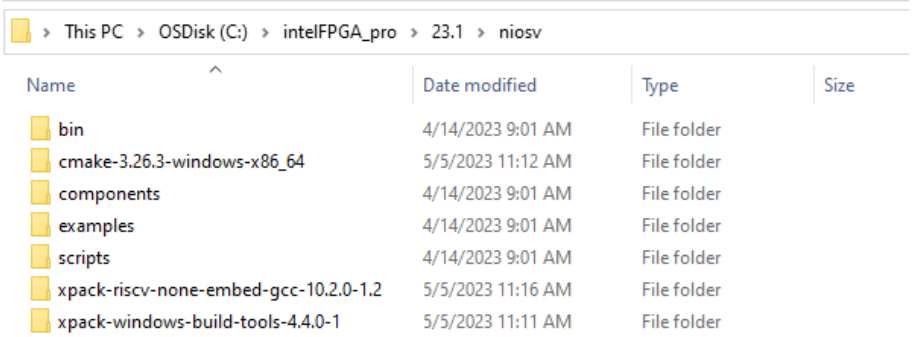
Nios V FIFO Interrupt Example Instructions

1. Quartus Prime Pro version 21.3 or newer is required for Nios V

2. Install required software

    a. A free Nios V license is required and can be obtained from the Self service licensing center.
    b. Installation of the RiscFree IDE may also be required. This can be installed during the Quartus installation or standalone.
    c. 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
    d. CMake packages for binary distributes
        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
    e. 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
    f. Upon completion of steps a-c your niosv/ directory should look like the following.



3. Edit Windows PATH environment variable.
    a. Open the Start Search, type in "env", and choose "Edit the system environment variables":
    b. Click the "Environment Variables…" button near the bottom.
    c. In the "User variables for <user>" frame highlight "Path" and then press "Edit..."
    d. Use the "New" or "Edit" button to ensure the following paths have been added.

```
%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%\nios2eds\bin
%QUARTUS_PATH%\quartus\sopc_builder\bin
%QUARTUS_PATH%\niosv\bin
%QUARTUS_PATH%\quartus\bin64
%QUARTUS_PATH%\syscon\bin
%QUARTUS_PATH%\nios2eds\sdk2\bin
%QUARTUS_PATH%\riscfree\toolchain\riscv32-unknown-elf\bin
```

4. Verify correctness of PATH environment variable
    a. Double click the open_cmd_prompt.bat file to open a Windows Command Prompt in the present working directory. From the Windows Command Prompt run path_env_check.bat. The output should look something like the following:

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

Testing riscvv32-unknown-elf-gcc path...
C:\intelFPGA_pro\23.2\riscfree\toolchain\riscv32-unknown-elf\bin\riscv32-unknown-elf-gcc.exe

Testing xpack-riscv-none-embed-gcc path...
C:\intelFPGA_pro\23.2\niosv\xpack-riscv-none-embed-gcc-10.2.0-1.2\bin\riscv-none-embed-ar.exe

Testing xpack-windows-build-tools path...
C:\intelFPGA_pro\23.2\niosv\xpack-windows-build-tools-4.4.0-1\bin\make.exe

Testing cmake path...
C:\intelFPGA_pro\23.2\niosv\cmake-3.26.3-windows-x86_64\bin\cmake.exe

Testing sopc_builder path...
C:\intelFPGA_pro\23.2\quartus\sopc_builder\bin\qsys-edit.exe

