

Kevin Weldon

Software Debug and Control

January 2022



Goals

- To demonstrate a platform which can be used to set control registers and read/respond to status (CSR) registers
- Gain familiarity with Platform Designer as a tool to quickly design address mapped interfaces
- Learn the basics of Intel's Tcl API
- Design simple Nios V design to demonstrate CSR access
- Understand the Nios V tool flow
- Obtain complete working example designs of both the Tcl and C flows

Tool Utilized in Example Designs

- Platform Design
 - System integration tool used to automatically generate interconnect logic connect IP and subsystems
- Nios V
 - Next generation soft processor based on the open-source RISC-V Instruction Set
- JTAG to Avalon Bridge
 - IP core providing a connection between host systems (desktop computer) and an Avalon Interface based Platform Designer System
- Avalon Interface
 - Defines a memory mapped and stream interface standard to communicate between IP blocks in Platform Designer

Documentation / Instructions

- JTAG_Read_Write_Example_Guide_Tcl.doc
- NiosV_Read_Write_Example_Guide_C.doc

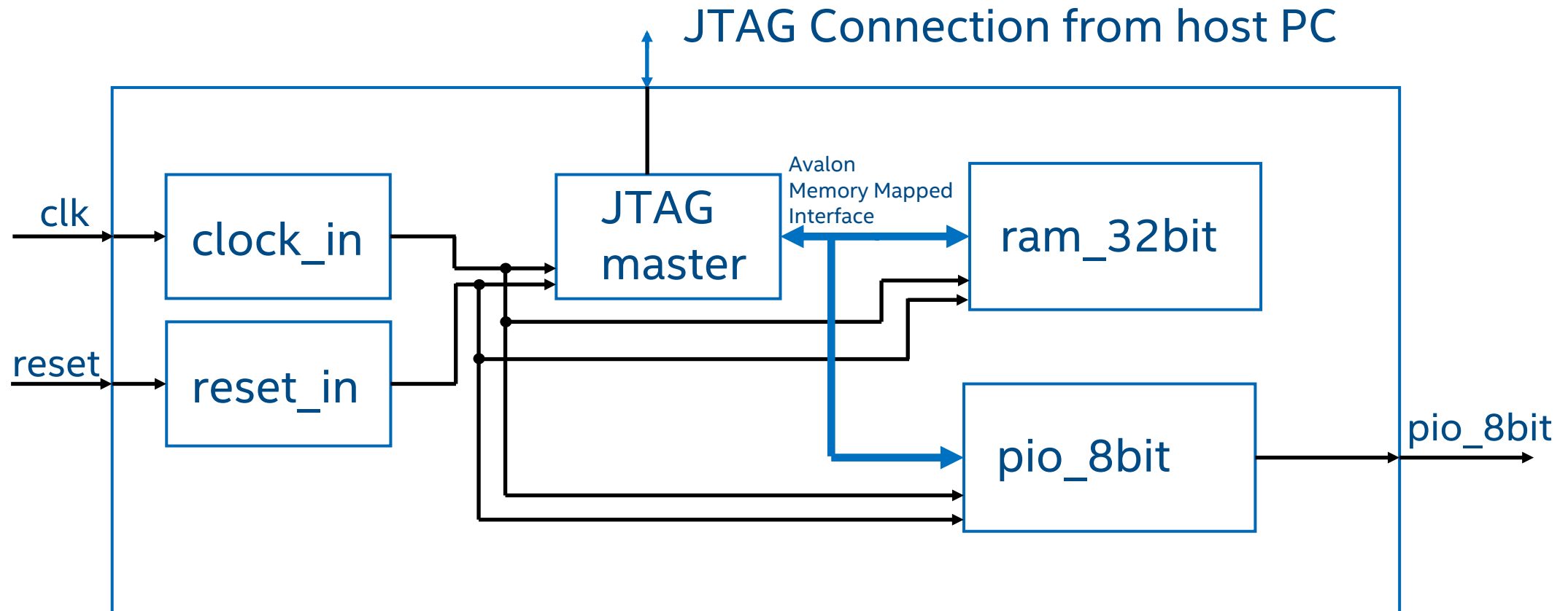
Debug and Control Using TCL



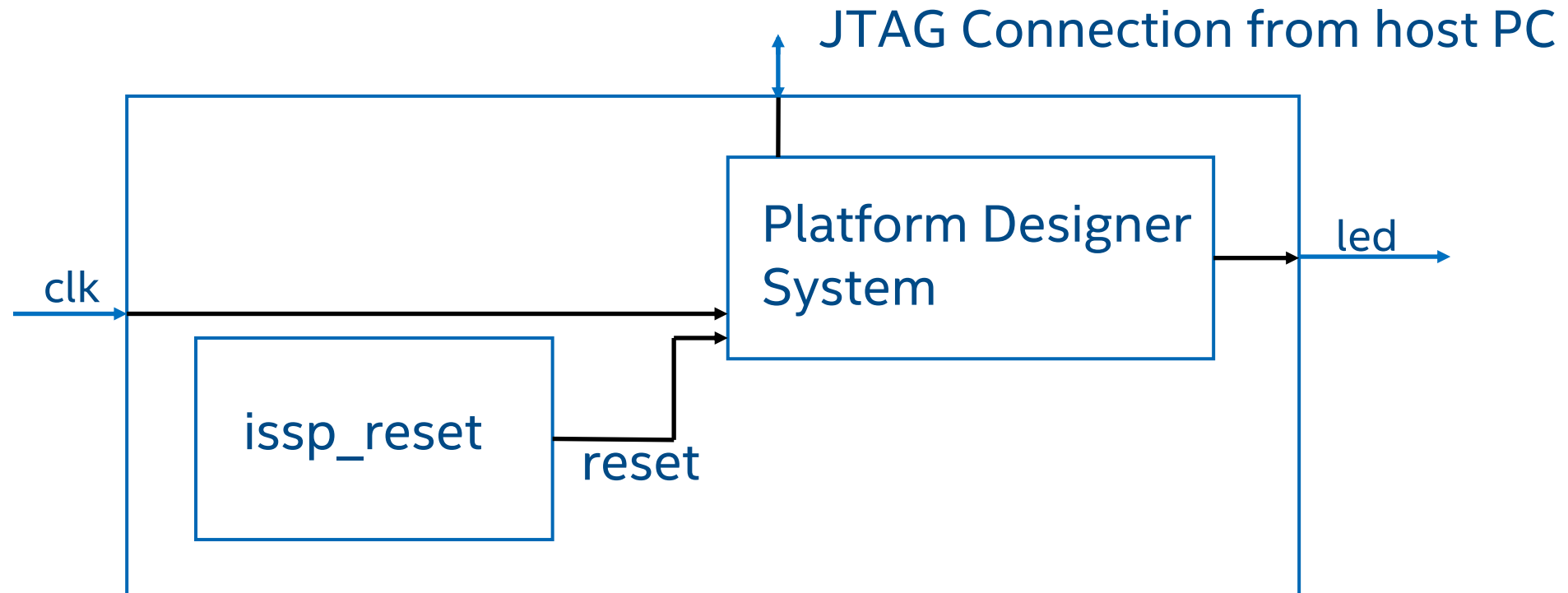
Platform Designer System – JTAG to Avalon Bridge

