

DE25-Nano Board

Demonstration Manual



FPGA

Contents

Chapter 1	Overview	3
Chapter 2	Examples For FPGA.....	4
2.1	DE25-Nano Factory Configuration	4
2.2	Basic Nios V control demo for Temperature/ Power/ Fan	6
2.3	Board Information IP	8
2.4	ADC_NiosV Reading.....	11
2.5	ADC_RTL	14
2.6	HDMI_TX.....	17
2.7	UART in Verilog	19
2.8	SDRAM_Test_RTL.....	20
2.9	RTL_LPDDR4_AXI4_Test	22
2.10	LPDDR4_Test_NiosV	25
Chapter 3	Examples for HPS SoC	29
3.1	I2C Interfaced G-sensor	29
3.2	Build C/C++ Project	32
Chapter 4	Program the QSPI Flash	34
Chapter 5	Additional Information.....	37
5.1	Getting Help.....	37



Chapter 1

Overview

This Manual will introduce the various application demonstrations on **DE25-Nano board**. These demonstrations cover most of the interfaces on the board. Let users familiarize using these interfaces of the board. Demonstrations according to FPGA fabrics and HPS are divided into two categories:

- **Pure use of FPGA fabric resources (Chapter 2)**
- **Pure use of HPS fabric resources (Chapter 3)**

Finally, to complete the following demonstration, user needs to install the following software in the computer:

- [Intel Quartus® Prime Pro Edition Software Version 25.1.1](#) or later.
- [Ashling* RiscFree* IDE for Altera](#)

Note: To run the demo bath file with the Nios V CPU of the demonstration on windows system, user need to install the Windows Subsystem for Linux (WSL) first then you can run the batch file. Please refer to the link to install: [Getting Start Install WSL](#)

Chapter 2

Examples

For FPGA

This chapter provides examples of advanced designs implemented by RTL or Qsys on the **DE25-Nano**. These reference designs cover the features of peripherals connected to the FPGA, such as ADC, HDMI, temperature monitor, PLL clock setting and Power monitor. All the associated files can be found in the directory **\Demonstrations\FPGA** of DE25-Nano System CD.

2.1 DE25-Nano Factory Configuration

The DE25-Nano board has a default configuration bitstream pre-programmed, which demonstrates some of the basic features on board. The setup required for this demonstration and the location of its files are shown below.

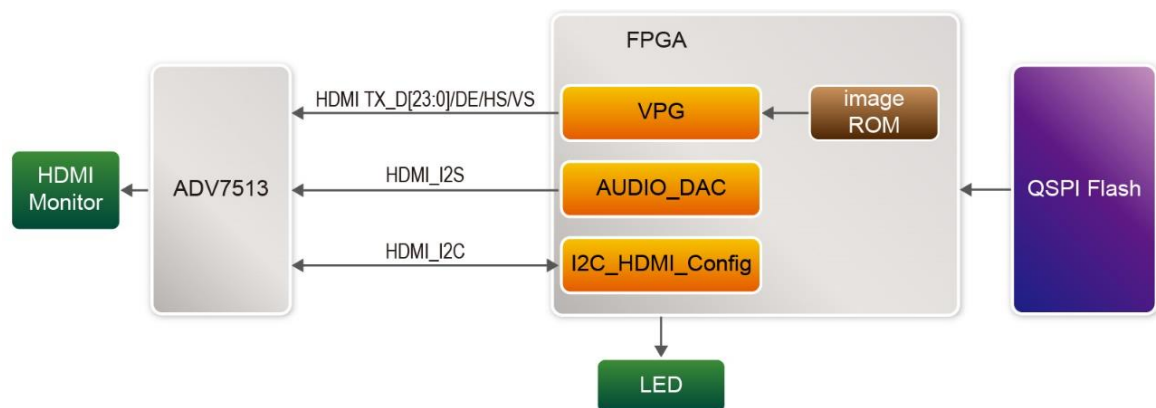


Figure 2-1 Block Diagram of the default demo

■ Design Tools

- Quartus Prime 25.1.1 Pro Edition

■ Demonstration Source Code

- Quartus Project directory: `\Demonstrations\HDMI_ASx4`
- Bitstream used: `golden_top.sof` or `golden_top.jic`

■ Demonstration Batch File

Demo Batch File Folder: `\Demonstrations\HDMI_ASx4\demo_batch`

The demo batch file includes following files:

- Demo Batch File : test.bat
- Convert jic Batch File: convert.bat
- Flash Batch File: flash_program.bat
- FPGA Configure File: golden_top sof, golden_top.jic

■ Demonstration Setup and Instructions

- Set SW5 MSEL[2:0] to 001, use a Type-C USB cable to connected the board(J2) to PC, and plug the 5V DC adapter to J11. use HDMI cable to connect the board to HDMI monitor. If necessary (that is, if the default factory configuration is not currently stored in the QSPI device), download the bit stream to the board via JTAG interface.
- You should now be able to observe the eight user LEDs are blinking, and the HDMI monitor displays the board image.
- The test.bat is executed to program .sof to the FPGA, the flash_program.bat is executed to program the .jic to the flash, the convert.bat is executed to convert .sof to .jic file then program the .jic to the flash.
- If users want to program a new design into the QSPI flash device, the easiest method is to copy the new .sof file to the demo_batch folder and execute the convert.bat to program the flash device.

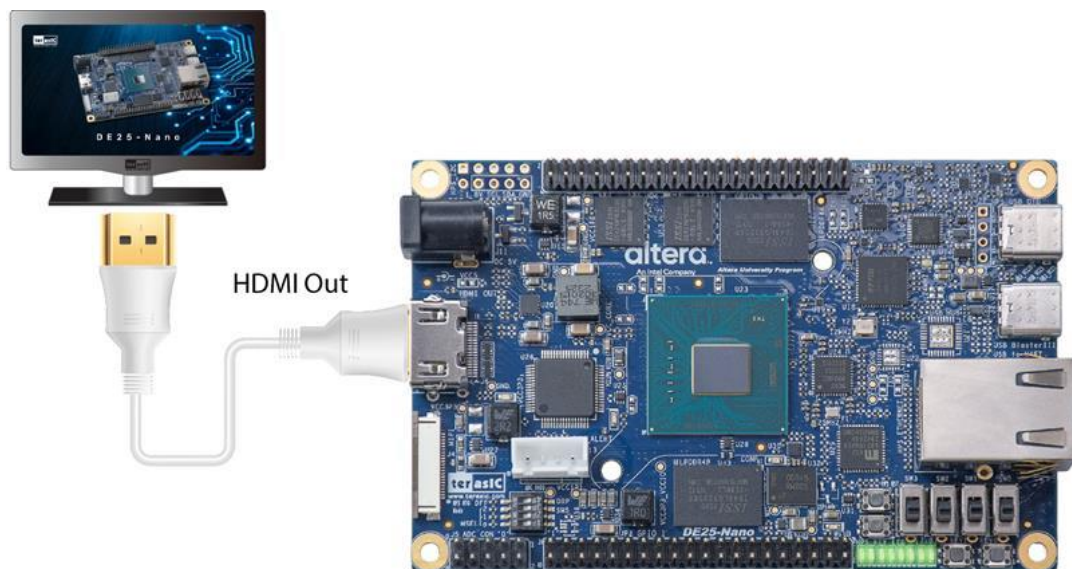


Figure 2-2 Setup for the default demo

2.2 Basic Nios V control demo for Temperature/ Power/ Fan

This demonstration shows how to use the Nios V processor to measure the power This demonstration shows how to use the Nios V processor to monitor board temperature and monitor/control fan speed. Terasic Board Management IP is used in this demonstration.

■ System Block Diagram

Figure 2-3 illustrates the system block diagram for this demonstration. The Terasic Board Management IP is responsible for directly monitoring and controlling the temperature sensor and fan controller. The Nios V processor communicates with the Board Management IP through the GPIO controllers.

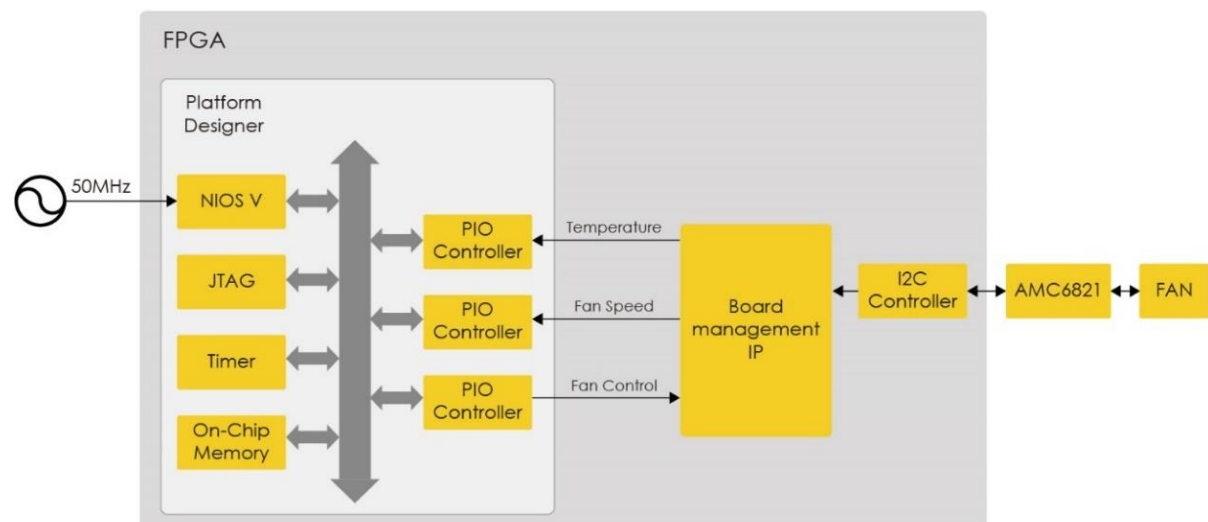


Figure 2-3 Block Diagram of the Nios V Demonstration

The program provides an interactive menu in the command-line window, as shown in

```
C:\WINDOWS\system32\cmd.exe
===== DE25-Nano Demo Program =====
[0] Show Board Info
[1] Set Fan Speed
Input your chioce:
```

Figure 2-4. Through this menu, users can query board information, such as temperature and fan speed, or set the fan speed manually.

Note: After entering a choice number, press **ENTER** to confirm the selection.

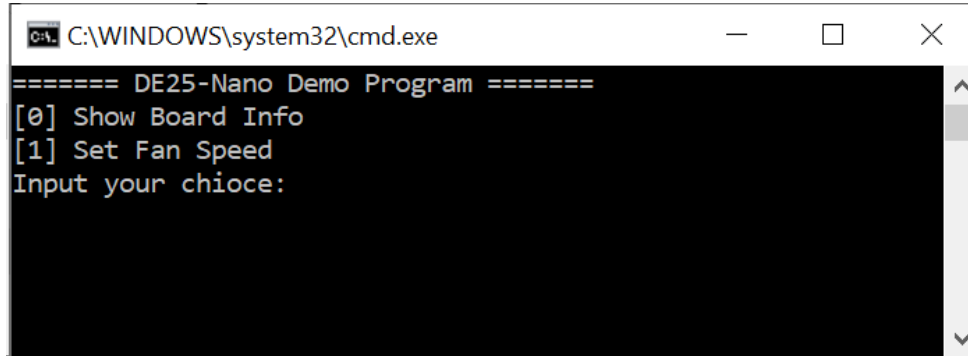


Figure 2-4 Menu of Demo Program

In the **Show Board Info** menu, the program displays the FPGA/board temperature and fan rotation speed. In the **Set Fan Speed** menu, the user can specify a fixed fan speed or select **Auto** mode. In **Auto** mode, the Board Management IP automatically adjusts the fan speed based on the FPGA and board temperatures.

