MBImager Peripherals User Guide

Using the microblaze soft processor to control hdl modules or receive signals from those modules requires the implementation of certain peripherals in the design. This guide will show how to use these peripherals.

Note that there are 2 versions of many of the peripherals. Designs which will be implemented in hardware should use the versions ending in 00.a, while designs which will be simulated should use versions ending in 01.s. For peripherals with only a 00.a version, the 00.a version can be used for simulation as well. The 2 peripheral types function the same from a designer perspective, but the 01.s versions use simulator compatible .v files instead of the netlists used in some 00.a peripherals.

Table of Contents

HDL Control

The HDL Control peripheral allows up to 1024 control signals in the HDL to be directed by the microblaze.

On the HDL side, the peripheral presents 32 32-bit output ports, named OUT0 through OUT31. Any combination of these ports can be made external and connected to arbitrary HDL module inputs.

On the processor side, 32 32-bit memory-mapped registers are provided. The value of each output port exactly matches the content of the corresponding register, as soon as the data can pass over the AXI bus. The address of the register corresponding to OUT0 is the base address of the peripheral (defined in xparameters.h). The address of OUT1's is the (base address) + 0x4, for OUT2 it is (base address) + 0x8. This continues up until the address corresponding to OUT31 is (base address) + 0x7c. Writing to the registers is best done using the Xil_Out32 macro.

HDL Receive

The HDL Receive peripheral allows up to 1024 signals from the HDL to be read by the microblaze.

On the HDL side, the peripheral provides 32 32-bit input ports named IN0 through IN31. Any combination of these ports can be made external and connected to arbitrary HDL outputs, or combinations of outputs.

On the processor side, 32 32-bit registers are provided. The content of these registers matches the values provided to the input ports, with a delay corresponding to the AXI bus. The address of the register corresponding to IN0 is the base address of the peripheral (defined in xparameters.h). The address of IN1's is the (base address) + 0x4, for IN2 it is (base address) + 0x8. This continues up until the address corresponding to IN31 is (base address) + 0x7c. Reading from the registers is best done using the Xil_In32 macro.

okWireIns

The okWireIns peripheral is a specialized version of the HDL Receive peripheral which allows Opal Kelly Frontpanel WireIns to connect to registers accessible to the microblaze.

On the HDL side, this peripheral takes a single input: the okHE output from the okHost instance being used by Frontpanel.

On the processor side, this peripheral provides 32 32-bit registers, each of which corresponds to the dataout line of a single okWireIn. The address of the register corresponding to the WireIn with address 0x00 is the base address of the peripheral (defined in xparameters.h). The address corresponding to the WireIn with address 0x01 is (base address) + 0x04, for the WireIn with address 0x02 it's (base address) + 0x08. This continues until the memory address corresponding to endpoint address 0x1f is (base address) + 0x7c. Reading from the registers is best done using the Xil_In32 macro.

okWireOuts

The okWireOuts peripheral is a specialized version of the HDL Control peripheral which allows Opal Kelly Frontpanel WireOuts to be connected to registers accessible to the microblaze.

On the HDL side, this peripheral takes two inputs: the okHE output from the okHost instance being used by Frontpanel, and an okEH line. The okEH line may need to be used as an input to an okWireOR if other Frontpanel outputs such as TriggerOuts are being used.

On the processor side, this peripheral provides 32 32-bit registers, each of which corresponds to the datain line of a single okWireOut. The address of the register corresponding to the WireOut with address 0x20 is the base address of the peripheral (defined in xparameters.h). The address corresponding to the WireOut with address 0x21 is (base address) + 0x04, for the WireOut with address 0x22 it's (base address) + 0x08. This continues until the memory address corresponding to endpoint address 0x3f is (base address) + 0x7c. Writing to the registers is best done using the Xil_Out32 macro.

okTriggerIns

The okTriggerIns peripheral interfaces okTriggerIns with the microblaze soft processor. It allows the processor to query whether or not any bit in a TriggerIn has been triggered and to clear the status of a trigger. The peripheral is also configurable to cause processor interrupts when specific triggers are activated. A driver is provided to facilitate these functions.

From the HDL side, this peripheral takes the okHE output from the okHost instance being used by Frontpanel as in input. It also provides an output designated Intr, which should be connected to the system's Interrupt Controller in the mhs if interrupts will be used in the design.