Use	Connections	Name	Description	Export
<input checked="" type="checkbox"/>		clock_in	Clock Bridge Intel FPGA IP	
		in_clk	Clock Input	clk
		out_clk	Clock Output	<i>Double-click to export</i>
<input checked="" type="checkbox"/>		reset_in	Reset Bridge Intel FPGA IP	
		clk	Clock Input	<i>Double-click to export</i>
		in_reset	Reset Input	reset
		out_reset	Reset Output	<i>Double-click to export</i>
<input checked="" type="checkbox"/>		master_0	JTAG to Avalon Master Bridge Intel FPGA IP	
		clk	Clock Input	<i>Double-click to export</i>
		clk_reset	Reset Input	<i>Double-click to export</i>
		master_reset	Reset Output	<i>Double-click to export</i>
		master	Avalon Memory Mapped Host	<i>Double-click to export</i>
<input checked="" type="checkbox"/>		ram_32bit	On-Chip Memory (RAM or ROM) Intel FPGA IP	
		clk1	Clock Input	<i>Double-click to export</i>
		s1	Avalon Memory Mapped Agent	<i>Double-click to export</i>
		reset1	Reset Input	<i>Double-click to export</i>
<input checked="" type="checkbox"/>		pio_8bit	PIO (Parallel I/O) Intel FPGA IP	
		clk	Clock Input	<i>Double-click to export</i>
		reset	Reset Input	<i>Double-click to export</i>
		s1	Avalon Memory Mapped Agent	<i>Double-click to export</i>
		external_connection	Conduit	pio_8bit