■ Demonstration File Location

- Hardware project directory: Board_Info_NiosV
- Bitstream used: golden_top.sof and nios_app.elf
- Software project directory: Board_Info_NiosV\software
- Demo batch file: Board_Info_NiosV\demo_batch\test.bat

■ Install Ashling RiscFree IDE

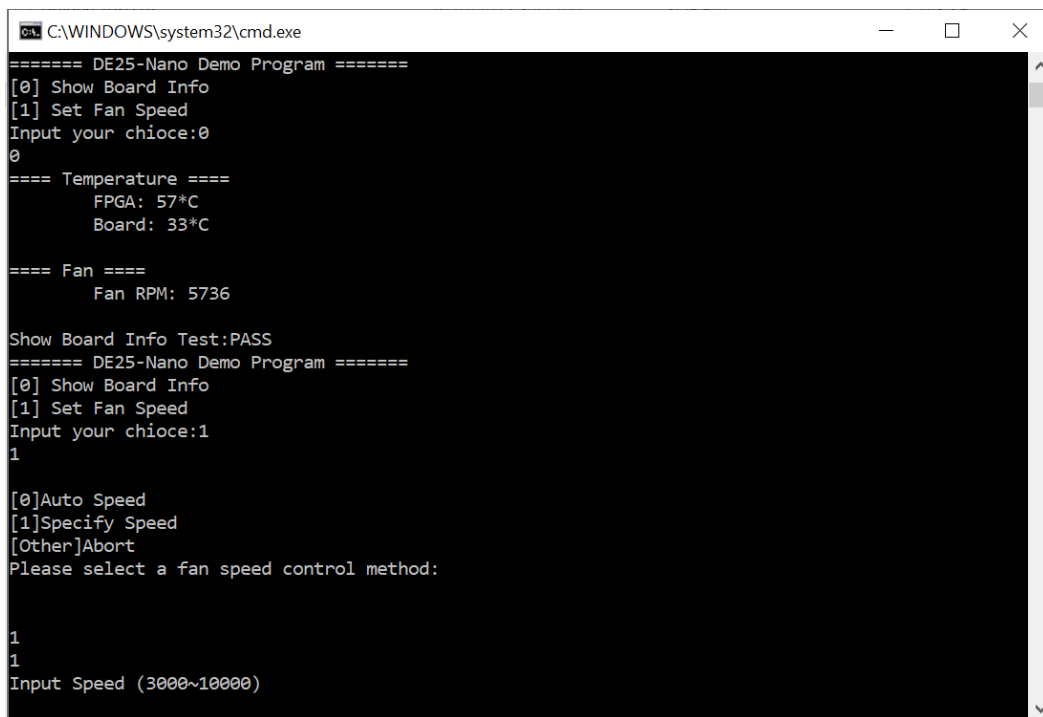
Before executing this demo with RISC-V core, users need to install Ashling RiscFree IDE for Intel® FPGAs to ensure that this batch file can be executed correctly. The file name should be *RiscFreeSetup-<Quartus Version>-windows.exe*. Users can find it under the [Quartus Pro download page](#) (Individual Files tab).

■ Demonstration Setup and Instructions

1. Make sure Quartus Prime is installed on the Host PC.
2. Use the USB Cable to connect your PC and the FPGA board and install USB Blaster III driver if necessary.
3. Power on the FPGA board.
4. Execute the demo batch file “test.bat” under the batch file folder: Board_Info_NiosV\demo_batch.
5. After the Nios V program is downloaded and executed successfully, a prompt

message will be displayed in command line window.

6. To monitor temperature and fan speed, enter '0' and press Enter in the nios-terminal, as shown in **Figure 2-5**.
7. To manually control the fan speed:
 - Enter '1' and press Enter in the nios-terminal.
 - A fan control menu will appear.
 - Enter '1' again and press Enter to enter manual control mode.
 - Input the desired fan speed and press Enter, as shown in **Figure 2-5**.



```
C:\WINDOWS\system32\cmd.exe
===== DE25-Nano Demo Program =====
[0] Show Board Info
[1] Set Fan Speed
Input your choice:0
0
==== Temperature ====
      FPGA: 57°C
      Board: 33°C

==== Fan ====
      Fan RPM: 5736

Show Board Info Test:PASS
===== DE25-Nano Demo Program =====
[0] Show Board Info
[1] Set Fan Speed
Input your choice:1
1

[0]Auto Speed
[1]Specify Speed
[Other]Abort
Please select a fan speed control method:

1
1
Input Speed (3000~10000)
```

Figure 2-5 Board Info Demo

2.3 Board Information IP

This section will introduce an IP which can be placed in the Agilex FPGA and allows users to obtain board status information such as temperature status and fan speed on the DE25-Nano board.

The DE25-Nano board provides several sensors to monitor the status of the board, such as FPGA/Board temperature, and fan speed status. The system primarily uses the I²C bus (HDMI_I2C) to communicate with on-board devices via the FPGA's I²C controller. It reads temperature data from the AMC6821(I²C slave address: 0x5C), including FPGA temperature and local-board temperature.

The system also interfaces with the AMC6821 to both configure the fan speed and read the current fan speed.

An automatic fan speed control function (AutoFan) adjusts the fan speed based on the temperature readings. This function can be enabled or disabled via the Auto_Fan_Speed (0 or 1):

- When set to 0, the fan speed is controlled manually by the user.
- When set to 1, the fan speed is automatically determined by the AutoFan logic and sent to the I2C controller

The Block as shown in **Figure 2-6**.

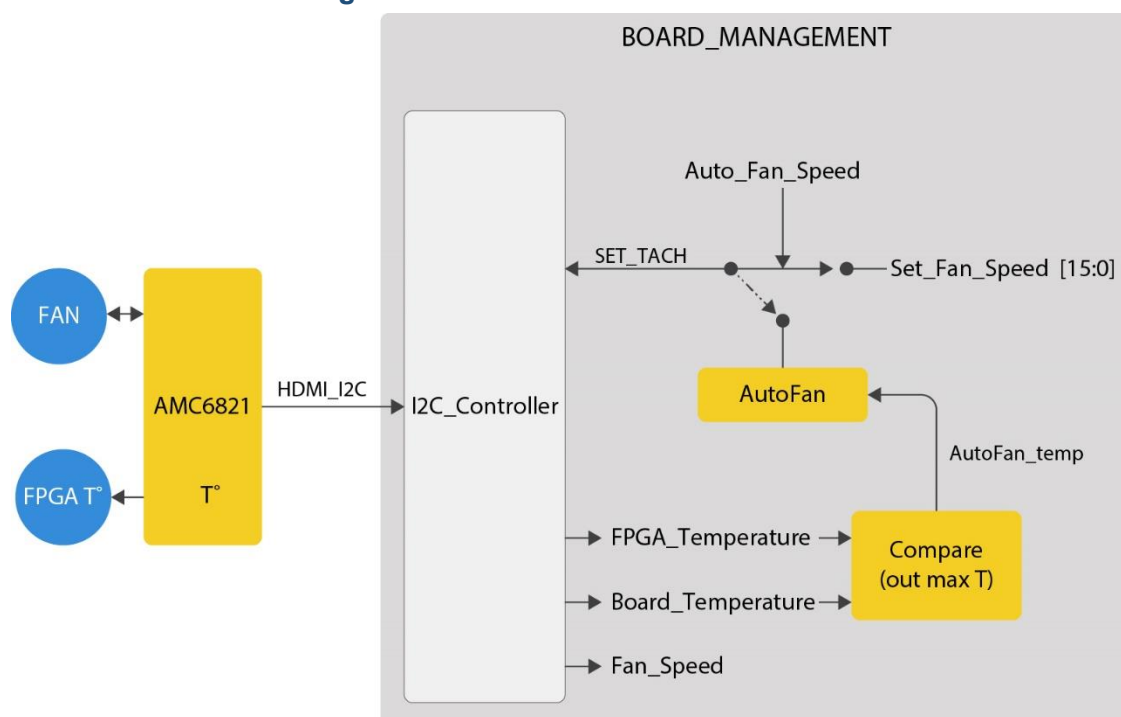


Figure 2-6 Block diagram of the fan speed control demonstration

User can place a board information IP (BOARD_MANAGEMENT.v) provided by Terasic in the Agilx FPGA, the board status can be obtained via I2C interface from the FPGA and output to user logic.

The board information IP can be obtained from the following path in the system CD:
Demonstration/FPGA/Board_info_RTL/board_management_ip/BOARD_MANAGEMENT.v

Figure 2-7 shows the input and output pins of the board information IP. Detailed pin descriptions and functions can be obtained from **Table 2-1**. Board information IP input and output ports. The user only needs to provide the IP 50Mhz clock and the reset control signal. The IP will automatically communicate with to get the board status value

via the I2C interface. When the logic level of the I2C_READY signal is from low to high, it means that the board status has been updated and can be used.

Finally, **Figure 2-8** shows the status of the IP during execution.

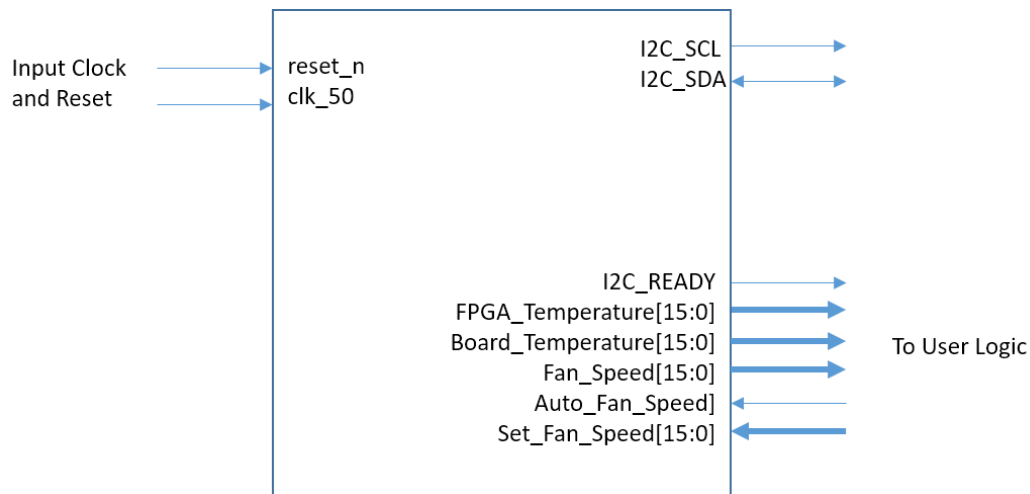


Figure 2-7 Pin out of the board information IP

Table 2-1 Board information IP input and output ports

Port Name	Direction	Width(Bit)	Description
clk_50	Input	1	Clock input for IP, please input 50Mhz clock.
reset_n	Input	1	Reset signal for IP, reset all logic.
I2C_SDA	BIR	1	Master I2C data . Please connect this signal to the HDMI_I2C_SDA pin.
I2C_SCL	Output	1	Master I2C clock, I2C master output to salve. Please connect this signal to the HDMI_I2C_SCL pin.
I2C_READY	Output	1	Information valid, logic high indicates board status updated ready.
Fan_Speed	Output	16	First fan speed of the board. The unit of the output value is RPM.
Board_Temperature	Output	16	First ambient temperature of the development board. The unit of the output value is Celsius.
FPGA_Temperature	Output	16	Core FPGA temperature of the development board. The unit of the output value is Celsius.
Auto_Fan_Speed	Input	1	1: enable audo speed control.0: user specify fan speed
Set_Fan_Speed	Input	16	Input rotational speed (unit: RPM)

Name	-128	0	128	256	384	512	640
⊕ u_BOARD_MANAGEMENT[FPGA_Temperature[15..0]]				60			
⊕ u_BOARD_MANAGEMENT[Board_Temperature[15..0]]				34			
⊕ u_BOARD_MANAGEMENT[Fan_Speed[15..0]]				91			
u_BOARD_MANAGEMENT[Auto_Fan_Speed]							
⊕ u_BOARD_MANAGEMENT[Set_Fan_Speed[15..0]]				9500			
u_BOARD_MANAGEMENT[I2C_READY]							

Figure 2-8 Waveform of the board status output

2.4 ADC_NiosV Reading

This demonstration illustrates steps to evaluate the performance of the 8-channel 12-bit analog-to-digital converter (ADC) TLA2518. The DC 5.0V on the 2x5 header is used to drive the analog signals by a trimmer potentiometer. The voltage should be adjusted within the range between 0 and 3.3V. The 12-bit voltage measurement is displayed on the NIOS terminal.

