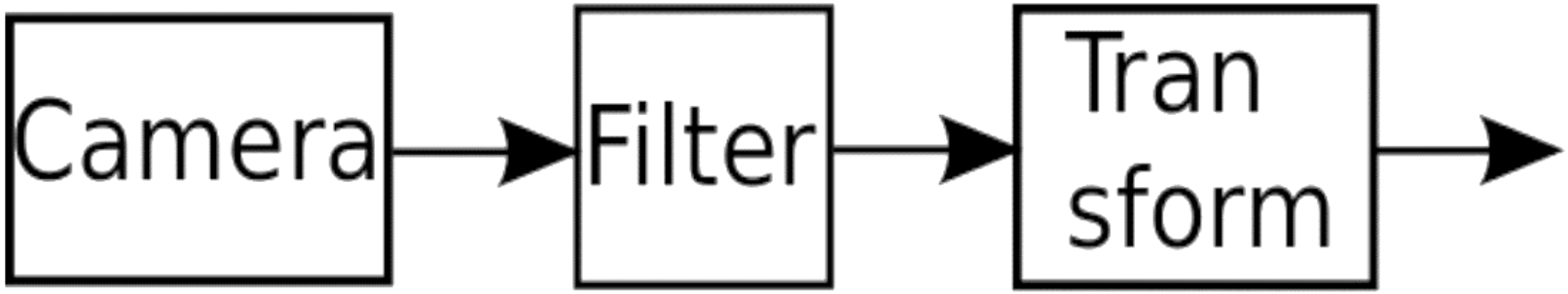
- The AXI4-Stream channel is modeled after the Write Data channel of the AXI4.
- Unlike AXI4, AXI4-Stream interfaces can burst an **unlimited** amount of data.