Platform Designer System – Block View



Top Level System – Block View



Reset Control Using Tcl

- toggle_issp.tcl script is used to toggle the reset register
 - The script makes a connection to the FPGA through JTAG
 - The In System Sources and Probes (ISSP) register instantiated in the design is then toggled, providing a reset to the system

```
start_insystem_source_probe -hardware_name $hardware2use -device_name $device
write_source_data -instance_index $ISSP_INDEX -value "1"
after $RESET_MS
write_source_data -instance_index $ISSP_INDEX -value "0"
end_insystem_source_probe
```

Debug & Control Tcl Script

- The read_write_example.tcl script is used to interface with the FPGA.

```
global PIO_8BIT_BASE
global RAM_32BIT_BASE

set PIO_8BIT_BASE 0x00000000
set RAM_32BIT_BASE 0x00000100

proc main {} {
    global PIO_8BIT_BASE
    global RAM_32BIT_BASE

    #####
    # Open the service path #
    #####
    set jm [open_path];
    puts "Opening master: $jm\n"
    open_service master $jm

    #####
    # Read and write 32-bit RAM #
    #####
    # Write to the first location in the 32-bit RAM
    master_write_32 $jm $RAM_32BIT_BASE 0x01234567
    # Write to the second location in the 32-bit RAM
    master_write_32 $jm [expr $RAM_32BIT_BASE+4] 0x89ABCDEF
    # Read the first location in the 32-bit RAM
    set read_value [master_read_32 $jm $RAM_32BIT_BASE 1]
    puts "Read $read_value at address $RAM_32BIT_BASE"
    # Read the second location in the 32-bit RAM
    set read_value [master_read_32 $jm [expr $RAM_32BIT_BASE+4] 1]
    puts "Read $read_value at address 0x[format %08X [expr $RAM_32BIT_BASE+4]]"

    puts "\nCtrl-C to exit.";














    #####
    # Blink the LED #
    #####
    while {1} {
        master_write_8 $jm $PIO_8BIT_BASE 0x01
        after 1000;
        master_write_8 $jm $PIO_8BIT_BASE 0x00
        after 1000;
    }

    return 0;
}

main;
```

Example Project Contents

Filename	Description
cleanup.bat	Removes all restored and generated files.
compile_design.bat	Compiles the FPGA from a Windows Prompt.
NiosV_Read_Write_Example_Guide.pdf	Description and instructions for the example design.
open_cmd_prompt.bat	Opens a Windows Command Prompt in the present working directory.
path_env_check.bat	Windows batch file to help check correctness of PATH environment variable.
program_fpga.bat	Windows batch file script to run the program_fpga.tcl and toggle_issp.tcl scripts from a Windows Prompt.
program_fpga.tcl	Tcl script to assist in the programming of the FPGA.
read_write_example.tcl	Software application file written in tcl.
restore_qar.bat	Windows batch file script to restore the Quartus project from the Quartus archive (.qar file)
run_app.bat	Windows batch file script to run the read_write_example.tcl application.
toggle_issp.tcl	Tcl script to toggle the internal reset register using In System Sources and Probes (ISSP).
top_21_4_0_67.qar	Quartus project archive file, including Platform Designer system and Verilog files.

-  cleanup.bat
-  compile_design.bat
-  NiosV_Read_Write_Example_Guide_TCL.pdf
-  open_cmd_prompt.bat
-  path_env_check.bat
-  program_fpga.bat
-  program_fpga.tcl
-  read_write_example.tcl
-  read_write_example.tcl~
-  restore_qar.bat
-  run_app.bat
-  toggle_issp.tcl
-  top_21_4_0_67.qar

Demo



intel[®]