Figure 2-9 shows the block diagram of this demonstration. Nios V uses SPI (4-Wire Serial) IP to communicate with the TLA2518. Nios V program is running on on-chip memory. For Nios V processor users, Intel provides the HAL system library header file that defines the SPI core registers. The SPI core does not match the generic device model categories supported by the HAL, so it cannot be accessed via the HAL API or the ANSI C standard library. Intel provides a routine to access the SPI hardware that is specific to the SPI core.

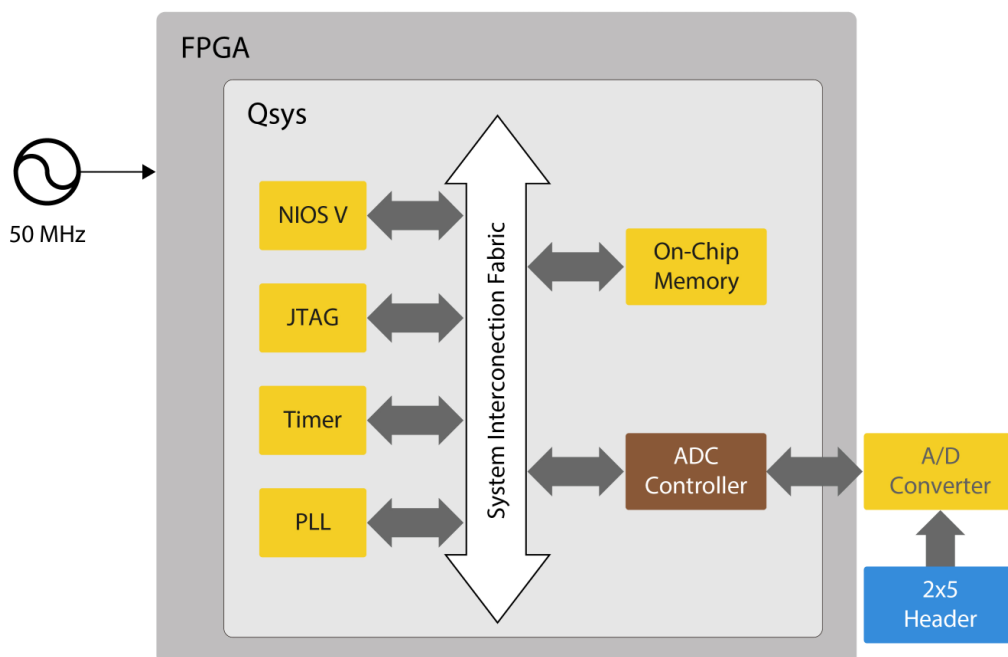


Figure 2-9 Block diagram of ADC reading

Figure 2-10 depicts the pin arrangement of the 2x5 header. This header is the input source of ADC convertor in this demonstration. Users can connect a trimmer to the specified ADC channel (ADC_IN0 ~ ADC_IN7) that provides voltage to the ADC convert. The FPGA will read the associated register in the convertor via serial interface and translates it to voltage value to be displayed on the Nios Terminal.

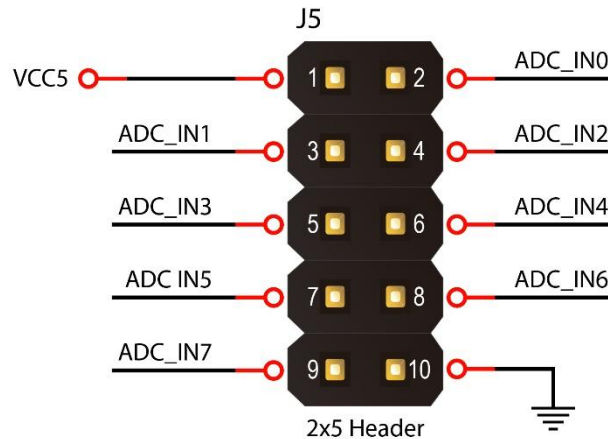


Figure 2-10 Pin distribution of the 2x5 Header for the ADC

Figure 2-11 shows the block diagram of ADC chip TLA2518. The TLA2518 is an easy-to-use, 1000ksps, 8-channel, multiplexed, 12-bit, 1-MSPS, successive approximation registers analog-to-digital converter (SAR ADC). The TLA2518 communicates via an SPI compatible interface and supports averaging multiple data samples with a single start of conversion.

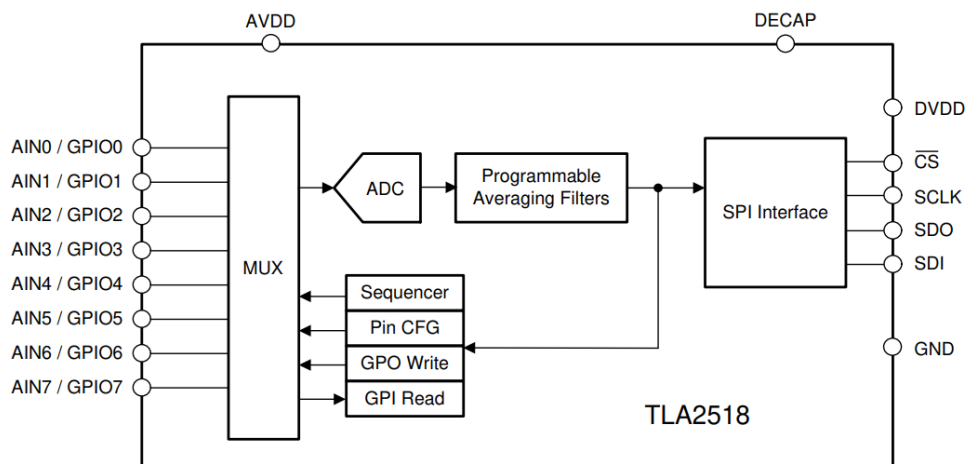


Figure 2-11 TLA2518 Block Diagram

■ System Requirements

- DE25-Nano board x1
- Trimmer Potentiometer x1

- Wire Strip x3

■ Demonstration File Locations

- Hardware project directory: ADC_NiosV
- Bitstream used: golden_top.sof
- Software project directory: ADC_NiosV\software
- Demo batch file : ADC_NiosV\demo_batch\test.bat

■ Install Ashling RiscFree IDE

Before executing this demo with RISC-V core, users need to install Ashling RiscFree IDE for Intel® FPGAs to ensure that this batch file can be executed correctly. The file name should be *RiscFreeSetup-<Quartus Version>-windows.exe*. Users can find it under the [Quartus Pro download page](#) (Individual Files tab).

■ Demonstration Setup

Please follow below procedures to set up the demonstrations.

1. Connect the trimmer to one of eight for ADC channel on the 2x5 header, as shown in **Figure 2-12**, as well as the +5V and GND signals. The setup shown below is connected to ADC channel 0. **Note, the input voltage do no exceeds 3.3V.**
2. Set SW[2:0] to position down/down/down to specify channel 0.
3. Connect a USB cable to the Type-C USB connector (J2) on the DE25-Nano board and the host PC.
4. Execute the demo batch file test.bat to load the bitstream and software execution file to the FPGA.
5. The Nios Terminal will display the voltage of the channel 0, as shown in **Figure 2-13**.
6. Provide any input voltage to other ADC channels and set SW[2:0] to the corresponding channel if user want to measure other channels.

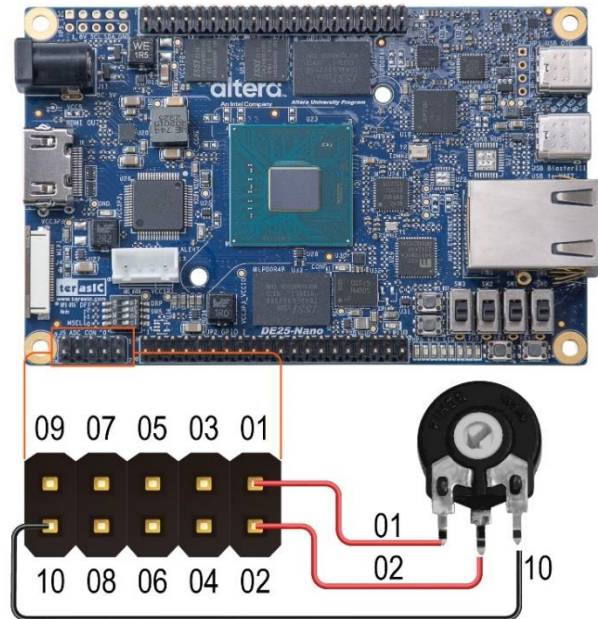


Figure 2-12 Hardware setup for the ADC reading demonstration

```

C:\WINDOWS\system32\cmd.exe
Loading section .text, size 0x1499c lma 0x40314
Loading section .rodata, size 0xa00 lma 0x54cb0
Loading section .rdata, size 0x18e0 lma 0x56f90
Start address 0x0040b00, load size 94096
Transfer rate: 109 KB/sec, 15682 bytes/write.
[Inferior 1 (Remote target) detached]
ADC Demo
ch0=3.05V (0ecah)
ch0=3.05V (0ec7h)
ch0=3.07V (0ee2h)
ch0=3.10V (0f06h)
ch0=3.15V (0f40h)
ch0=3.20V (0f83h)
ch0=3.24V (0fbah)
ch0=3.27V (0fdah)
ch0=3.27V (0fd8h)
ch0=3.24V (0fb7h)
ch0=3.21V (0f8eh)
ch0=3.16V (0f52h)
ch0=3.11V (0f0fh)
ch0=3.07V (0edfh)
ch0=3.04V (0eb8h)
ch0=3.03V (0eae h)
ch0=3.05V (0ec9h)
ch0=3.08V (0ef2h)
ch0=3.13V (0f26h)
ch0=3.18V (0f6ah)
ch0=3.24V (0fb0h)
ch0=3.26V (0fd2h)

```

Figure 2-13 ADC Channel 0 reading

2.5 ADC_RTL

This demonstration illustrates steps to evaluate the performance of the 8-channel 12-bit analog-to-digital converter (ADC) TLA2518. The DC 5.0V on the 2x5 header is used to drive the analog signals by a additional Voltage Divider Board. The input voltage is scaled down to eight discrete levels, approximately from 2.994V to 0.374V, and each

level is connected to ADC input channels ADC_IN0 through ADC_IN7, respectively. The 12-bit measurement is displayed on the Signal Tape Logic Analyzer.