AXI Stream

Signal Processing

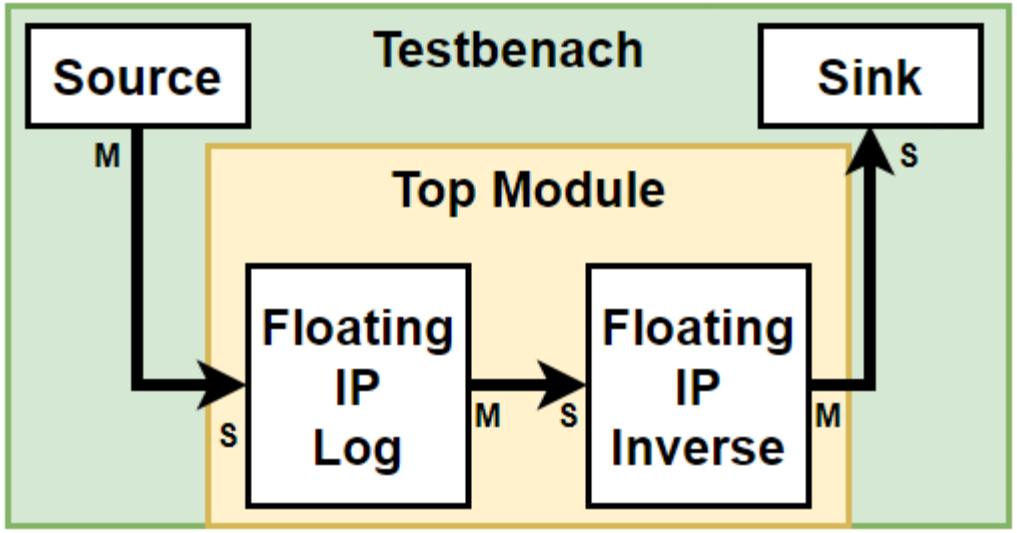


Video Processing

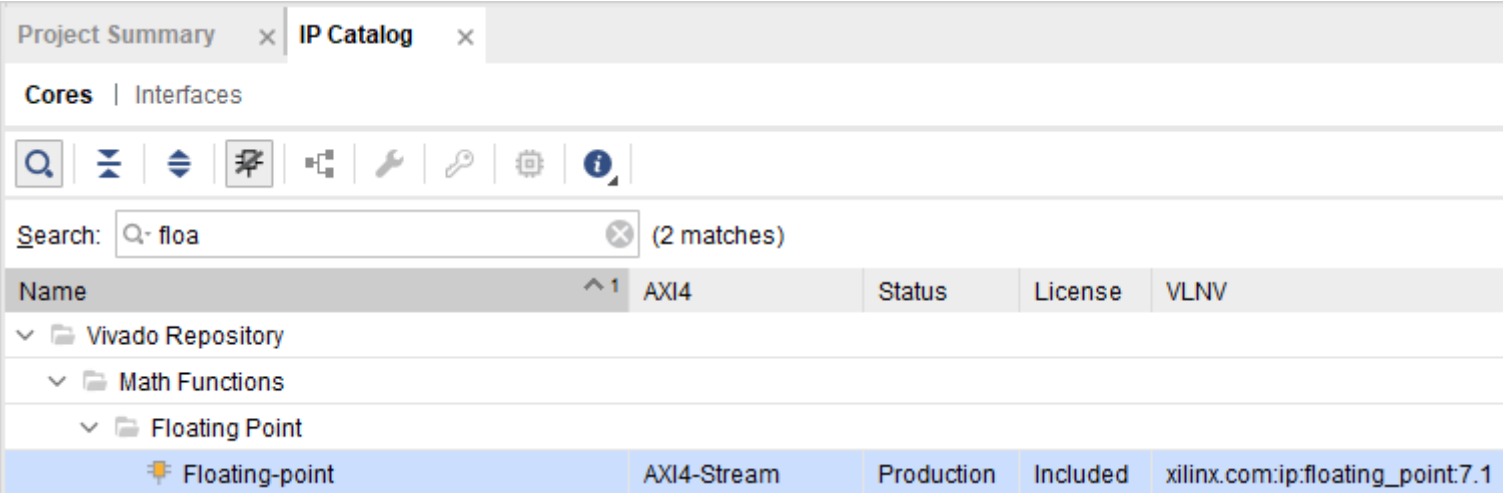


Lab

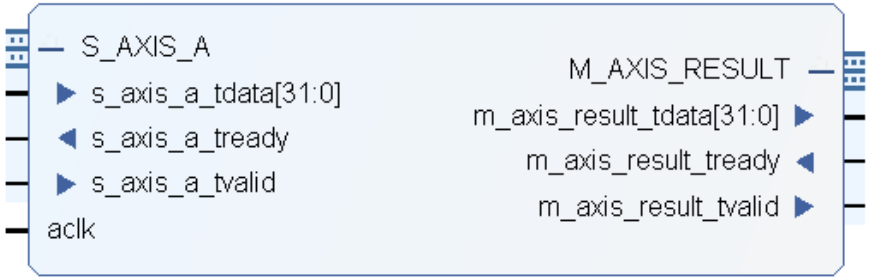
Proposed Approach



Locate the IP in Vivado



Log Operation



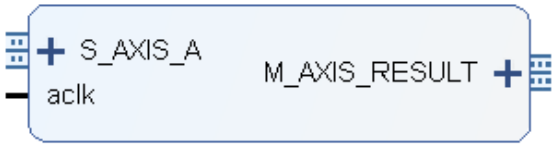
Customize IP

Floating-point (7.1)