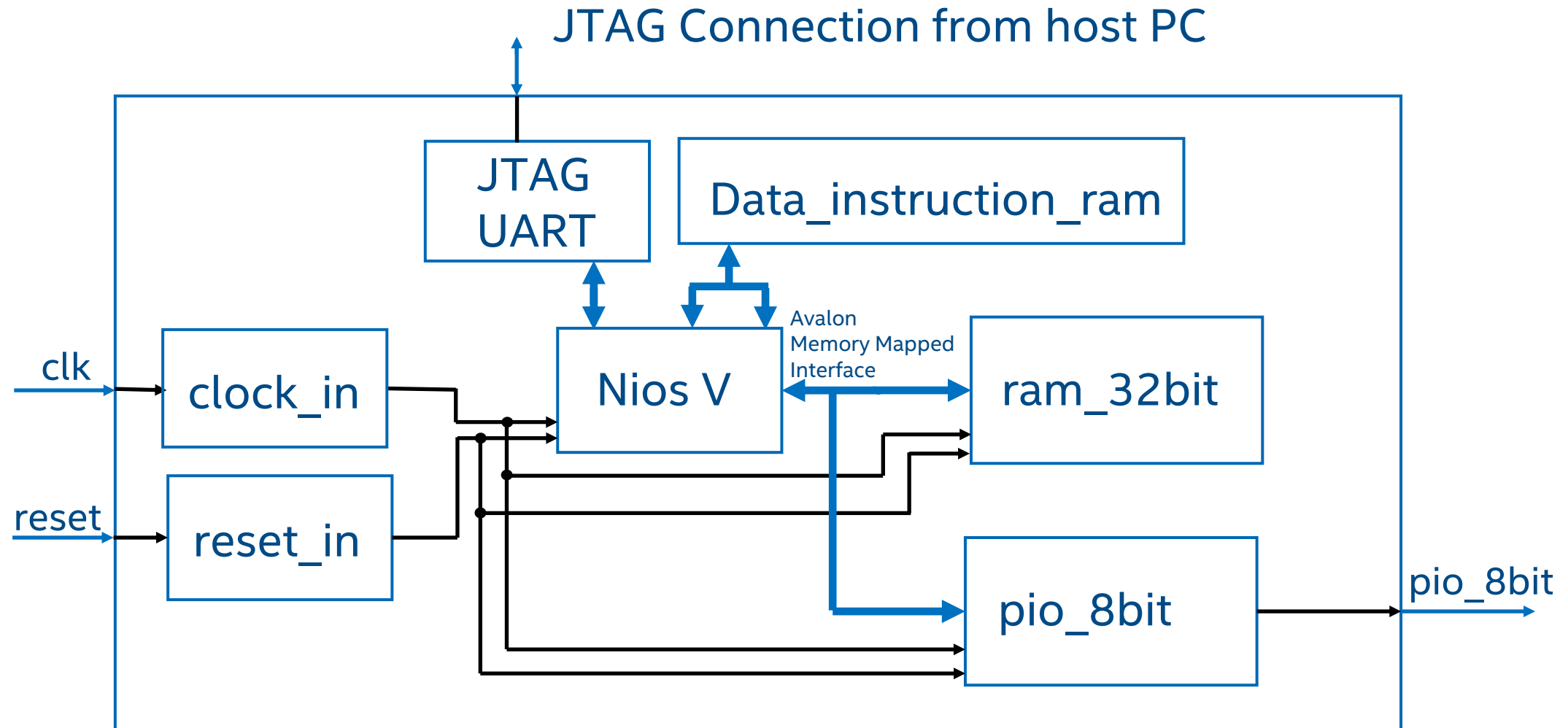
Debug and Control Using C



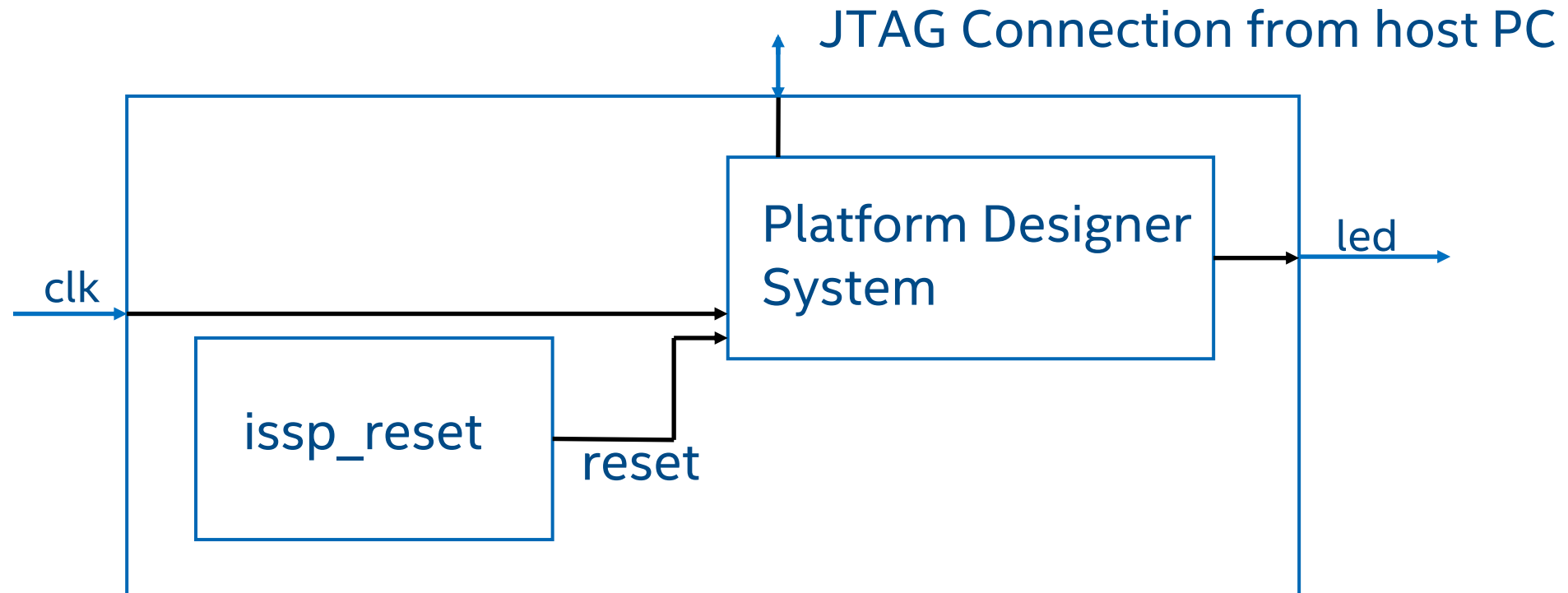
Platform Designer System – JTAG to Avalon Bridge