Figure 2-14 shows the block diagram of this demonstration. Design an SPI (4-wire serial) IP core in RTL style to communicate with the TLA2518.

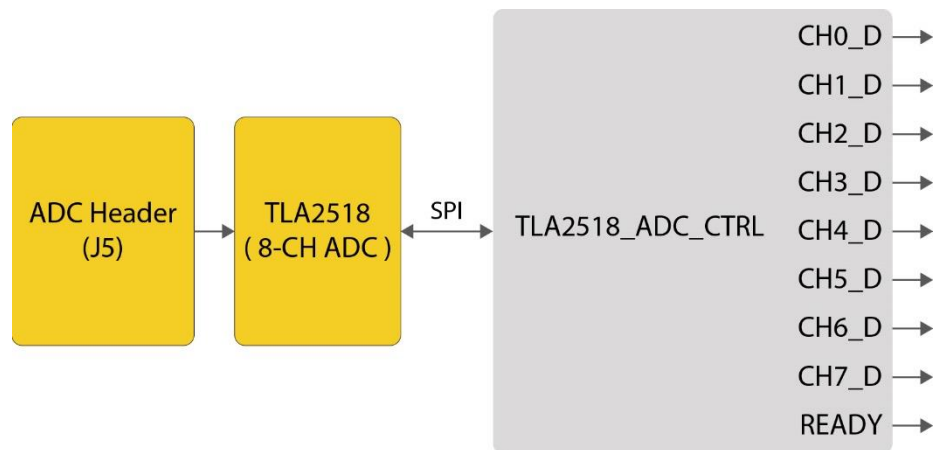


Figure 2-14 Block diagram of ADC reading

Figure 2-15 depicts the pin arrangement of the 2x5 header. This header is the input source of ADC converter in this demonstration. Users apply 8 individual voltages (each < 3.3V) to ADC_IN0 to ADC_IN7 as inputs for ADC conversion. The FPGA will read the associated register in the converter via serial interface and translates value to be displayed on the Signal Tape Logic Analyzer.

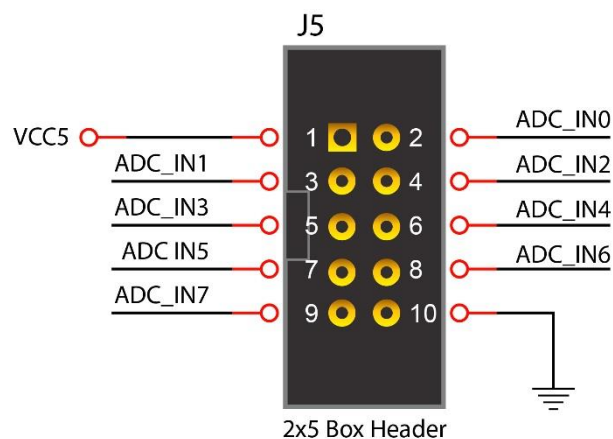


Figure 2-15 Pin distribution of the 2x5 Header for the ADC

Figure 2-16 shows the block diagram of ADC chip TLA2518. The TLA2518 is an easy-to-use, 1000ksps, 8-channel, multiplexed, 12-bit, 1-MSPS, successive approximation registers analog-to-digital converter (SAR ADC). The TLA2518 communicates via an SPI compatible interface and supports averaging multiple data samples with a single start of conversion.

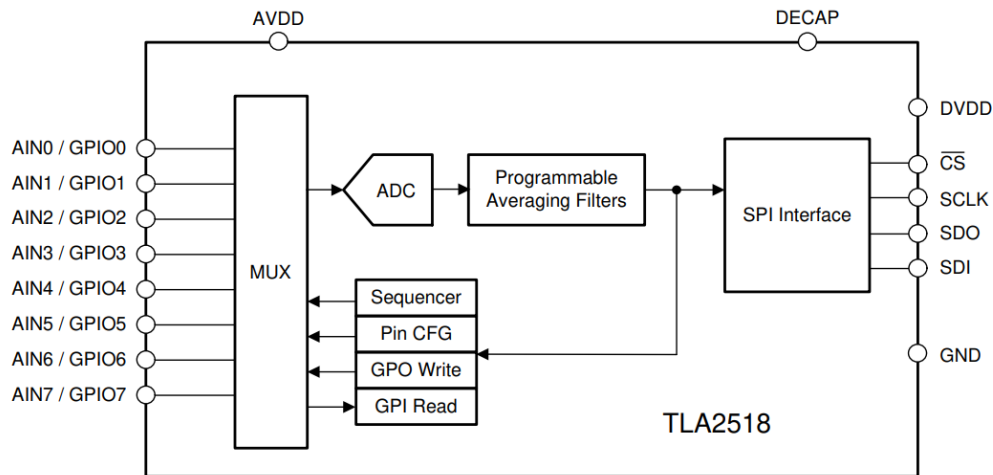


Figure 2-16 TLA2518 Block Diagram

■ System Requirements

- DE25-Nano board x1
- Voltage Divider Board x1

■ Demonstration Source Code

- Bitstream used: golden_top.sof
- Quartus Project directory: ADC_RTL

■ Demonstration Batch File

Demo Batch File Folder: ADC_RTL\demo_batch

The demo batch file includes following files:

- Batch File for USB Blaster III: test.bat
- FPGA Configure File: golden_top.sof
- Signal Tape File: stp1.stp

■ Demonstration Setup

Please follow below procedures to set up the demonstration

1. First, prepare an additional voltage divider board, as shown in **Figure 2-17**. Then, plug it into the J5 header on the DE25-Nano.
2. Connect a USB cable to the Type-C USB connector (J2) on the DE25-Nano board and the host PC.
3. Execute the demo batch file test.bat to load the bitstream to the FPGA.
4. Display the converted values of all ADC channels in the Quartus Signal Tap Analyzer, as shown in **Figure 2-18**.

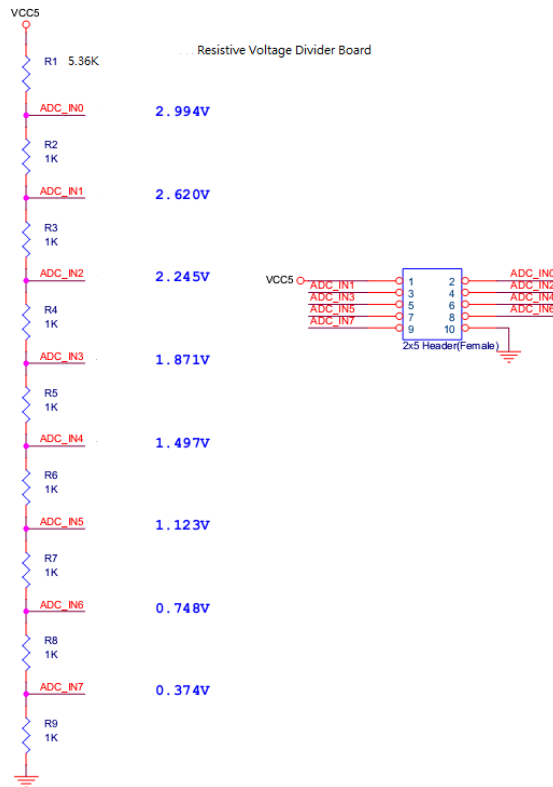


Figure 2-17 The additional voltage divider board is set up for the ADC_RTL demonstration

Name	128	256	384	512	640	768	896
...LA2518_ADC_CTRL CH0_D[11..0]					3840		
...LA2518_ADC_CTRL CH1_D[11..0]					3348		
...LA2518_ADC_CTRL CH2_D[11..0]					2873		
...LA2518_ADC_CTRL CH3_D[11..0]					2386		
...LA2518_ADC_CTRL CH4_D[11..0]					1913		
...LA2518_ADC_CTRL CH5_D[11..0]					1433		
...LA2518_ADC_CTRL CH6_D[11..0]					957		
...LA2518_ADC_CTRL CH7_D[11..0]					479		

Figure 2-18 ADC Channel 0~7 reading

2.6 HDMI_TX

This demonstration outputs a color bar to an HDMI monitor using the DE25-Nano board. **Figure 2-19** shows the block diagram of the design.

When the demonstration bitstream is loaded into FPGA. The I2C_HDMI_Config block will configure the HDMI transmitter chip (ADV7513) via I2C bus (I2C slave address=0x72). When SW[0] is set to 1, the AUDIO_DAC block generates a 1 kHz sine-wave tone, which is sent via I2S to the ADV7513. The VPG block outputs the color-bar

video signal to the ADV7513.

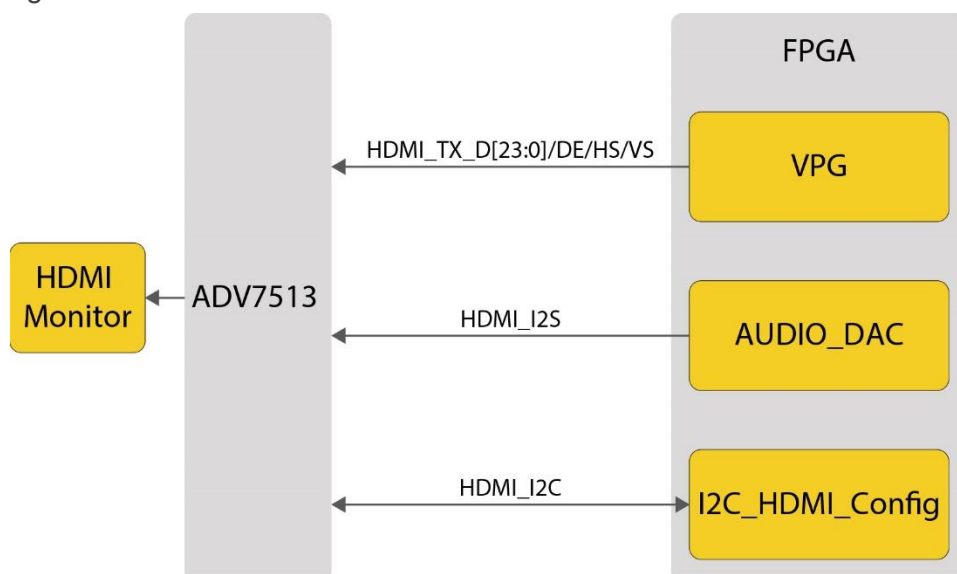


Figure 2-19 Block diagram of HDMI-TX demonstration

■ Demonstration Source Code

- Project directory: HDMI_TX
- Bitstream used: golden_top.sof

■ Demonstration File Locations

Demo batch directory: \HDMI_TX\demo_batch

The folder includes the following files:

- Batch file: test.bat
- FPGA configuration file : golden_top.sof

■ Demonstration Setup

Please follow below procedures to set up the demonstrations.

1. Connect the HDMI output(J6) of the DE25-Nano board to a HDMI interface monitor.
2. Connect a USB cable to the Type-C USB connector (J2) on the DE25-Nano board and the host PC.
3. Load the bitstream into the FPGA by executing the batch file 'test.bat' from the directory \HDMI_TX\demo_batch\.
4. **Figure 2-20** Setup for the HDMI_TX demonstration.