Documentation | IP Location | Switch to Defaults

IP Symbol | Implementation Details

Show disabled ports



Component Name FP_log

Operation Selection | Precision of Inputs | Optimizations | Interface Optic

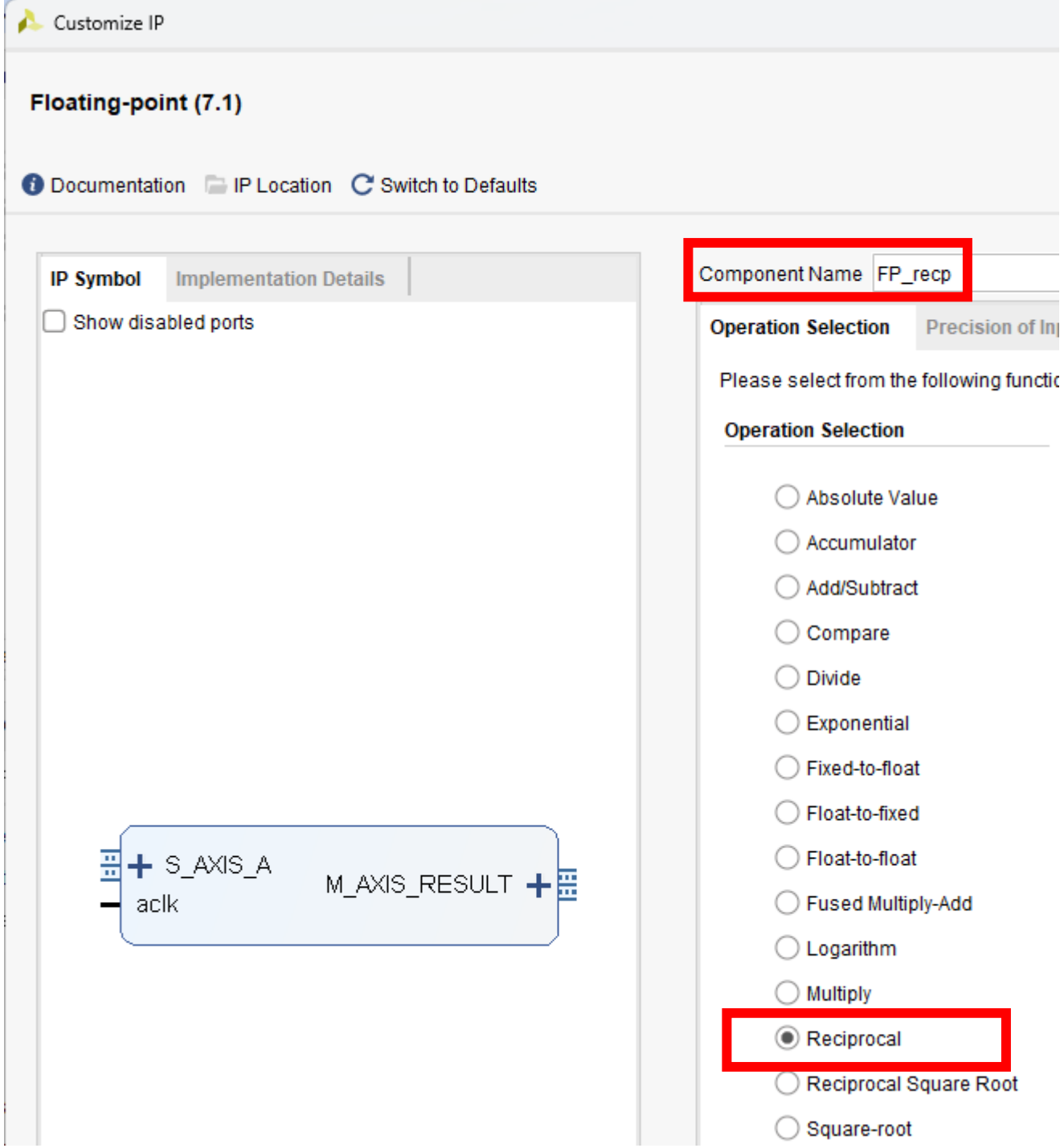
Please select from the following functions:

Operation Selection

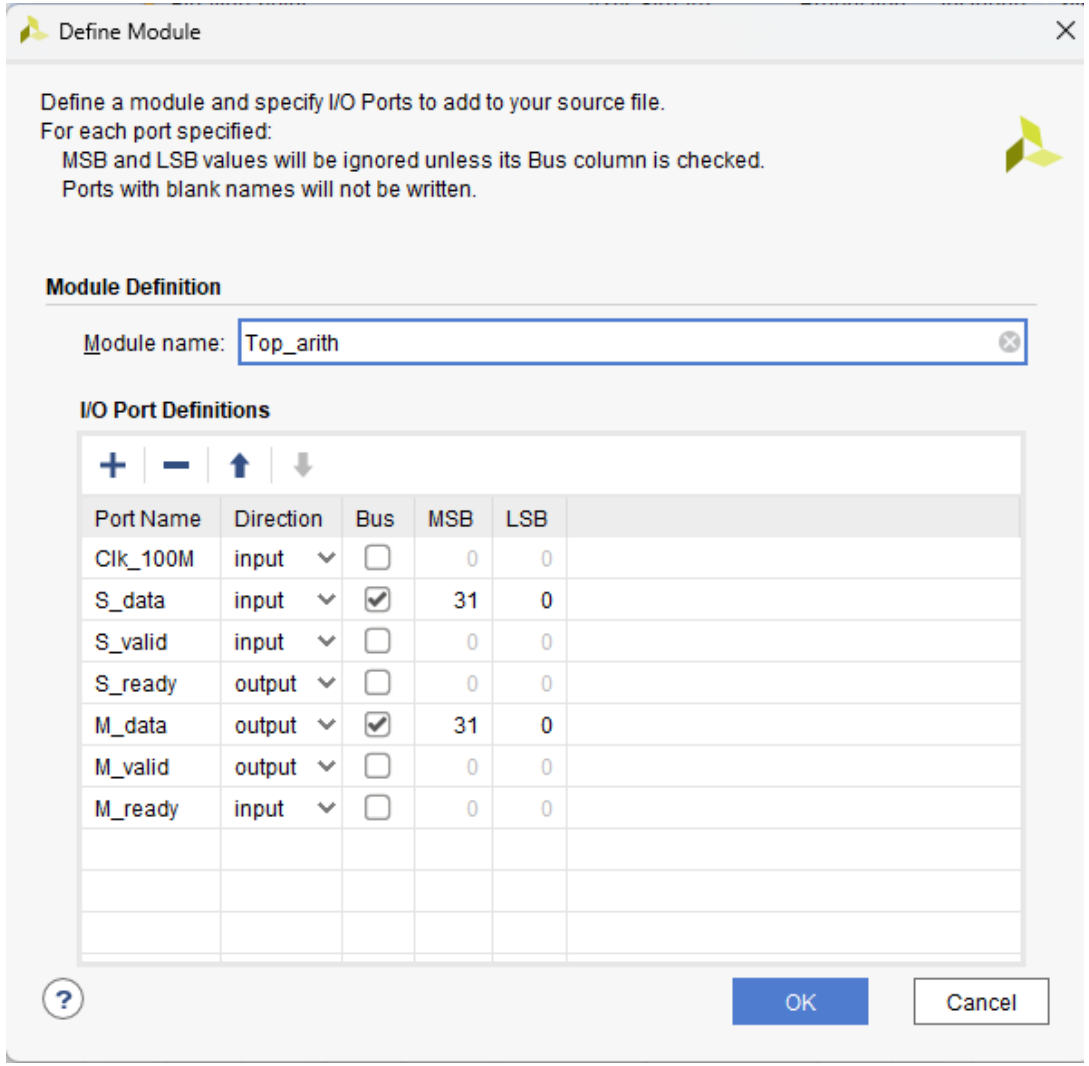
- Absolute Value
- Accumulator
- Add/Subtract
- Compare
- Divide
- Exponential
- Fixed-to-float
- Float-to-fixed
- Float-to-float
- Fused Multiply-Add
- Logarithm
- Multiply
- Reciprocal
- Reciprocal Square Root
- Square-root