Testing niosv path...
C:\intelFPGA_pro\23.2\niosv\bin\niosv-app.exe

Testing quartus path...
C:\intelFPGA_pro\23.2\quartus\bin64\quartus_sh.exe
```

    If you see a “Could not find files...” message, you will need to correct your PATH environment variable prior to proceeding.

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

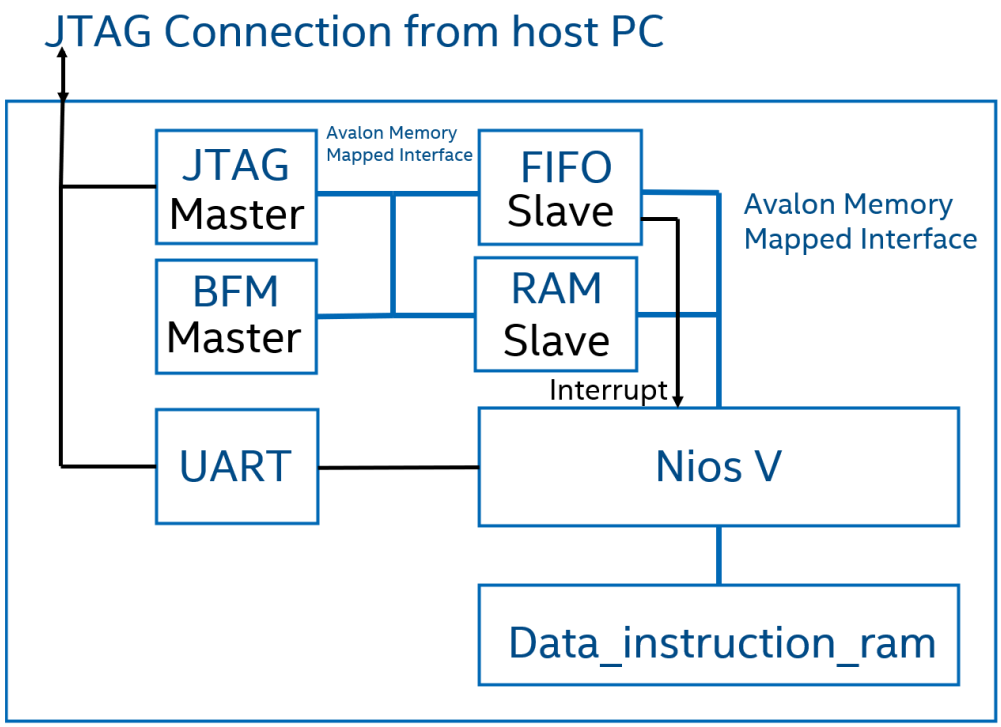
5. Unzip NiosV_Read_Write_Example Design.zip file
    a. Below is a description of the files included in the NiosV_Read_FIFO_Interrupt_Example directory.

Filename	Description
control.c	Software application file written in C.

Makefile	Makefile to build and manage FPGA and software application projects.
NiosV_FIFO_Interrupt_Example_Guilde.pdf	This document.
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.tcl	Tcl script to assist in the programming of the FPGA.
run_testbench.bat	Windows batch file to run testbench_code.tcl
simulation/	Simulation directory
testbench_code.tcl	System Console Tcl script used to communicate to the FPGA over JTAG.
toggle_issp.tcl	Tcl script to toggle the internal reset register using In System Sources and Probes (ISSP).
top_23_2_0_94.qar	Quartus project archive file. Including Platform Designer system and top-level System Verilog file.

6. Restore and View Quartus project
    a. From the command line type “make restore”
    b. Open the Quartus project and the Platform Designer system (sys.qsys). The system consists of the following:
        i. Clock and reset inputs
        ii. jtag_master – Creates an interface mapping between the System Console/JTAG and the Avalon Memory Mapped interface within the Platform Designer system.
        iii. mm_master_bfm – Avalon Memory Mapped Bus Functional Model used during simulation only. This module is synthesized away during Quartus compilation.
        iv. Nios V CPU
        v. data_instruction_ram - This RAM is used to hold the application code which the Nios V CPU will execute. During FPGA compilation the ./data_instruction_ram.hex file is located into this RAM.
        vi. jtag_uart - Used to capture STDIO from the application code.
        vii. command-fifo – FIFO used to receive packets from the jtag_master. The NiosV receives an interrupt from this FIFO when a complete packet is present.
        viii. CSR_RAM – Nios V writes packet data to this RAM based on the address in the packet.

Name	Description
clock_in	Clock Bridge Intel FPGA IP
reset_in	Reset Bridge Intel FPGA IP
jtag_master	JTAG to Avalon Master Bridge Intel FPGA IP
mm_master_bfm	Avalon Memory Mapped Master BFM Intel FPGA IP
intel_niosv_m	Nios V/m Processor Intel FPGA IP
data_instruction_ram	On-Chip Memory (RAM or ROM) Intel FPGA IP
jtag_uart	JTAG UART Intel FPGA IP
command_fifo	Avalon FIFO Memory Intel FPGA IP
CSR_RAM	On-Chip Memory (RAM or ROM) Intel FPGA IP



7. Compile Project
    a. The project is currently targeted towards an Arria 10 development board. You will need to change the target to match your board and update the location of the “clk” pin to match your board.
    b. To compile the FPGA, from the command line run “make fpga”. This will:
        i. Generate the Board Support Package
        ii. Generate the software application (create the data_instruction_ram.hex file)
        iii. Compile the FPGA

8. Program the FPGA
    a. Connect your computer to your board and run “make program” from the command line. This script will program the FPGA, toggle the built-in reset register (using In System Sources and Probes) and start the UART terminal. You should see the following output.