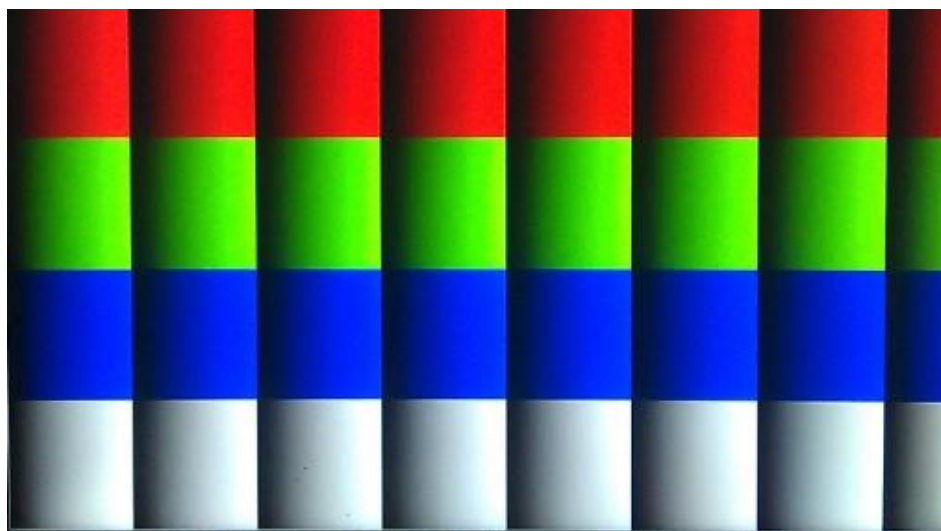


Figure 2-20 The video pattern in the HDMI_TX demonstration

2.7 UART in Verilog

DE25-Nano system CD offers UART loopback test code written in Verilog HDL.

■ System Block Diagram

Figure 2-21 shows the function block diagram of this demonstration. As shown in the block diagram, FPGA_UART_RX is the UART signal output from the FT2232 on the DE25-Nano input to the FPGA. Conversely, FPGA_UART_TX is the UART signal transmitted from the FPGA and input to the FT2232 on the DE25- Nano. KEY0 functions as a cutoff switch for the UART loopback.

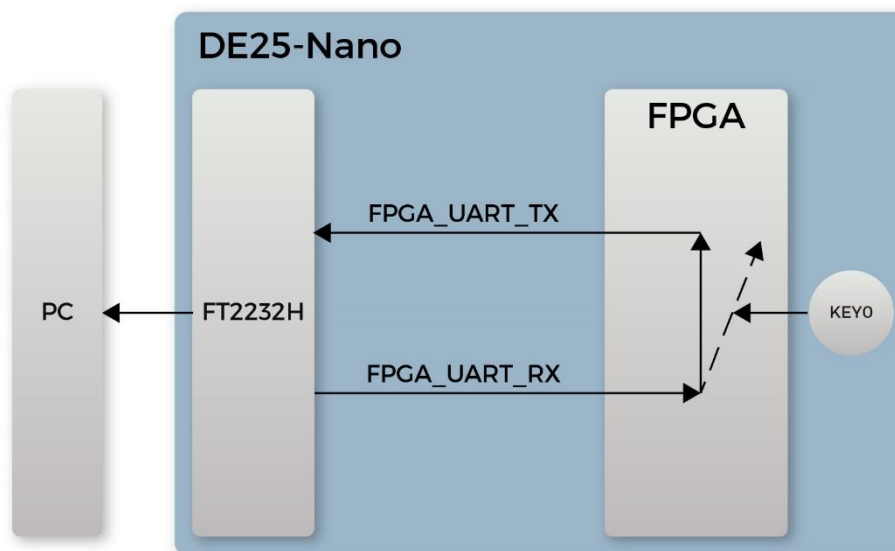


Figure 2-21 Block Diagram of the UART in Verilog

■ Design Tools

- Quartus Prime 25.1.1 Pro Edition

■ Demonstration Source Code

- Quartus Project directory: UART
- Bitstream used: golden_top.sof

■ Demonstration Batch File

Demo Batch File Folder: UART\demo_batch

The demo batch file includes following files:

- Demo Batch File : test.bat
- FPGA Configure File: golden_top.sof

■ Demonstration Setup and Instructions

- Please make sure both Quartus Pro v25.1.1 and UART FTDI driver are installed on the host PC.
- Power on the board.
- Execute the demo batch file “ test.bat” from the directory UART\demo_batch.
- Use a UART communication tool such as PuTTY on the PC. Select the correct COM port (the COM port number can be found in Windows Device Manager), set the baud rate to 115200, and open the UART terminal window. Typing characters from the PC keyboard should result in echoed characters appearing in the terminal. Press KEY0 to verify whether the UART transmission is interrupted.

2.8 SDRAM_Test_RTL

DE25-Nano system CD offers another SDRAM test with its test code written in Verilog HDL. The memory size of the SDRAM bank tested is still total 32Mx16bx2.

■ System Block Diagram

Figure 2-22 shows the function block diagram of this demonstration. The SDRAM controller uses 50MHz as a reference clock and generates 125MHz as the memory clock.

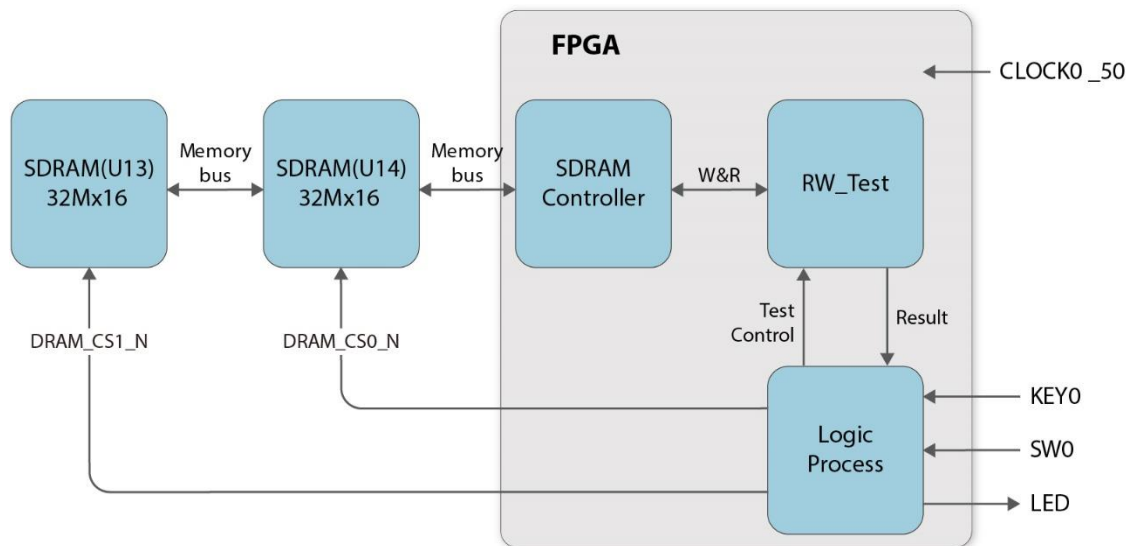


Figure 2-22 Block Diagram of the SDRAM test in Verilog

First, set SW0 to 1 to test U14; set SW0 to 0 to test U13. RW_test module writes the entire memory with a test sequence first before comparing the data read back with the regenerated test sequence, which is same as the data written to the memory. KEY0 triggers test control signals for the SDRAM, and the LEDs will indicate the test result according to [Table 2-2](#).

■ Design Tools

- Quartus Prime 25.1.1 Pro Edition

■ Demonstration Source Code

- Quartus Project directory: SDRAM_Test_RTL
- Bitstream used: golden_top.sof

■ Demonstration Batch File

Demo Batch File Folder: SDRAM_Test_RTL\demo_batch

The demo batch file includes following files:

- Demo Batch File : test.bat
- FPGA Configure File: golden_top.sof

■ Demonstration Setup and Instructions

- Please make sure both Quartus Pro and UBIII driver are installed on the host PC.
- Power on the board.
- Execute the demo batch file “ test.bat” from the directoy SDRAM_Test_RTL\demo_batch.

- set SW0 to 1 to test U14; set SW0 to 0 to test U13.
- Press KEY0 on the board to start the verification process. When KEY0 is pressed, The LEDs [2:0] should be [OFF, ON, ON]. When KEY0 is then released, LED0 and LED1 should start blinking.
- After approximately 8 seconds, LED0 should stop blinking and stay ON to indicate the test is PASS. 错误!未找到引用源。 lists the status of LED indicators.
- If LED1 is not blinking, it means 50MHz clock source is not working.
- Press KEY0 again to repeat the SDRAM test.

Table 2-2 Status of LED Indicators

Name	Description
LED2	Reset
LED1	Blinks
LED0	ON if the test is PASS after releasing KEY0

2.9 RTL_LPDDR4_AXI4_Test

This demonstration performs a memory test function using RTL code on the two LPDDR4 modules of DE25-Nano board. The LPDDR4-A module is controlled by FPGA fabric in this demo. The memory size of each LPDDR4 used in this test is 1GB.

■ Function Block Diagram

Figure 2-23 shows the function block diagram of this demonstration. There are two LPDDR4 controllers IP (LPDDR4A and LPDDR4B) in this project. All of the controllers use 166.666MHz as a reference clock. The test program will write data into LPDDR4, after writing 1GB capacity, it will read the values from LPDDR4 and check whether there is any error.

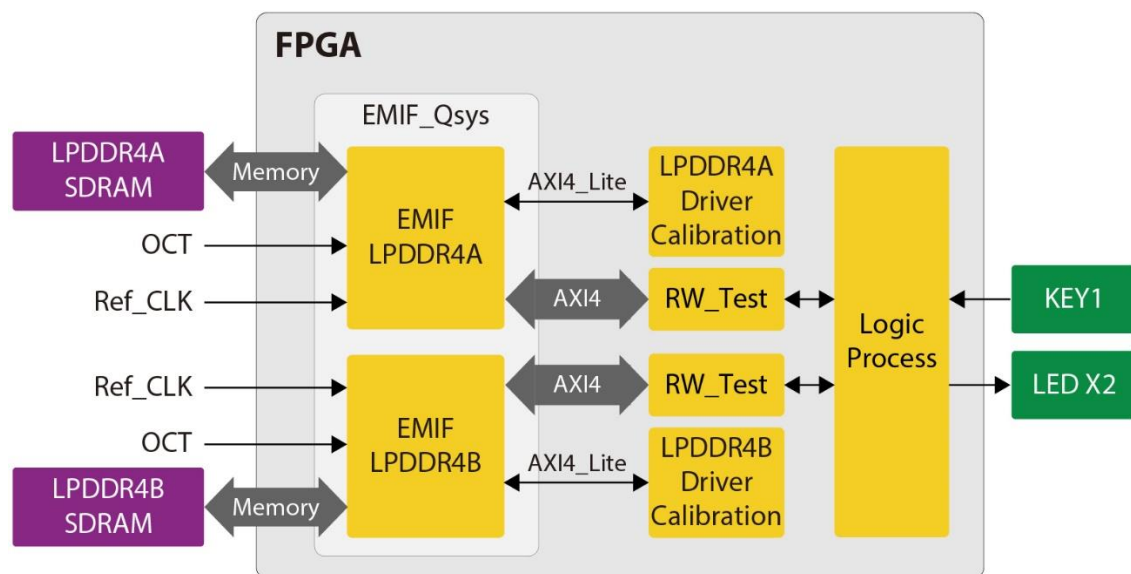


Figure 2-23 Block diagram of the LPDDR4 RTL demonstration

- **EMIF_Qsys:** It includes two LPDDR4 external memory interface (EMIF) IP which will handle all the LPDDR4 protocols and timings. It provides standard AXI4 for high-speed data transmission and AXI4-Lite for register configuration interfaces externally.
- **LPDDR4 Driver Calibration:** LPDDR4 has an extremely high signal rate and therefore must be calibrated at startup to compensate for signal delays and impedance variations on the board. This module acts as an AXI-Lite Master, after the system starts, it automatically gives instructions to the control registers of EMIF_Qsys to trigger and monitor the built-in hardware calibration program. After the calibration is successful, the reset signal will be released to the subsequent test modules.
- **AXI4 Master:** This is the core logic to verify LPDDR4. It acts as an AXI4 Master. After the calibration is completed, it will sequentially execute the test process of write, read and verify. It will generate a large amount of pseudo-random data, write it to LPDDR4 at high speed through the AXI4 bus, then read the data from the same address, and perform real-time comparison internally to ensure the accuracy of the data. After the test is completed, it will output the test_pass and test_complete signals.
- **Reset & Debouncer:** The debouncer module ensures that when users press KEY1, a clean and noise-free pulse can be generated. The hyper_pipe module is responsible for safely synchronizing this reset signal to the high-speed aclk frequency domain, preventing metastability issues and ensuring the reliability of

system reset.

