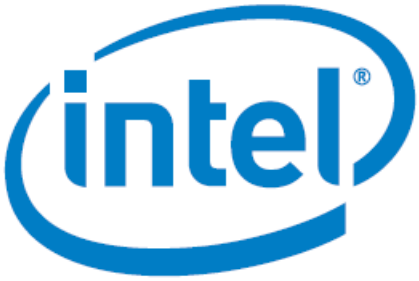


Intel® Agilex® 7 FPGA Custom Instruction Design on Nios® V/g processor - CRC

Date: 6/30/2023

Revision: 1

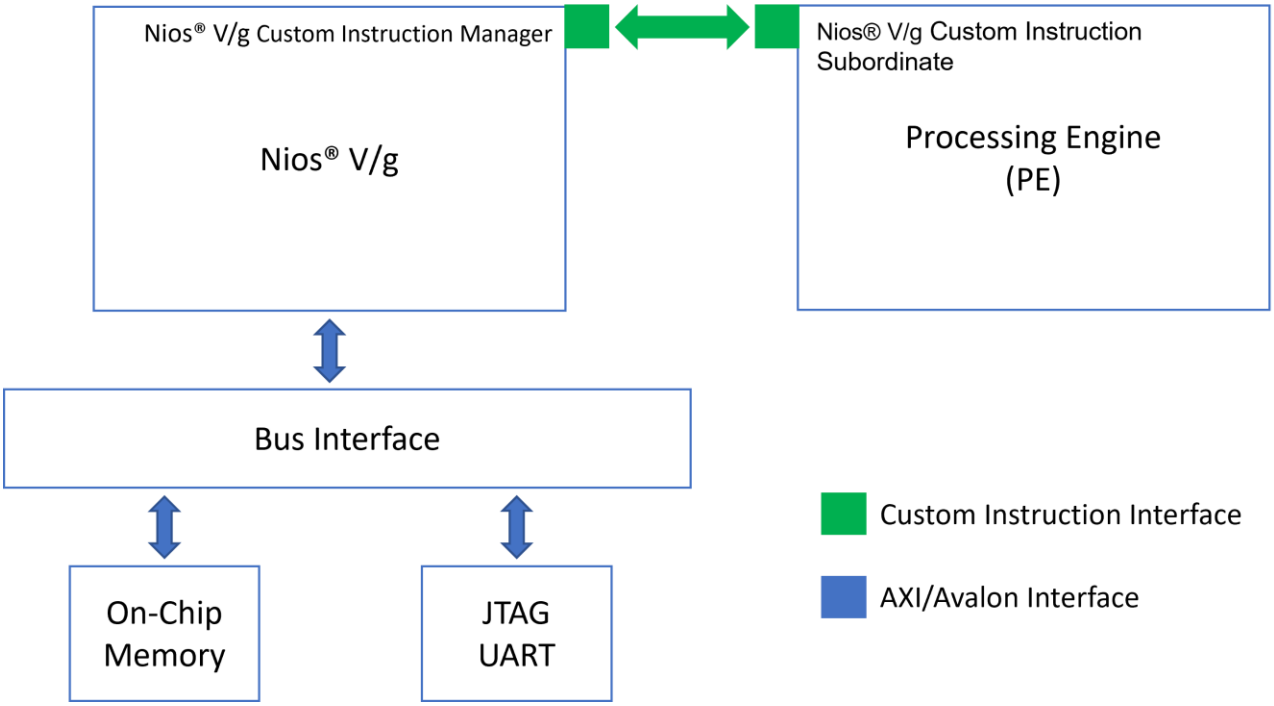
Contents

1. Theory of Operation

This design demonstrates the Cyclic Redundancy Check (CRC) algorithm using the custom instruction feature of the Nios® V/g processor using Intel Agilex® 7 F-Series FPGA Development Kit.

a. Block Diagram

A Processing Engine (PE) that performs the CRC algorithm is connected to the Nios® V/g processor using the custom instruction interface. The current version of the Nios® V/g processor custom instruction interface supports operations up-to 32-Bit.



b. IP Cores used

The following IPs are used in this design.

- NIOSV/g soft processor core
- On Chip RAM
- JTAG UART
- Custom PE CRC

2. Executing the Design on Devkit

a. Creating the Design

Note: Please refer to the readme.txt file in the package for the steps to create the design, application and generate the programing files.

- Unpackage/extract the design in your working directory

- Locate the “ready_to_test” folder within the package
- The folder contains the necessary files for executing the application on the board. Refer to the readme file for the steps to program the application files on the board.
- Validate the design by observing the prints on the terminal

b. Expected Results

The following is the output as observed on the JTAG UART terminal. The output is analogous to the logic from the application code. Users should be able to observe same output on their terminal/setup.

```
+-----------------------------------------------------------+
| Comparison between software and custom instruction CRC32  |
+-----------------------------------------------------------+


System specification
--------------------
System clock speed = 50 MHz
Number of buffer locations = 32
Size of each buffer = 256 bytes


Initializing all of the buffers with pseudo-random data
-------------------------------------------------------
Initialization completed


Running the software CRC
------------------------
Completed


Running the optimized software CRC
----------------------------------
Completed


Running the custom instruction CRC
----------------------------------
Completed


Validating the CRC results from all implementations
---------------------------------------------------
All CRC implementations produced the same results


Processing time for each implementation
---------------------------------------
Software CRC = 49 ms
Optimized software CRC = 82 ms
Custom instruction CRC = 01 ms


Processing throughput for each implementation
---------------------------------------------
Software CRC = 1337 Kbps
Optimized software CRC = 799 Kbps
Custom instruction CRC = 65536 Kbps


Speedup ratio
-------------
Custom instruction CRC vs software CRC = 49
Custom instruction CRC vs optimized software CRC = 82
Optimized software CRC vs software CRC= 0
```