```
Capture stdout...
juart-terminal
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "USB-BlasterII [USB-1]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)


Entering control loop.
```

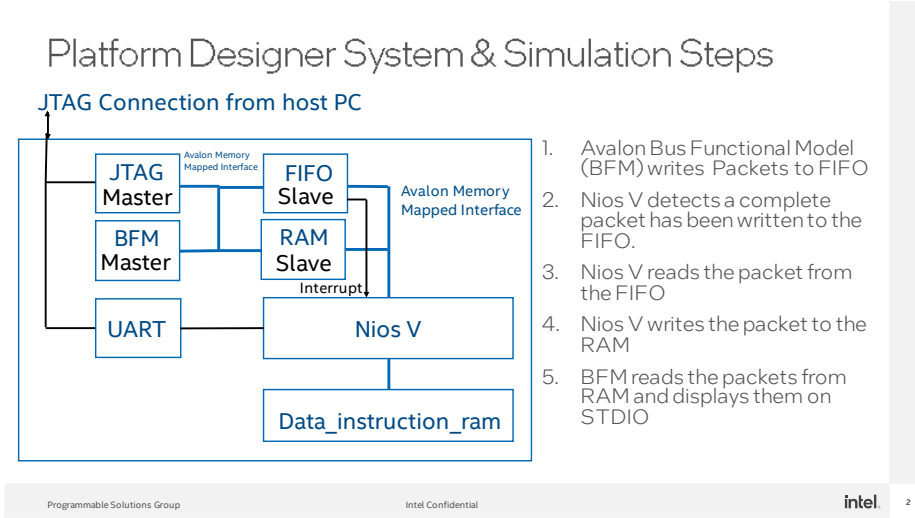
b. Open a second Command Prompt by double clicking on the open_cmd_prompt.bat file. In the second Command Prompt run the run_testbench.bat batch file. You should see the following output.

```
Wrote data 1048576 to address 0x00000000
Wrote data 1 to address 0x00000000
Wrote data 2 to address 0x00000000
Wrote data 3 to address 0x00000000
Wrote data 1048588 to address 0x00000000
Wrote data 4 to address 0x00000000
Wrote data 5 to address 0x00000000
Wrote data 6 to address 0x00000000
Read Data 0x00000001 from address 0x100000
Read Data 0x00000002 from address 0x100004
Read Data 0x00000003 from address 0x100008
Read Data 0x00000004 from address 0x10000c
Read Data 0x00000005 from address 0x100010
Read Data 0x00000006 from address 0x100014
```

In the Command Prompt connected to the Nios V you should now see the following output:

```
Entering control loop.
Interrupt Occured at address: 0x91040
FIFO has complete packet
Read from FIFO: 0x100000
Read from FIFO: 0x1
Read from FIFO: 0x2
Read from FIFO: 0x3
Interrupt Occured at address: 0x91040
FIFO has complete packet
Read from FIFO: 0x10000c
Read from FIFO: 0x4
Read from FIFO: 0x5
Read from FIFO: 0x6
```

9. Simulating the design in QuestaSim
   a. The following slide outlines the steps which occur during simulation as well as during hardware testing above.



b. Packet definition

1. A packet is defined as four, 32-bit words

Address
Data 0
Data 1
Data 2

c. To run the simulation, navigate to the ./simulation directory and type the following in the command prompt:

   i. %> quartus_sh --t runme.tcl

d. You should see the following output:

```
# Entering control loop.
# top_tb: [4000820.00ns] Wrote Data 00100000 to address 00000000, byteenable=f
# top_tb: [4000840.00ns] Wrote Data 00000001 to address 00000000, byteenable=f
# top_tb: [4000860.00ns] Wrote Data 00000002 to address 00000000, byteenable=f
# top_tb: [4000880.00ns] Wrote Data 00000003 to address 00000000, byteenable=f
# Interrupt Occured at address: 0x91040
# FIFO has complete packet
# Read from FIFO: 0x100000
# Read from FIFO: 0x1
# Read from FIFO: 0x2
# Read from FIFO: 0x3
# top_tb: [8000900.00ns] Wrote Data 0010000c to address 00000000, byteenable=f
# top_tb: [8000920.00ns] Wrote Data 00000004 to address 00000000, byteenable=f
# top_tb: [8000940.00ns] Wrote Data 00000005 to address 00000000, byteenable=f
# top_tb: [8000960.00ns] Wrote Data 00000006 to address 00000000, byteenable=f
# Interrupt Occured at address: 0x91040
# FIFO has complete packet
# Read from FIFO: 0x10000c
# Read from FIFO: 0x4
# Read from FIFO: 0x5
# Read from FIFO: 0x6
# top_tb: [16001030.00ns] Read Data 00000001 from address 00100000
# top_tb: [16001130.00ns] Read Data 00000002 from address 00100004
# top_tb: [16001230.00ns] Read Data 00000003 from address 00100008
# top_tb: [16001330.00ns] Read Data 00000004 from address 0010000c
# top_tb: [16001430.00ns] Read Data 00000005 from address 00100010
# top_tb: [16001530.00ns] Read Data 00000006 from address 00100014
# Simulation stopped at 16005530.00ns
# ** Note: $stop    : ./top_tb.sv(83)
#    Time: 16005530 ns  Iteration: 1  Instance: /top_tb
# Break in Task stop_sim at ./top_tb.sv line 83
# Stopped at ./top_tb.sv line 83
VSIM 2>
```