Connections	Name	Description	Export
	clock_in	Clock Bridge Intel FPGA IP	
	in_clk	Clock Input	clk
	out_clk	Clock Output	Double-click to export
	reset_in	Reset Bridge Intel FPGA IP	
	clk	Clock Input	Double-click to export
	in_reset	Reset Input	reset
	out_reset	Reset Output	Double-click to export
	cpu	Nios V/m Processor Intel FPGA IP	
	clk	Clock Input	Double-click to export
	reset	Reset Input	Double-click to export
	platform_irq_rx	Interrupt Receiver	Double-click to export
	instruction_manager	AXI4 Manager	Double-click to export
	data_manager	AXI4 Manager	Double-click to export
	timer_sw_agent	Avalon Memory Mapped Agent	Double-click to export
	dm_agent	Avalon Memory Mapped Agent	Double-click to export
	data_instruction_ram	On-Chip Memory (RAM or ROM) Intel FPGA...	
	clk1	Clock Input	Double-click to export
	s1	Avalon Memory Mapped Agent	Double-click to export
	reset1	Reset Input	Double-click to export
	jtag_uart	JTAG UART Intel FPGA IP	
	clk	Clock Input	Double-click to export
	reset	Reset Input	Double-click to export
	avalon_jtag_slave	Avalon Memory Mapped Agent	Double-click to export
	irq	Interrupt Sender	Double-click to export
	pio_8bit	PIO (Parallel I/O) Intel FPGA IP	
	clk	Clock Input	Double-click to export
	reset	Reset Input	Double-click to export
	s1	Avalon Memory Mapped Agent	Double-click to export
	external_connection	Conduit	pio_8bit
	ram_32bit	On-Chip Memory (RAM or ROM) Intel FPGA...	
	clk1	Clock Input	Double-click to export
	s1	Avalon Memory Mapped Agent	Double-click to export
	reset1	Reset Input	Double-click to export