From the processor side, a driver provides the necessary functions to use the peripheral. The driver will be included in your project by libgen, to use it simply include "oktriggerins.h" in your C code.

Before the driver can be used, the function OKTRIGGERINS_Initialize must be called, with the base address of the peripheral as its parameter. The base address can be found in xparameters.h.

To read the state of a trigger, the function OKTRIGGERINS_GetTrigger takes a trigger (number 0x40-0x5f), a 32-bit mask, and a 32-bit buffer as its parameters, and stores the state of the trigger bitwise anded with the mask in the buffer. This allows specific bits to be checked more easily. A bit in the value returned is 0 normally, and 1 if a trigger has been sent on that bit for that TriggerIn. The number of the trigger corresponds to the triggers ep_address. So the TriggerIn with ep_address 0x41 is trigger number 0x41.

To clear the state of a trigger, setting all trigger bits to 0, call the function OKTRIGGERINS_ClearTrigger, with the trigger (number 0x40-0x5f) as its parameter.

To set the interrupt mask of a trigger, the function OKTRIGGERINS_SetInterruptMask takes a trigger (number 0x40-0x5f) and a 32-bit mask. This function both clears the state of the trigger, setting all trigger bits to 0, and also sets the interrupt mask of the trigger to the mask. The interrupt mask determines which trigger bits send processor interrupts when triggered. A 1 in the mask indicates that the trigger bit in the same position should send a processor interrupt when triggered. Default masks are 0.

To set an interrupt handler for a bit(s) in a trigger, the function OKTRIGGERINS_RegisterHandler registers a handler function handler to be called with a specified parameter data when an interrupt is sent by the given trigger (number 0x40-0x5f) on a bit included in the 32-bit mask. It is the responsibility of the handler function to clear the trigger, and failure to do this will prevent additional interrupts from processing. The default handler simply clears the trigger and does nothing else.

When setting up interrupts, the function OKTRIGGERINS_Handler must be registered with the system's interrupt controller (the CallBackRef can be null).

All driver api functions return the type XStatus. They return XST_SUCCESS if everything is fine, and XST_FAILURE if the trigger given is invalid and XST_DEVICE_NOT_FOUND if the driver has not been initialized.

okTriggerOuts

The okTriggerOuts peripheral allows the microblaze to send TriggerOut signals through Frontpanel.

On the HDL side, this peripheral takes two inputs: the okHE output from the okHost instance being used by Frontpanel, and an okEH line. The okEH line may need to be used as an input to an okWireOR if other Frontpanel outputs such as WireOuts are being used.

From the processor side, this peripheral provides 32 32-bit registers. After writing a value to these registers, the corresponding TriggerOut will be triggered with the written value as its trigger, then the value of the register will return to 0. The address of the register corresponding to the TriggerOut with address 0x60 is the base address of the peripheral (defined in xparameters.h). The address corresponding to the TriggerOut with address 0x61 is (base address) + 0x04, for the TriggerOut with address 0x62 it's (base address) + 0x08. This continues until the memory address corresponding to endpoint address 0x7f is (base address) + 0x7c. Writing to the registers is best done using the Xil_Out32 macro.

Example

Here is a simple example using all of these peripherals, and the results of simulating it.