■ Design Tools

- Quartus Prime 25.1.1 Pro Edition

■ Demonstration Source Code

- Project Directory: Demonstration\FPGA\RTL_LPDDR4_AXI4_Test
- Bit Stream: golden_top.sof
- Demonstration Batch File : RTL_LPDDR4_AXI4_Test\demo_batch

The demo batch file includes following files:

- ◆ Batch File: test.bat
- ◆ FPGA Configuration File: golden_top.sof

■ Demonstration Setup

1. Make sure Quartus Prime Pro Edition is installed on the Host PC.
2. Connect the DE25-Nano board to the Host PC via the Type-C USB cable. Install the USB-Blaster III driver if necessary.
3. Power on the DE25-Nano board.
4. Execute the demo batch file “test.bat” under the batch file folder \RTL_LPDDR4_AXI4_Test\demo_batch.
5. Press **KEY1** (see [Figure 2-24](#)) to start LPDDR4 write & loopback verify process. It will take about 1 second to perform the test. While testing, the LED will blink. When LED stop blinking it means the test process is done.
6. The test result will show on LED0 and LED1. The **LED0** represents the test result for the LPDDR4A, the **LED1** represents the test result for the LPDDR4B. If the LED0/LED1 light on, it means the test result is passed. If the LED0/LED1 is off, it means the test result is failed.
7. User can press **KEY1** again to regenerate the test control signals for a repeat test.

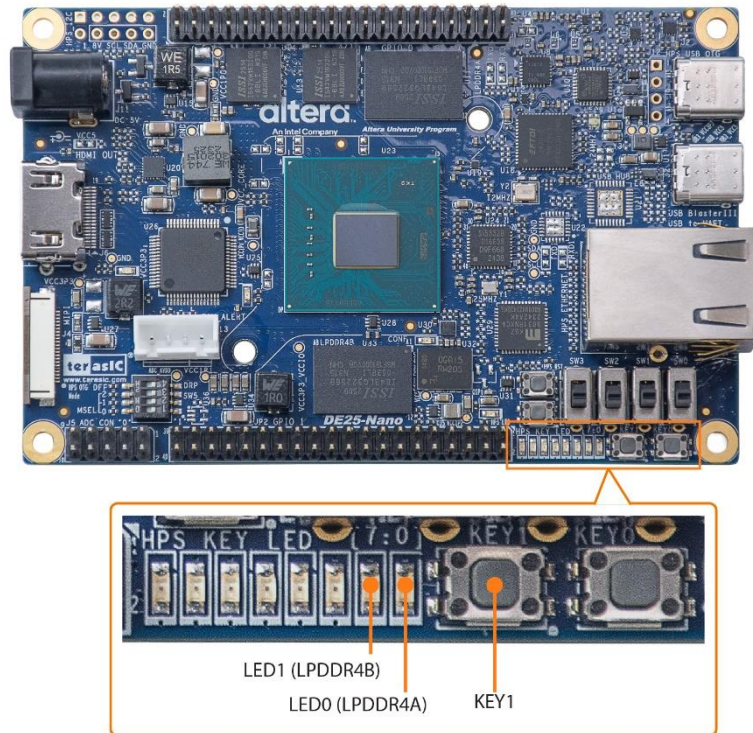


Figure 2-24 Location of the KEY and LED on the DE25-Nano board

2.10 LPDDR4_Test_NiosV

Many applications use a high-performance RAM, such as a LPDDR4 SDRAM, to provide temporary storage. In this demonstration hardware and software designs are provided to illustrate how to perform LPDDR4 memory access in the Platform Designer (formerly Qsys). We describe how the memory controller Agilex External Memory Interfaces is used to access the two LPDDR4 SDRAM on the FPGA board, and how the Nios V processor is used to read and write the LPDDR4 for hardware verification. The LPDDR4 SDRAM controller handles the complex aspects of using the LPDDR4 by initializing the memory devices, managing the LPDDR4 banks, and keeping the devices refreshed at the appropriate intervals.

■ System Block Diagram

Figure 2-25 shows the system block diagram of this demonstration. In the Platform Designer (formerly Qsys), one PLL clock generator (Si5332B) are used. The Si5332B will provide 166.666Mhz clock to the LPDDR4A and LPDDR4B as the reference clock. There are two LPDDR4 Controllers which are used in the demonstrations. Each controller is responsible for one LPDDR4 (LPDDR4A and LPDDR4B). Each LPDDR4 controllers are configured as 1GB LPDDR4 controller. The Nios V processor is used to

perform the memory test. The Nios V software program is running in the On-Chip Memory. A PIO Controller is used to monitor buttons status which is used to trigger starting memory testing.

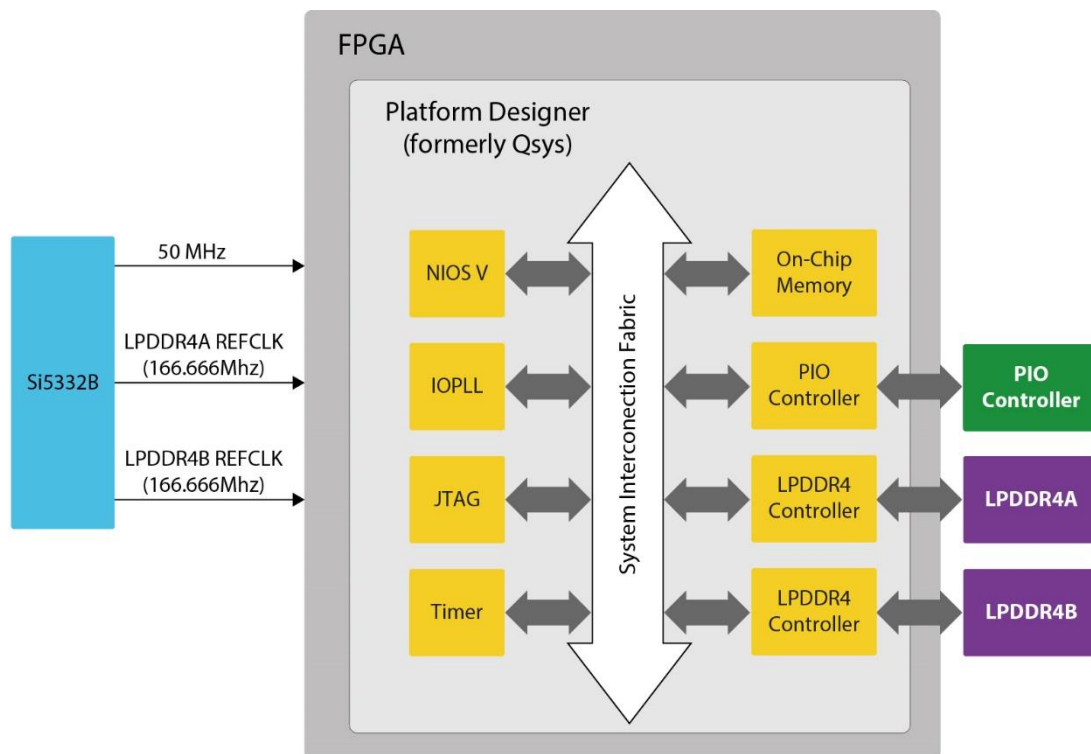


Figure 2-25 Block diagram of the LPDDR4 Basic Demonstration

The system flow is controlled by a Nios V program. First, the Nios V program writes test patterns into the whole 1GB of LPDDR4. Then, it calls Nios V system function, `alt_dache_flush_all()`, to make sure all data has been written to LPDDR4. Finally, it reads data from LPDDR4 for data verification. Maybe the process takes a long time, and there is a quick test. The Nios V program writes a constant pattern into the address line and data line and reads it back for verification. The program will show progress in Nios V terminal when writing/reading data to/from the LPDDR4. When verification process is completed, the result is displayed in the Nios V terminal.

■ Design Tools

- Quartus Prime 25.1.1 Pro Edition

■ Demonstration Source Code

- Quartus Project directory: LPDDR4_Test_NiosV
- Ashling RiscFree IDE: LPDDR4_Test_NiosV/software

■ Nios V Project Compilation

Before you attempt to compile the reference design under Ashling RiscFree IDE, make sure the project is cleaned first by clicking 'Clean' from the 'Project' menu of Ashling RiscFree IDE.

■ Demonstration Batch File

Demo Batch File Folder: LPDDR4_Test_NiosV\demo_batch

The demo batch file includes following files:

- Batch File for USB-Blaster III: test.bat
- FPGA Configure File: golden_top.sof
- Nios V Program: MEM_TEST.elf

■ Demonstration Setup

Please follow below procedures to set up the demonstrations.

1. Make sure Quartus Prime and Ashling RiscFree IDE are installed on your PC.
2. Power on the FPGA board.
3. Use a USB Cable to connect the PC and the FPGA board and install USB Blaster III driver if necessary.
4. Execute the demo batch file "test.bat" under the folder "LPDDR4_Test_NiosV\demo_batch".
5. After the Nios V program is downloaded and executed successfully, a prompt message will be displayed in the Nios V Command Shell.
6. For LPDDR4 test, please input key '0' and press 'Enter' in the Nios V Command Shell terminal as shown in **Figure 2-26**. The program will display progressing and result information.
7. For LPDDR4 quick test, please input key '1' and press 'Enter' in the Nios V Command Shell terminal as shown in **Figure 2-27**. The program will display progressing and result information. Press KEY0~KEY1 of the FPGA board to start LPDDR4 verify process, and press KEY0 for continued test.

```

Select C:\WINDOWS\system32\cmd.exe
[1] LPDDR4x2 Quick Test
Input your chioce:
0
===== LPDDR4x2 Test! Size=A: 1GB, B: 1GB =====
=====
Press any BUTTON on the board to start test [BUTTON-0 for continued test]
=====> LPDDR4x2 Testing, Iteration: 1
== LPDDR4-A Testing...
LPDDR4A address bank: 0GB ~ 1GB:
write...
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
read/verify...
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
LPDDR4A test:Pass, 341 seconds
== LPDDR4-B Testing...
LPDDR4B address bank: 0GB ~ 1GB:

```

Figure 2-26 Progress option [0] LPDDR4x2 Test

```

Select C:\WINDOWS\system32\cmd.exe
Program received signal SIGINT, Interrupt.
0x00000004 in ?? ()
Invalid reset option
Loading section .entry, size 0x20 lma 0x0
Loading section .exceptions, size 0x2f4 lma 0x20
Loading section .text, size 0x24008 lma 0x314
Loading section .rodata, size 0x16c8 lma 0x24320
Loading section .rwddata, size 0x1a00 lma 0x273e8
Start address 0x00001a44, load size 160740
Transfer rate: 181 KB/sec, 22962 bytes/write.
[Inferior 1 (Remote target) detached]
===== Agilex 5 NIOS LPDDR4x2 Program =====
[0] LPDDR4x2 Test
[1] LPDDR4x2 Quick Test
Input your chioce:
1
1
===== LPDDR4x2 Test! Size=A: 1GB, B: 1GB =====
=====
Press any BUTTON on the board to start test [BUTTON-0 for continued test]
=====> LPDDR4x2 Testing, Iteration: 1
== LPDDR4-A Testing...
LPDDR4A address bank: 0GB ~ 1GB:
PASS
LPDDR4A test:Pass, 42 seconds
== LPDDR4-B Testing...
LPDDR4B address bank: 0GB ~ 1GB:

```

Figure 2-27 Progress and Result Information for “LPDDR4x2 Quick Test”

Chapter 3

Examples for HPS SoC

This chapter provides several C-code examples based on the Intel SoC Linux. These examples demonstrate major features connected to HPS interface on DE25-Nano board such as G-sensor. All the associated files can be found in the directory CD/Demonstrations/SoC of the DE25-Nano System CD.