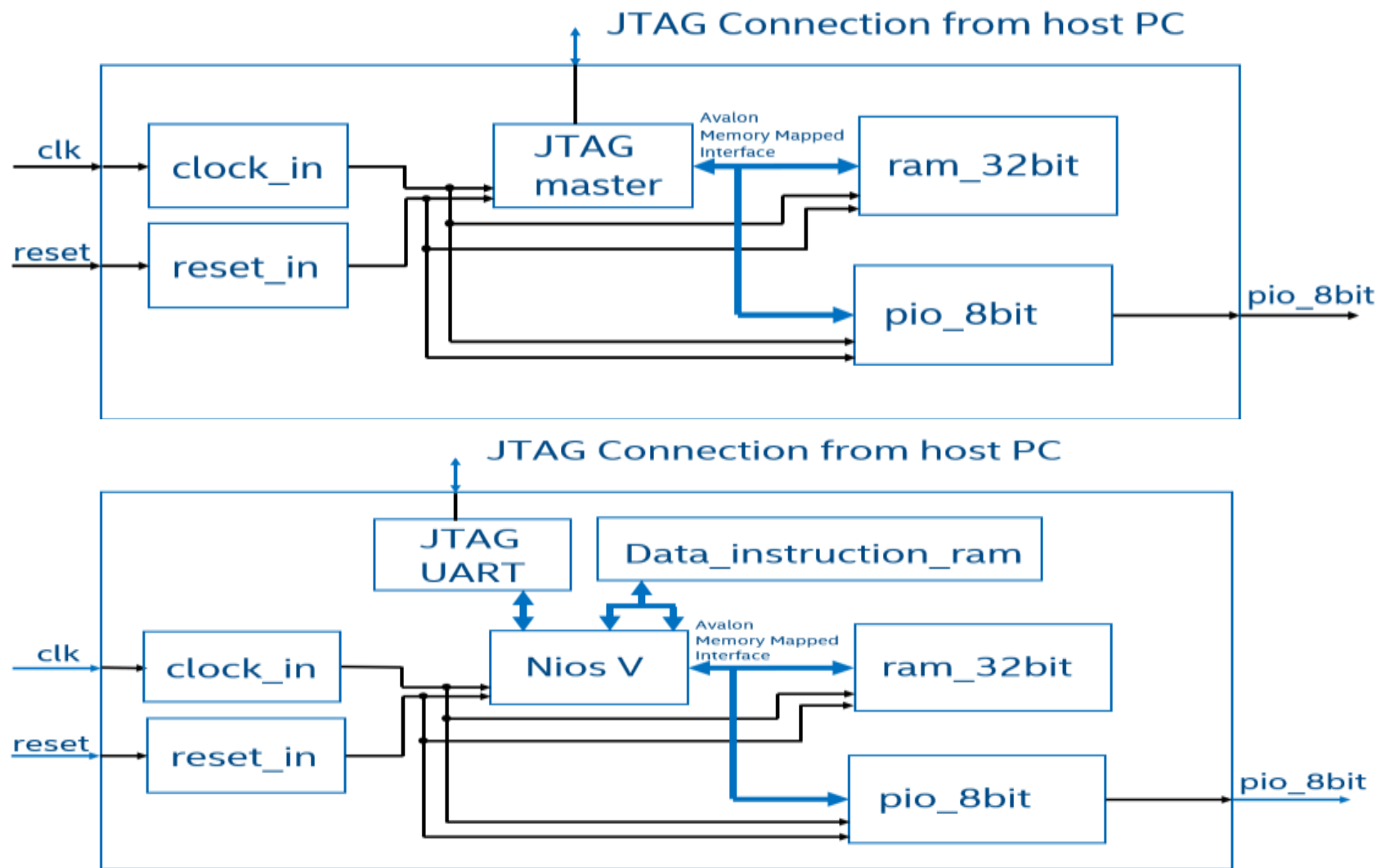
Platform Designer System – Block View



Top Level System – Block View



Block Diagram Comparison



Debug & Control C Program

- The read_write_example.c program is used to interface with the rest of the FPGA

IOWR_32DIRECT(RAM_32BIT_BASE, 0, 0x01234567);

- The above command is similar to the following C instruction:

(uint32_t)(RAM_32BIT_BASE)= 0x01234567;

- And is equivalent to the following Tcl API command:

master_write_32 \$jm \$RAM_32BIT_BASE 0x01234567

```
#include <stdio.h>
#include <io.h>
#include "system.h"
#include <unistd.h>

int main() {
    int read_value;

    ////////////////////////////////////
    // Read and write 32-bit RAM //2
    ////////////////////////////////////
    // Write to the first location in the 32-bit RAM
    IOWR_32DIRECT(RAM_32BIT_BASE, 0, 0x01234567);
    // Write to the second location in the 32-bit RAM
    IOWR_32DIRECT(RAM_32BIT_BASE, 4, 0x89ABCDEF);
    // Read the first location in the 32-bit RAM
    read_value = IORD_32DIRECT(RAM_32BIT_BASE, 0);
    printf("Read 0x%08X at address 0x%08X\n", read_value, RAM_32BIT_BASE);
    // Read the second location in the 32-bit RAM
    read_value = IORD_32DIRECT(RAM_32BIT_BASE, 4);
    printf("Read 0x%08X at address 0x%08X\n", read_value, RAM_32BIT_BASE+4);

    printf("\nCtrl-C to exit.\n");

    ////////////////////////////////////
    // Blink the LED //
    ////////////////////////////////////
    while (1) {
        IOWR_8DIRECT(PIO_8BIT_BASE, 0, 0x01);
        usleep(1000000);
        IOWR_8DIRECT(PIO_8BIT_BASE, 0, 0x00);
        usleep(1000000);
    }

    return 0;
}
```