System.mhs:

```
# ##############################################################################
# Created by Base System Builder Wizard for Xilinx EDK 14.7 Build EDK_P.20131013
# Mon Jun 05 11:56:39 2017
# Target Board:  Custom
# Family:    spartan6
# Device:    xc6slx45
# Package:   csg484
# Speed Grade:  -2
# ##############################################################################
 PARAMETER VERSION = 2.1.0


 PORT RESET = RESET, DIR = I, SIGIS = RST, RST_POLARITY = 1
 PORT CLK_P = CLK, DIR = I, DIFFERENTIAL_POLARITY = P, SIGIS = CLK, CLK_FREQ = 100000000
 PORT CLK_N = CLK, DIR = I, DIFFERENTIAL_POLARITY = N, SIGIS = CLK, CLK_FREQ = 100000000
 PORT okHe = net_okHe, DIR = I, VEC = [112:0]
 PORT okwireouts_0_okEH_pin = okwireouts_0_okEH, DIR = O, VEC = [64:0]
 PORT oktriggerouts_0_okEH_pin = oktriggerouts_0_okEH, DIR = O, VEC = [64:0]
 PORT oktriggerouts_0_okHE_pin = oktriggerouts_0_okHE, DIR = I, VEC = [112:0]
 PORT hdl_control_0_OUT0_pin = hdl_control_0_OUT0, DIR = O, VEC = [31:0]
 PORT hdl_receive_0_IN0_pin = hdl_receive_0_IN0, DIR = I, VEC = [31:0]


BEGIN proc_sys_reset
```

```
 PARAMETER INSTANCE = proc_sys_reset_0
 PARAMETER HW_VER = 3.00.a
 PARAMETER C_EXT_RESET_HIGH = 1
 PORT MB_Debug_Sys_Rst = proc_sys_reset_0_MB_Debug_Sys_Rst
 PORT Dcm_locked = proc_sys_reset_0_Dcm_locked
 PORT MB_Reset = proc_sys_reset_0_MB_Reset
 PORT Slowest_sync_clk = clk_100_0000MHz
 PORT Interconnect_aresetn = proc_sys_reset_0_Interconnect_aresetn
 PORT Ext_Reset_In = RESET
 PORT BUS_STRUCT_RESET = proc_sys_reset_0_BUS_STRUCT_RESET
END

BEGIN lmb_v10
 PARAMETER INSTANCE = microblaze_0_ilmb
 PARAMETER HW_VER = 2.00.b
 PORT SYS_RST = proc_sys_reset_0_BUS_STRUCT_RESET
 PORT LMB_CLK = clk_100_0000MHz
END

BEGIN lmb_bram_if_cntlr
 PARAMETER INSTANCE = microblaze_0_i_bram_ctrl
 PARAMETER HW_VER = 3.10.c
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x0000ffff
 BUS_INTERFACE SLMB = microblaze_0_ilmb
 BUS_INTERFACE BRAM_PORT = microblaze_0_i_bram_ctrl_2_microblaze_0_bram_block
END

BEGIN lmb_v10
```

```
 PARAMETER INSTANCE = microblaze_0_dlmb
 PARAMETER HW_VER = 2.00.b
 PORT SYS_RST = proc_sys_reset_0_BUS_STRUCT_RESET
 PORT LMB_CLK = clk_100_0000MHz
END

BEGIN lmb_bram_if_cntlr
 PARAMETER INSTANCE = microblaze_0_d_bram_ctrl
 PARAMETER HW_VER = 3.10.c
 PARAMETER C_BASEADDR = 0x00000000
 PARAMETER C_HIGHADDR = 0x0000ffff
 BUS_INTERFACE SLMB = microblaze_0_dlmb
 BUS_INTERFACE BRAM_PORT = microblaze_0_d_bram_ctrl_2_microblaze_0_bram_block
END

BEGIN bram_block
 PARAMETER INSTANCE = microblaze_0_bram_block
 PARAMETER HW_VER = 1.00.a
 BUS_INTERFACE PORTA = microblaze_0_i_bram_ctrl_2_microblaze_0_bram_block
 BUS_INTERFACE PORTB = microblaze_0_d_bram_ctrl_2_microblaze_0_bram_block
END

BEGIN microblaze
 PARAMETER INSTANCE = microblaze_0
 PARAMETER HW_VER = 8.50.c
 PARAMETER C_INTERCONNECT = 2
 PARAMETER C_USE_BARREL = 1
 PARAMETER C_USE_FPU = 0
 PARAMETER C_DEBUG_ENABLED = 1
 PARAMETER C_ICACHE_BASEADDR = 0X00000000
 PARAMETER C_ICACHE_HIGHADDR = 0X3FFFFFFF
 PARAMETER C_USE_ICACHE = 0
 PARAMETER C_ICACHE_ALWAYS_USED = 0
 PARAMETER C_DCACHE_BASEADDR = 0X00000000
 PARAMETER C_DCACHE_HIGHADDR = 0X3FFFFFFF
 PARAMETER C_USE_DCACHE = 0
 PARAMETER C_DCACHE_ALWAYS_USED = 0
 BUS_INTERFACE ILMB = microblaze_0_ilmb
 BUS_INTERFACE DLMB = microblaze_0_dlmb