To install the demonstrations on the Host computer: Copy the directory Demonstrations into a local directory of your choice. ARM Toolchain is required for users to compile the c-code project.

3.1 I2C Interfaced G-sensor

This demonstration shows how to control the G-sensor by accessing its registers through the built-in I2C kernel driver in embedded Linux.

■ Function Block Diagram

Figure 3-1 shows the function block diagram of this demonstration. The G-sensor on the FPGA SoC board is connected to the I2C controller in HPS. The G-Sensor I2C 7-bit device address is 0x32. The system I2C bus driver is used to access the register files in the G-sensor. The G-sensor interrupt signal is connected to the PIO controller. This demonstration uses polling method to read the register data.

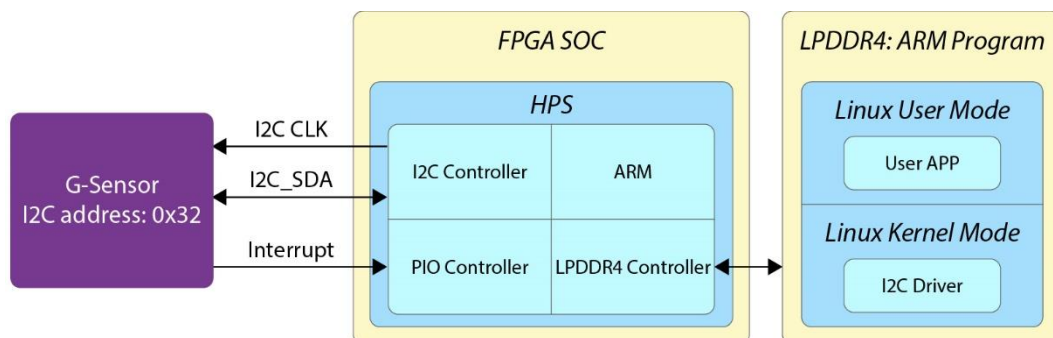


Figure 3-1 Block diagram of the G-sensor demonstration

■ I2C Driver

The procedures to read a register value from G-sensor register files by the existing I2C bus driver in the system are:

1. Open I2C bus driver `"/dev/i2c-0"`: `file = open("/dev/i2c-0", O_RDWR);`
2. Specify G-sensor's I2C address `0x32`: `ioctl(file, I2C_SLAVE, 0x53);`
3. Specify desired register index in g-sensor: `write(file, &Addr8, sizeof(unsigned char));`
4. Read one-byte register value: `read(file, &Data8, sizeof(unsigned char));`

The G-sensor I2C bus is connected to the I2C0 controller, as shown in the **Figure 3-2**. The driver name given is `'/dev/i2c-0'`.

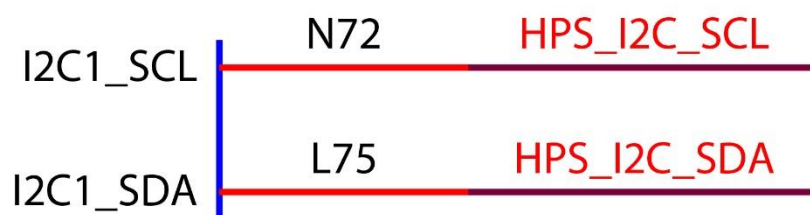


Figure 3-2 Connection of HPS I2C signals

The step 4 above can be changed to the following to write a value into a register.

`write(file, &Data8, sizeof(unsigned char));`

The step 4 above can also be changed to the following to read multiple byte values.

`read(file, &szData8, sizeof(szData8)); // where szData is an array of bytes`

The step 4 above can be changed to the following to write multiple byte values.

`write(file, &szData8, sizeof(szData8)); // where szData is an array of bytes`

■ G-sensor Control

The ST LIS2DW12 provides I2C and SPI interfaces. I2C interface is selected by setting the CS pin to high on the DE25-Nano board.

The LIS2DW12 G-sensor provides user-selectable resolution up to 14-bit $\pm 16g$. The resolution can be configured through the CTRL1(0x20) register. The data format in this demonstration is configured as:

- Full 14-bit resolution mode

- Full Scale $\pm 2g$ range mode

The X/Y/Z data value can be derived from the OUT_X_L(0x28), OUT_X_H(0x29), OUT_Y_L(0x2A), OUT_Y_H(0x2B), OUT_Z_L(0x2C), and OUT_Z_H(0x2D) registers. The OUT_X_L represents the least significant byte and the OUT_X_H represents the most significant byte. It is recommended to perform multiple-byte read of all registers to prevent change in data between sequential registers read. The following statement reads 6 bytes of X, Y, or Z value.

lis2dw12_read_reg(pdev, REG_OUT_X_L, buff, 6)

■ Demonstration Source Code

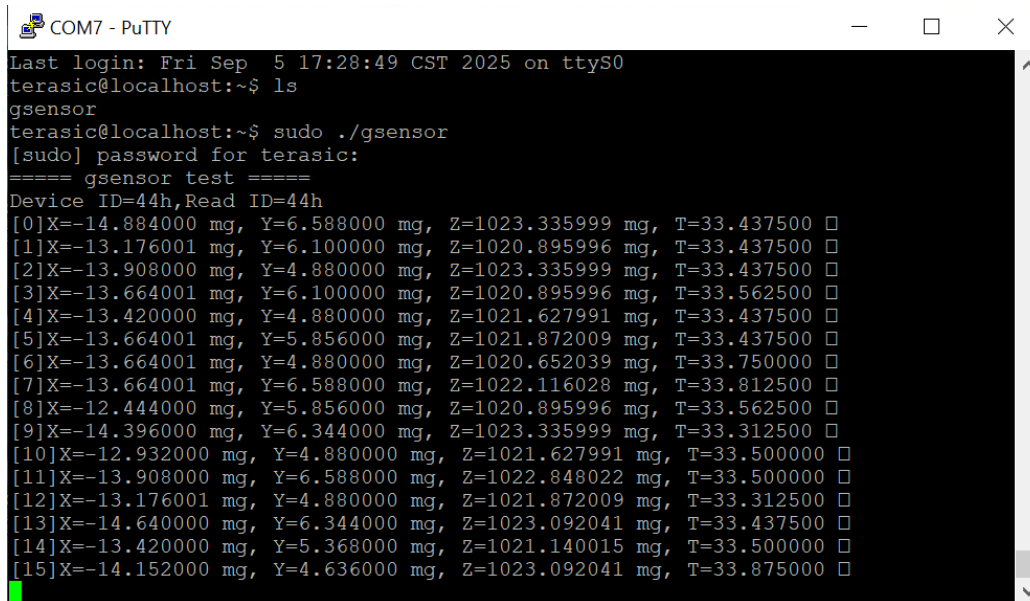
- Build tool: ARM GNU/Linux Toolchain
- Project directory: \Demonstration\SoC\hps_gsensor
- Binary file: gsensor
- Build command: make ('make clean' to remove all temporal files)
- Execute command: ./gsensor [loop count]

■ Demonstration Setup

Please follow below procedures to set up the demonstrations.

1. Connect a USB cable to the Type-C USB connector (J2) on the DE25-Nano board and the host PC.
2. Copy the executable file "gsensor" into the microSD card under the "/home/root" folder in Linux.
3. Insert the booting microSD card into the DE25-Nano board.
4. Power on the DE25-Nano board.
5. Launch PuTTY to establish connection to the UART port of DE25-Nano board. Type "root" to login Linux.
6. Execute "./gsensor" in the UART terminal of PuTTY to start the G-sensor polling.
7. The demo program will show the X, Y, and Z values in the PuTTY, as shown in

Figure 3-3.

A terminal window titled 'COM7 - PuTTY' showing the execution of the 'gsensor' program. The user 'terasic' logs in on 'Fri Sep 5 17:28:49 CST 2025'. They run 'ls' and then 'gsensor'. A password prompt is shown. The program then displays 'Device ID=44h, Read ID=44h' followed by 16 rows of sensor data (X, Y, Z, T) in mg and °C, each followed by a square symbol. The data values vary slightly between rows, representing different sensor locations or orientations.

```
COM7 - PuTTY
Last login: Fri Sep 5 17:28:49 CST 2025 on ttyS0
terasic@localhost:~$ ls
gsensor
terasic@localhost:~$ sudo ./gsensor
[sudo] password for terasic:
===== gsensor test =====
Device ID=44h, Read ID=44h
[0]X=-14.884000 mg, Y=6.588000 mg, Z=1023.335999 mg, T=33.437500 °
[1]X=-13.176001 mg, Y=6.100000 mg, Z=1020.895996 mg, T=33.437500 °
[2]X=-13.908000 mg, Y=4.880000 mg, Z=1023.335999 mg, T=33.437500 °
[3]X=-13.664001 mg, Y=6.100000 mg, Z=1020.895996 mg, T=33.562500 °
[4]X=-13.420000 mg, Y=4.880000 mg, Z=1021.627991 mg, T=33.437500 °
[5]X=-13.664001 mg, Y=5.856000 mg, Z=1021.872009 mg, T=33.437500 °
[6]X=-13.664001 mg, Y=4.880000 mg, Z=1020.652039 mg, T=33.750000 °
[7]X=-13.664001 mg, Y=6.588000 mg, Z=1022.116028 mg, T=33.812500 °
[8]X=-12.444000 mg, Y=5.856000 mg, Z=1020.895996 mg, T=33.562500 °
[9]X=-14.396000 mg, Y=6.344000 mg, Z=1023.335999 mg, T=33.312500 °
[10]X=-12.932000 mg, Y=4.880000 mg, Z=1021.627991 mg, T=33.500000 °
[11]X=-13.908000 mg, Y=6.588000 mg, Z=1022.848022 mg, T=33.500000 °
[12]X=-13.176001 mg, Y=4.880000 mg, Z=1021.872009 mg, T=33.312500 °
[13]X=-14.640000 mg, Y=6.344000 mg, Z=1023.092041 mg, T=33.437500 °
[14]X=-13.420000 mg, Y=5.368000 mg, Z=1021.140015 mg, T=33.500000 °
[15]X=-14.152000 mg, Y=4.636000 mg, Z=1023.092041 mg, T=33.875000 °
```

Figure 3-3 Terminal output of the G-sensor demonstration

8. Press "CTRL + C" to terminate the program.

3.2 Build C/C++ Project

This section describes how to recompile the above C/C++ project included in the System CD.