Code comparison

```
#include <stdio.h>
#include <io.h>
#include "system.h"
#include <unistd.h>

int main() {
    int read_value;

    ////////////////////////////////////
    // Read and write 32-bit RAM //2
    ////////////////////////////////////
    // Write to the first location in the 32-bit RAM
    IOWR_32DIRECT(RAM_32BIT_BASE, 0, 0x01234567);
    // Write to the second location in the 32-bit RAM
    IOWR_32DIRECT(RAM_32BIT_BASE, 4, 0x89ABCDEF);
    // Read the first location in the 32-bit RAM
    read_value = IORD_32DIRECT(RAM_32BIT_BASE, 0);
    printf("Read 0x%08X at address 0x%08X\n", read_value, RAM_32BIT_BASE);
    // Read the second location in the 32-bit RAM
    read_value = IORD_32DIRECT(RAM_32BIT_BASE, 4);
    printf("Read 0x%08X at address 0x%08X\n", read_value, RAM_32BIT_BASE+4);

    printf("\nCtrl-C to exit.\n");

    ////////////////////////////////////
    // Blink the LED //
    ////////////////////////////////////
    while (1) {
        IOWR_8DIRECT(PIO_8BIT_BASE, 0, 0x01);
        usleep(1000000);
        IOWR_8DIRECT(PIO_8BIT_BASE, 0, 0x00);
        usleep(1000000);
    }

    return 0;
}
```

```
global PIO_8BIT_BASE
global RAM_32BIT_BASE

set PIO_8BIT_BASE 0x00000000
set RAM_32BIT_BASE 0x00000100

proc main {} {
    global PIO_8BIT_BASE
    global RAM_32BIT_BASE

    #####
    # Open the service path #
    #####
    set jm [open_path];
    puts "Opening master: $jm\n"
    open_service master $jm

    #####
    # Read and write 32-bit RAM #
    #####
    # Write to the first location in the 32-bit RAM
    master_write_32 $jm $RAM_32BIT_BASE 0x01234567
    # Write to the second location in the 32-bit RAM
    master_write_32 $jm [expr $RAM_32BIT_BASE+4] 0x89ABCDEF
    # Read the first location in the 32-bit RAM
    set read_value [master_read_32 $jm $RAM_32BIT_BASE 1]
    puts "Read $read_value at address $RAM_32BIT_BASE"
    # Read the second location in the 32-bit RAM
    set read_value [master_read_32 $jm [expr $RAM_32BIT_BASE+4] 1]
    puts "Read $read_value at address 0x[format %08X [expr $RAM_32BIT_BASE+4]]"

    puts "\nCtrl-C to exit.";

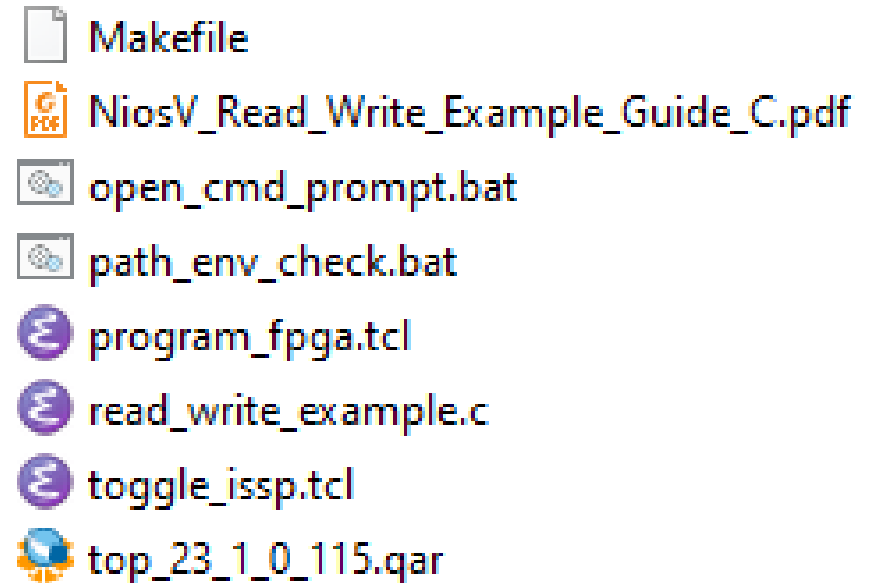
    #####
    # Blink the LED #
    #####
    while {1} {
        master_write_8 $jm $PIO_8BIT_BASE 0x01
        after 1000;
        master_write_8 $jm $PIO_8BIT_BASE 0x00
        after 1000;
    }

    return 0;
}

main;
```