 BUS_INTERFACE M_AXI_DP = axi4lite_0
 BUS_INTERFACE DEBUG = microblaze_0_debug
 BUS_INTERFACE INTERRUPT = axi_intc_0_INTERRUPT
 PORT MB_RESET = proc_sys_reset_0_MB_Reset
 PORT CLK = clk_100_0000MHz
END

BEGIN mdm
 PARAMETER INSTANCE = debug_module
 PARAMETER HW_VER = 2.10.a
 PARAMETER C_INTERCONNECT = 2
 PARAMETER C_USE_UART = 1
 PARAMETER C_BASEADDR = 0x41400000
 PARAMETER C_HIGHADDR = 0x4140ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 BUS_INTERFACE MBDEBUG_0 = microblaze_0_debug
 PORT Debug_SYS_Rst = proc_sys_reset_0_MB_Debug_Sys_Rst
 PORT S_AXI_ACLK = clk_100_0000MHz
END
```

```
BEGIN clock_generator
 PARAMETER INSTANCE = clock_generator_0
 PARAMETER HW_VER = 4.03.a
 PARAMETER C_CLKIN_FREQ = 100000000
 PARAMETER C_CLKOUT0_FREQ = 100000000
 PARAMETER C_CLKOUT0_GROUP = NONE
 PORT LOCKED = proc_sys_reset_0_Dcm_locked
 PORT CLKOUT0 = clk_100_0000MHz
 PORT RST = RESET
 PORT CLKIN = CLK
END

BEGIN axi_interconnect
 PARAMETER INSTANCE = axi4lite_0
 PARAMETER HW_VER = 1.06.a
 PARAMETER C_INTERCONNECT_CONNECTIVITY_MODE = 0
 PORT interconnect_aclk = clk_100_0000MHz
 PORT INTERCONNECT_ARESETN = proc_sys_reset_0_Interconnect_aresetn
END

BEGIN axi_intc
 PARAMETER INSTANCE = axi_intc_0
 PARAMETER HW_VER = 1.04.a
 PARAMETER C_BASEADDR = 0x41200000
 PARAMETER C_HIGHADDR = 0x4120ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 BUS_INTERFACE INTERRUPT = axi_intc_0_INTERRUPT
 PORT S_AXI_ACLK = clk_100_0000MHz
 PORT Intr = oktriggerins_0_INTR
END

BEGIN okwireins
 PARAMETER INSTANCE = okwireins_0
 PARAMETER HW_VER = 2.01.s
 PARAMETER C_NUM_REG = 32
 PARAMETER C_BASEADDR = 0x7d600000
 PARAMETER C_HIGHADDR = 0x7d60ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT S_AXI_ACLK = clk_100_0000MHz
 PORT okHE = net_okHe
END

BEGIN okwireouts
 PARAMETER INSTANCE = okwireouts_0
 PARAMETER HW_VER = 2.01.s
 PARAMETER C_NUM_REG = 32
 PARAMETER C_BASEADDR = 0x77000000
 PARAMETER C_HIGHADDR = 0x7700ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT S_AXI_ACLK = clk_100_0000MHz
 PORT okHE = net_okHe
 PORT okEH = okwireouts_0_okEH
END

BEGIN oktriggerins
 PARAMETER INSTANCE = oktriggerins_0
 PARAMETER HW_VER = 3.00.a
 PARAMETER C_NUM_REG = 32
 PARAMETER C_BASEADDR = 0x7a400000
 PARAMETER C_HIGHADDR = 0x7a40ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT S_AXI_ACLK = clk_100_0000MHz
 PORT INTR = oktriggerins_0_INTR
 PORT okHE = net_okHe
END

BEGIN oktriggerouts
 PARAMETER INSTANCE = oktriggerouts_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_NUM_REG = 32
 PARAMETER C_BASEADDR = 0x73e00000
 PARAMETER C_HIGHADDR = 0x73e0ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT S_AXI_ACLK = clk_100_0000MHz
 PORT okEH = oktriggerouts_0_okEH
 PORT okHE = net_okHe
END

BEGIN hdl_control
 PARAMETER INSTANCE = hdl_control_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_NUM_REG = 32
 PARAMETER C_BASEADDR = 0x7e200000
 PARAMETER C_HIGHADDR = 0x7e20ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT S_AXI_ACLK = clk_100_0000MHz
 PORT OUT0 = hdl_control_0_OUT0
END


BEGIN hdl_receive
 PARAMETER INSTANCE = hdl_receive_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_NUM_REG = 32
 PARAMETER C_BASEADDR = 0x7e220000
 PARAMETER C_HIGHADDR = 0x7e22ffff
 BUS_INTERFACE S_AXI = axi4lite_0
 PORT S_AXI_ACLK = clk_100_0000MHz
 PORT IN0 = hdl_receive_0_IN0
END
```