Logarithm operation selected. **RESULT = ln(A)**

Reciprocal operation



Top Module



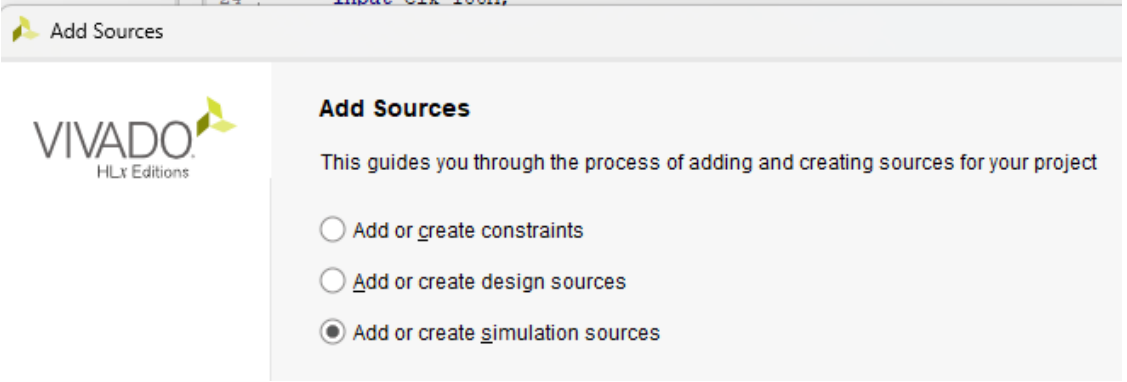

```
module Top_arith(
    input Clk_100M,
    input [31:0] S_data,
    input S_valid,
    output S_ready,
    output [31:0] M_data,
    output M_valid,
    input M_ready
    );

wire int_valid, int_ready;
wire [31:0] int_data;
FP_log l1 (
  .aclk(Clk_100M),                                  // input wire aclk
  .s_axis_a_tvalid(S_valid),              // input wire s_axis_a_tvalid
  .s_axis_a_tready(S_ready),              // output wire s_axis_a_tready
  .s_axis_a_tdata(S_data),                // input wire [31 : 0] s_axis_a_tdata
  .m_axis_result_tvalid(int_valid),  // output wire m_axis_result_tvalid
  .m_axis_result_tready(int_ready),  // input wire m_axis_result_tready
  .m_axis_result_tdata(int_data)    // output wire [31 : 0] m_axis_result_tdata
);

FP_recp r1 (
  .aclk(Clk_100M),                                  // input wire aclk
  .s_axis_a_tvalid(int_valid),              // input wire s_axis_a_tvalid
  .s_axis_a_tready(int_ready),              // output wire s_axis_a_tready
  .s_axis_a_tdata(int_data),                // input wire [31 : 0] s_axis_a_tdata
  .m_axis_result_tvalid(M_valid),  // output wire m_axis_result_tvalid
  .m_axis_result_tready(M_ready),  // input wire m_axis_result_tready
  .m_axis_result_tdata(M_data)    // output wire [31 : 0] m_axis_result_tdata
);

endmodule
```

Testbench



- Floating point converter: https://www.h-schmidt.net/FloatConverter/IEEE754.html

```
module arith_test(

    );

    reg Clk_100M, S_valid, M_ready;
    reg [31:0] S_data;
    wire M_valid, S_ready;
    wire [31:0] M_data;
    Top_arith t1 (.Clk_100M(Clk_100M),.S_data(S_data),.S_valid(S_valid),.S_ready(S_ready),.M_data(M_data),.M_valid(M_valid),.M_ready(M_ready));

    initial begin
        Clk_100M = 0;
        S_valid = 0;
        S_data = 0;
        M_ready = 1;
    end

    always
        #5 Clk_100M = ~Clk_100M;

    initial begin
        S_data = 32'b01000000101000000000000000000000;
        S_valid = 1;
        while (S_ready == 0)
            S_valid = 1;
        #5 S_valid = 0;
        @(posedge M_valid);
        #10 $stop;
    end
```

Simulations