Example Project Contents









Filename	Description
Makefile	Makefile to build and manage FPGA and software application projects.
NiosV_Read_Write_Example_Guide.pdf	This document.
open_cmd_prompt.bat	Opens a Windows Command Prompt in the present working directory.
path_env_check.bat	Window batch file to help check correctness of PATH environment variable.
program_fpga.tcl	Tcl script to assist in the programming of the FPGA.
read_write_example.c	Software application file written in C.
toggle_issp.tcl	Tcl script to toggle the internal reset register using In System Sources and Probes (ISSP).
top_23_1_0_115.qar	Quartus project archive file. Including Platform Designer system and top-level System Verilog file.



Software Required for Nios V

- a. GNU RISC-V Embedded GCC
 - i. <https://github.com/xpack-dev-tools/riscv-none-embed-gcc-xpack/releases/>
 - ii. Download file: [xpack-riscv-none-embed-gcc-10.2.0-1.2-win32-x64.zip](#)
 - iii. Extract file in Quartus install directory
 - 1. C:\intelFPGA_pro\23.1\niosv
- b. CMake packages for binary distributives
 - i. <https://cmake.org/download/>
 - ii. Download file: [cmake-3.26.3-windows-x86_64.zip](#)
 - iii. Extract file in Quartus install directory
 - 1. C:\intelFPGA_pro\23.1\niosv
- c. xPack Windows Build Tools
 - i. <https://github.com/xpack-dev-tools/windows-build-tools-xpack/releases/>
 - ii. Download file: [xpack-windows-build-tools-4.4.0-1-win32-x64.zip](#)
 - iii. Extract file in Quartus install directory
 - 1. C:\intelFPGA_pro\23.1\niosv

Software Required for Nios V

 > This PC > OSDisk (C:) > intelFPGA_pro > 23.1 > niosv		
Name	Date modified	Type
 bin	4/14/2023 9:01 AM	File folder
 cmake-3.26.3-windows-x86_64	5/5/2023 11:12 AM	File folder
 components	4/14/2023 9:01 AM	File folder
 examples	4/14/2023 9:01 AM	File folder
 scripts	4/14/2023 9:01 AM	File folder
 xpack-riscv-none-embed-gcc-10.2.0-1.2	5/5/2023 11:16 AM	File folder
 xpack-windows-build-tools-4.4.0-1	5/5/2023 11:11 AM	File folder

Path Environment Variable for Nios V

```
...  
%QUARTUS_PATH%\niosv\xpack-riscv-none-embed-gcc-10.2.0-1.2\bin  
%QUARTUS_PATH%\niosv\xpack-windows-build-tools-4.4.0-1\bin  
%QUARTUS_PATH%\niosv\cmake-3.26.3-windows-x86_64\bin  
%QUARTUS_PATH%\quartus\sopc_builder\bin  
%QUARTUS_PATH%\niosv\bin  
%QUARTUS_PATH%\quartus\bin64  
%QUARTUS_PATH%\syscon\bin  
...
```

Verify Path Correctness

```
C:\work\NiosV\NiosV_Read_Write_Example>path_env_check.bat

This script will check to ensure your PATH environment variable
is set correctly.

Testing xpack-riscv-none-embed-gcc path...
C:\intelFPGA_pro\21.4\niosv\xpack-riscv-none-embed-gcc-8.3.0-2.3\bin\riscv-none-embed-ar.exe

Testing xpack-windows-build-tools path...
C:\intelFPGA_pro\21.4\niosv\xpack-windows-build-tools-4.2.1-2\bin\make.exe
C:\cygwin64\bin\make.exe

Testing cmake path...
C:\intelFPGA_pro\21.4\niosv\cmake-3.21.4-windows-x86_64\bin\cmake.exe

Testing socp_builder path...
C:\intelFPGA_pro\21.4\quartus\socp_builder\bin\qsys-edit.exe

Testing niosv path...
C:\intelFPGA_pro\21.4\niosv\bin\niosv-app.exe

Testing quartus path...
C:\intelFPGA_pro\21.4\quartus\bin64\quartus_sh.exe

C:\work\NiosV\NiosV_Read_Write_Example>
```

```
Testing quartus path...
INFO: Could not find files for the given pattern(s).
```


Demo



intel[®]