TestPeriph.c:

```
/*
 *
 * Xilinx, Inc.
 * XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS" AS A
 * COURTESY TO YOU.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION AS
 * ONE POSSIBLE   IMPLEMENTATION OF THIS FEATURE, APPLICATION OR
 * STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS IMPLEMENTATION
 * IS FREE FROM ANY CLAIMS OF INFRINGEMENT, AND YOU ARE RESPONSIBLE
 * FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE FOR YOUR IMPLEMENTATION
 * XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO
 * THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO
 * ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE
 * FROM CLAIMS OF INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.
 */

/*
 *
 *
 * This file is a generated sample test application.
 *
 * This application is intended to test and/or illustrate some
 * functionality of your system.  The contents of this file may
 * vary depending on the IP in your system and may use existing
 * IP driver functions.  These drivers will be generated in your
 * SDK application project when you run the "Generate Libraries" menu item.
 *
 */



#include <stdio.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "xintc.h"
#include "intc_header.h"
#include "oktriggerins.h"
#include "xil_exception.h"

void handle(void* data){
	Xil_Out32(XPAR_OKWIREOUTS_0_BASEADDR, *(u32*)data);
	OKTRIGGERINS_ClearTrigger(0x5f);
	Xil_Out32(0x73e00000, 0x1a3a115e);
```

```
		Xil_Out32(XPAR_OKWIREOUTS_0_BASEADDR + 8, Xil_In32(XPAR_OKWIREOUTS_0_BASEADDR +
8) + 1);
}

int main()
{

   static XIntc intc;

   Xil_ICacheEnable();
   Xil_DCacheEnable();

   print("---Entering main---\n\r");



   {
      int status;

      print("\r\n Running IntcSelfTestExample() for axi_intc_0...\r\n");

      status = IntcSelfTestExample(XPAR_AXI_INTC_0_DEVICE_ID);

      if (status == 0) {
         print("IntcSelfTestExample PASSED\r\n");
      }
      else {
         print("IntcSelfTestExample FAILED\r\n");
      }
```

```
    Xil_Out32(XPAR_OKWIREOUTS_0_BASEADDR + 8, 0x7ead1e);

    OKTRIGGERINS_Initialize(XPAR_OKTRIGGERINS_0_BASEADDR);
    OKTRIGGERINS_RegisterHandler(0x5f, 0xffffffff, handle, XPAR_OKWIREINS_0_BASEADDR);
    OKTRIGGERINS_SetInterruptMask(0x5f, 0xffffffff);
    OKTRIGGERINS_SetInterruptMask(0x40, 0xffffffff);

    Xil_Out32(XPAR_OKWIREOUTS_0_BASEADDR + 8, 0x7ead2e);

    XIntc_Initialize(&intc, XPAR_AXI_INTC_0_DEVICE_ID);
    XIntc_Connect(&intc, XPAR_OKTRIGGERINS_0_DEVICE_ID, OKTRIGGERINS_Handler, NULL);
    XIntc_Start(&intc, XIN_REAL_MODE);
    XIntc_Enable(&intc, XPAR_OKTRIGGERINS_0_DEVICE_ID);

    Xil_Out32(XPAR_OKWIREOUTS_0_BASEADDR + 8, 0x7ead3e);

    Xil_ExceptionInit();
    Xil_ExceptionRegisterHandler(XIL_EXCEPTION_ID_INT, XIntc_InterruptHandler, &intc);
    Xil_ExceptionEnable();

    Xil_Out32(XPAR_OKWIREOUTS_0_BASEADDR + 8, 0x7ead4e);

    while(1){
            Xil_Out32(XPAR_OKWIREOUTS_0_BASEADDR + 4, Xil_In32(XPAR_OKWIREINS_0_BASEADDR));
            Xil_Out32(0x7e200000, Xil_In32(0x7e220000));
    }
}
```

```
  /*
   * Peripheral SelfTest will not be run for debug_module
   * because it has been selected as the STDOUT device
   */



  print("---Exiting main---\n\r");

  Xil_DCacheDisable();
  Xil_ICacheDisable();

  return 0;
}
```

Test.v(testbench):

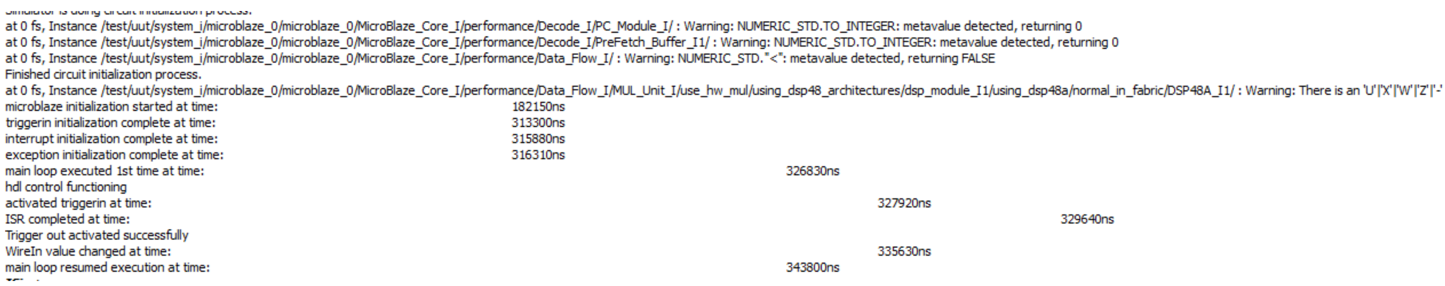
```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   13:03:33 06/05/2017
// Design Name:   system_top
// Module Name:   C:/Users/MBImager/Documents/MBImager_ams/ISE_projects/okinterruptsim/test.v
// Project Name:  okinterruptsim
// Target Device:
// Tool versions:
// Description:
```

```
//
// Verilog Test Fixture created by ISE for module: system_top
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module test;

	// Inputs
	reg RESET;
	reg CLK_P;
	reg CLK_N;
  wire [112:0] okHe;
	reg [31:0] wirein;

	// Outputs
	wire [64:0] okEH;
	wire [31:0] wireout;

	// Instantiate the Unit Under Test (UUT)
	system_top uut (
		.RESET(RESET),
		.CLK_P(CLK_P),
		.CLK_N(CLK_N),
                .okHe(okHe),
                .okwireouts_0_okEH_pin(okEHx[64:0]),
                .oktriggerouts_0_okEH_pin(okEHx[1*65 +: 65]),
                .hdl_receive_0_IN0_pin(wirein),
                .hdl_control_0_OUT0_pin(wireout)
        );
        //assign okEHx[65 +: 65] = 0;
        wire [2*65 - 1:0] okEHx;
        okWireOR #(.N(2)) wireor (.okEH(okEH), .okEHx(okEHx));

        reg [31:0] wire_through;
        reg [31:0] trigger_out;
        reg [31:0] setup_complete;

        initial wire_through = 0;
        initial trigger_out = 0;
        initial setup_complete = 0;

        integer temp;

        initial begin
                // Initialize Inputs
                RESET = 1;
                FrontPanelReset;
                wirein = 32'he110;
                // Wait 100 ns for global reset to finish
                #100;
    RESET = 0;
                // Add stimulus here
```

```
#1000;
SetWireInValue(8'h00, 32'h115a115e, 32'hffffffff);
UpdateWireIns();



while (setup_complete == 0) begin
        UpdateWireOuts();
        setup_complete = GetWireOutValue(8'h22);
end
$display ("microblaze initialization started at time:                %dns", $time);

while (setup_complete == 32'h7ead1e) begin
        UpdateWireOuts();
        setup_complete = GetWireOutValue(8'h22);
end
$display ("triggerin initialization complete at time:                %dns", $time);

while (setup_complete == 32'h7ead2e) begin
        UpdateWireOuts();
        setup_complete = GetWireOutValue(8'h22);
end
$display ("interrupt initialization complete at time:                %dns", $time);

while (setup_complete == 32'h7ead3e) begin
        UpdateWireOuts();
        setup_complete = GetWireOutValue(8'h22);
end
$display ("exception initialization complete at time:                %dns", $time);
```

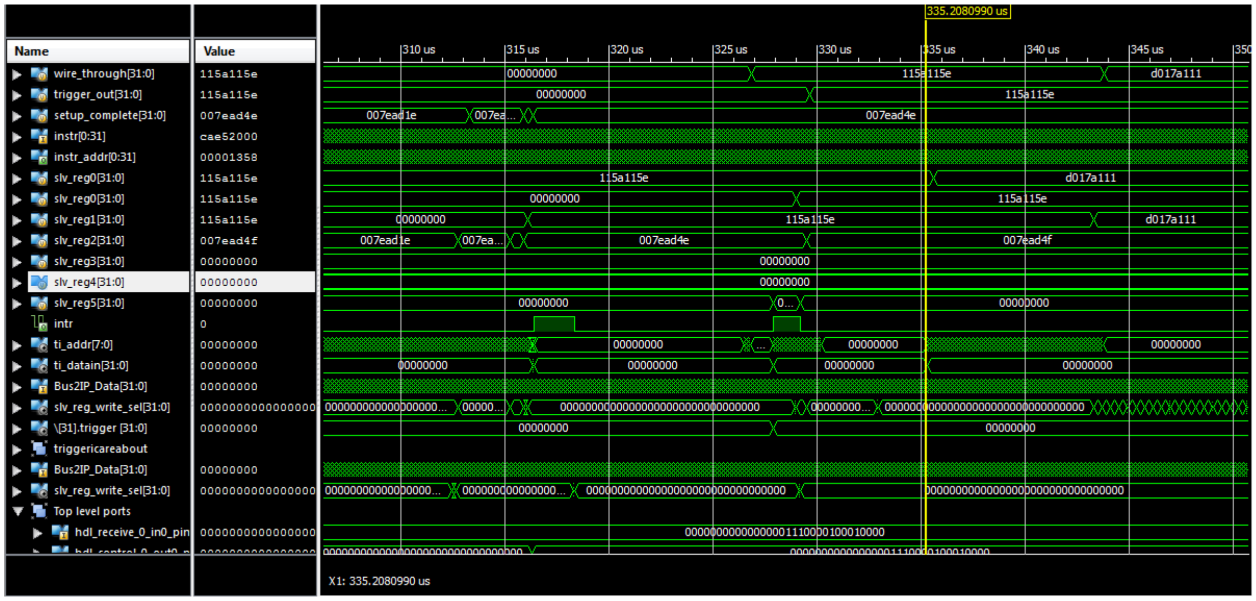
```
        ActivateTriggerIn(8'h40, 1);

        #10000;


        while (wire_through == 0) begin

                UpdateWireOuts();

                wire_through = GetWireOutValue (8'h21);

        end

        $display ("main loop executed 1st time at time:
%dns", $time);


        #1000;


        if (wireout == 32'he110)

                $display ("hdl control functioning");

        else

                $display ("hdl control failed");


        temp = GetWireOutValue (8'h20);

        ActivateTriggerIn(8'h5f, 0);

        $display ("activated triggerin at time:
%dns", $time);


        while (trigger_out == temp) begin

                UpdateWireOuts();

                trigger_out = GetWireOutValue (8'h20);

        end

        $display ("ISR completed at time:
                %dns", $time);
        UpdateTriggerOuts();

        if (IsTriggered(8'h60, 32'hffffffff))

                $display ("Trigger out activated successfully");

        else

                $display ("Trigger out failed to activate");


        #5000;


        SetWireInValue (8'h00, 32'hd017a111, 32'hffffffff);

        UpdateWireIns();

        $display ("WireIn value changed at time:
        %dns", $time);


        while (wire_through == 32'h115a115e) begin

                UpdateWireOuts();

                wire_through = GetWireOutValue (8'h21);

        end

        $display ("main loop resumed execution at time:
%dns", $time);


end


parameter HALF_PERIOD = 5;

initial CLK_P = 0;

initial CLK_N = 1;

always #HALF_PERIOD CLK_P = ~CLK_P;

always #HALF_PERIOD CLK_N = ~CLK_N;


parameter BlockDelayStates = 5;   // REQUIRED: # of clocks between blocks of pipe data
parameter ReadyCheckDelay = 5;    // REQUIRED: # of clocks before block transfer before
                                           //  host interface checks for ready (0-255)
parameter PostReadyDelay = 5;     // REQUIRED: # of clocks after ready is asserted and
              //  check that the block transfer begins (0-255)
parameter pipeInSize = 16;        // REQUIRED: byte (must be even) length of default
              //  PipeIn; Integer 0-2^32
parameter pipeOutSize = 16;       // REQUIRED: byte (must be even) length of default
              // PipeOut; Integer 0-2^32

parameter registerSetSize = 32;  // Size of array for register set commands.

//FrontPanel Data Structures for Pipes and Registers
reg [7:0] pipeIn [0:(pipeInSize-1)];
initial for (k=0; k<pipeInSize; k=k+1) pipeIn[k] = 8'h00; // Random words for DES encrypting
reg [7:0] pipeOut [0:(pipeOutSize-1)];
initial for (k=0; k<pipeOutSize; k=k+1) pipeOut[k] = 8'h00;
//Registers
reg [31:0] u32Address  [0:(registerSetSize-1)];
reg [31:0] u32Data     [0:(registerSetSize-1)];
reg [31:0] u32Count;

integer k;
`include "okHostCalls.v"
wire [31:0] okUHU;
wire [4:0] okUH;
wire [2:0] okHU;
okHost host (
        .okUHU(okUHU),
        .okUH(okUH),
                .okHU(okHU),

                .okEH(okEH),

                .okHE(okHe)

                );

endmodule
```

When simulating this design using ISIM, the console output looks like this:



```
at 0 fs, Instance /test/uut/system_i/microblaze_0/microblaze_0/MicroBlaze_Core_I/performance/Decode_I/PC_Module_I/ : Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
at 0 fs, Instance /test/uut/system_i/microblaze_0/microblaze_0/MicroBlaze_Core_I/performance/Decode_I/PreFetch_Buffer_I1/ : Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
at 0 fs, Instance /test/uut/system_i/microblaze_0/microblaze_0/MicroBlaze_Core_I/performance/Data_Flow_I/ : Warning: NUMERIC_STD."<": metavalue detected, returning FALSE
Finished circuit initialization process.
at 0 fs, Instance /test/uut/system_i/microblaze_0/microblaze_0/MicroBlaze_Core_I/performance/Data_Flow_I/MUL_Unit_I/use_hw_mul/using_dsp48_architectures/dsp_module_I1/using_dsp48a/normal_in_fabric/DSP48A_I1/ : Warning: There is an 'U'|'X'|'W'|'Z'|'-'
microblaze initialization started at time:                182150ns
triggerin initialization complete at time:                313300ns
interrupt initialization complete at time:                315880ns
exception initialization complete at time:                316310ns
main loop executed 1st time at time:                                        326830ns
hdl control functioning
activated triggerin at time:                                                    327920ns
ISR completed at time:                                                                          329640ns
Trigger out activated successfully
WireIn value changed at time:                                                   335630ns
main loop resumed execution at time:                                        343800ns
```

And the waveform output looks like this:



```
Name                        Value
wire_through[31:0]          115a115e
trigger_out[31:0]           115a115e
setup_complete[31:0]        007ead4e
instr[0:31]                 cae52000
instr_addr[0:31]            00001358
slv_reg0[31:0]              115a115e
slv_reg0[31:0]              115a115e
slv_reg1[31:0]              115a115e
slv_reg2[31:0]              007ead4f
slv_reg3[31:0]              00000000
slv_reg4[31:0]              00000000
slv_reg5[31:0]              00000000
intr                        0
ti_addr[7:0]                00000000
ti_datain[31:0]             00000000
Bus2IP_Data[31:0]           00000000
slv_reg_write_sel[31:0]     0000000000000000
\[31].trigger [31:0]        00000000
triggericareabout
Bus2IP_Data[31:0]           00000000
slv_reg_write_sel[31:0]     0000000000000000
Top level ports
  hdl_receive_0_in0_pin     0000000000000000
```