First, user need to download and install ARM GNU/Linux tool chain:

1. Login Linux or WSL on Windows.
2. Type "cd ~"
3. Type
"wget https://developer.arm.com/-/media/Files/downloads/gnu/11.2-2022.02/binrel/gcc-arm-11.2-2022.02-x86_64-aarch64-none-linux-gnu.tar.xz"
4. Type "tar xf gcc-arm-11.2-2022.02-x86_64-aarch64-none-linux-gnu.tar.xz"
5. Type "export PATH=`pwd`/gcc-arm-11.2-2022.02-x86_64-aarch64-none-linux-gnu/bin:\$PATH"
6. Type "export CROSS_COMPILE=aarch64-none-linux-gnu-"
7. Type "git clone <https://github.com/altera-opensource/intel-socfpga-hwlib>" to download HPS hardware library.

Here is the procedure to compile the example projects in System CD:

1. Login Linux or WSL on Windows.
2. Type "cd ~"
3. Type "export PATH=`pwd`/gcc-arm-11.2-2022.02-x86_64-aarch64-none-linux-gnu/bin:\$PATH"

4. Type “export CROSS_COMPILE=aarch64-none-linux-gnu-”
5. Copy the CD Demo project into the Linux System and go to the project folder.
6. Type “make” to build project as shown in **Figure 3-4**.

```
terasic@DESKTOP-KPV9G48:~$ cd gsensor
terasic@DESKTOP-KPV9G48:~/gsensor$ make clean
rm -f gsensor *.a *.o *~
terasic@DESKTOP-KPV9G48:~/gsensor$ make
aarch64-none-linux-gnu-gcc -g -Wall -Werror -c main.c -o main.o
aarch64-none-linux-gnu-gcc -g -Wall -Werror -c lis2dw12_test.c -o lis2dw12_test.o
aarch64-none-linux-gnu-gcc -g -Wall -Werror -c main.o lis2dw12_test.o lis2dw12_api.c -o gsensor
terasic@DESKTOP-KPV9G48:~/gsensor$ ls
Makefile  gsensor  lis2dw12_api.c  lis2dw12_api.h  lis2dw12_regs.h  lis2dw12_test.c  lis2dw12_test.h  lis2dw12_test.o  lis2dw12_types.h  main.c  main.o
terasic@DESKTOP-KPV9G48:~/gsensor$
```

Figure 3-4 Build C/C++ Project

Chapter 4

Program the QSPI Flash

There is a flash on the board for placing the bitstream files designed by the user. After the FPGA is powered up, the bit-stream file can be loaded into the FPGA from the flash.

This chapter will introduce how to program the factory default code into the flash on the board. It will also introduce how to convert the your own .sof file into a .jic file and program it into the flash.

■ MSEL switch setting

Before program the flash on the board, please make sure the MSEL pin of the FPGA is set to AS mode. So that user can program the flash via FPGA and JTAG interface. Please refer to the **Figure 4-1** to set the SW5 to AS mode: **MSEL[2:0] = "001"**.

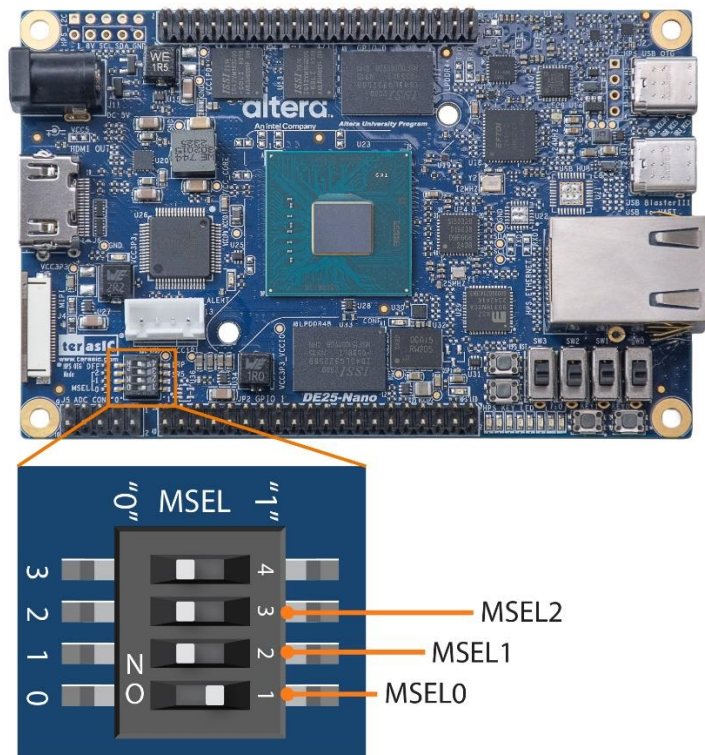


Figure 4-1 Position of slide switches SW5 for Configuration Mode

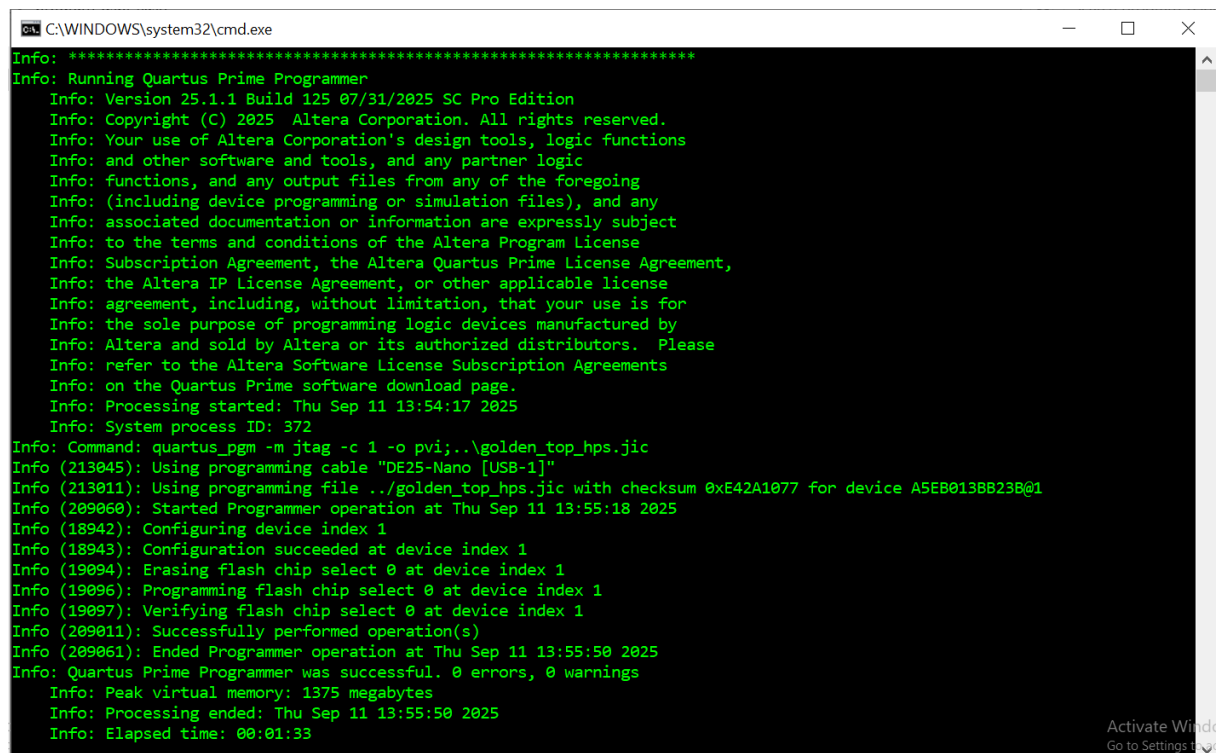
■ Program the Factory image file

The flash on the Board had programmed the factory file when shipped. After power on, user can check if the user LED7 is flashing, and after 10 seconds of booting. If not, please refer to following steps to re-program the flash with the factory code.

- Connect the Type-C USB cable to USB blaster III connector of the board.
- Make sure MSEL switch is set to 001 in AS mode.
- Copy the factory code to host from the path:

System CD\Demonstration\SoC_FPGA\GHRD\output_files\program_qspi_flash\

- Execute “flash_program.bat” as shown in **Figure 4-2** to erase and program the flash.



```
Info: *****
Info: Running Quartus Prime Programmer
Info: Version 25.1.1 Build 125 07/31/2025 SC Pro Edition
Info: Copyright (C) 2025 Altera Corporation. All rights reserved.
Info: Your use of Altera Corporation's design tools, logic functions
Info: and other software and tools, and any partner logic
Info: functions, and any output files from any of the foregoing
Info: (including device programming or simulation files), and any
Info: associated documentation or information are expressly subject
Info: to the terms and conditions of the Altera Program License
Info: Subscription Agreement, the Altera Quartus Prime License Agreement,
Info: the Altera IP License Agreement, or other applicable license
Info: agreement, including, without limitation, that your use is for
Info: the sole purpose of programming logic devices manufactured by
Info: Altera and sold by Altera or its authorized distributors. Please
Info: refer to the Altera Software License Subscription Agreements
Info: on the Quartus Prime software download page.
Info: Processing started: Thu Sep 11 13:54:17 2025
Info: System process ID: 372
Info: Command: quartus_pgm -m jtag -c 1 -o pvi;..\golden_top_hps.jic
Info (213045): Using programming cable "DE25-Nano [USB-1]"
Info (213011): Using programming file ../golden_top_hps.jic with checksum 0xE42A1077 for device A5EB013BB23B@1
Info (209060): Started Programmer operation at Thu Sep 11 13:55:18 2025
Info (18942): Configuring device index 1
Info (18943): Configuration succeeded at device index 1
Info (19094): Erasing flash chip select 0 at device index 1
Info (19096): Programming flash chip select 0 at device index 1
Info (19097): Verifying flash chip select 0 at device index 1
Info (209011): Successfully performed operation(s)
Info (209061): Ended Programmer operation at Thu Sep 11 13:55:50 2025
Info: Quartus Prime Programmer was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 1375 megabytes
Info: Processing ended: Thu Sep 11 13:55:50 2025
Info: Elapsed time: 00:01:33
```

Figure 4-2 Program the factory code to the board

■ Programming User's own design into Flash

If users want to program their own bitstream file into the flash of the board, Terasic had provided some batch files that help the user to convert the .sof file into a .jic file and then program it into the flash.

Users can obtain these tools from the following path:

System CD/Demonstration/FPGA/HDMI_ASx4/demo_batch

There are some files under this path, and the detailed functions are shown in **Figure 4-3** and **Table 4-1**:

‣ HDMI_ASx4 ‣ demo_batch







Name	Date modified	Type
 convert.bat	9/8/2025 2:23 PM	Windows Batch File
 flash.cdf	9/8/2025 2:23 PM	CDF File
 flash_program.bat	9/8/2025 2:23 PM	Windows Batch File
 golden_top.jic	9/8/2025 2:23 PM	JIC File
 golden_top.sof	9/8/2025 2:23 PM	SOF File
 test.bat	9/8/2025 2:23 PM	Windows Batch File

Figure 4-3 Batch files for programming flash

Table 4-1 File descriptions of the flash programming tool

File Name	Functions
convert.bat	This batch file can convert the golden_top.sof to golden_top.jic
flash.cdf	The flash programming configuration file
flash_program.bat	This batch file will program the .jic file into flash
golden_top.jic	The file that can be programmed into flash
golden_top.sof	The bitstream file of User's design
test.bat	This batch file can program the golden_top.sof into the batch file of FPGA

The following are the steps on how to use this tool to program the user's own bitstream file into flash.

1. Copy this demo_batch folder to host PC
2. Rename the user's own .sof file to *golden_top.sof* and overwrite it into the demo_batch folder to replace the original *golden_top.sof*.
3. Execute **convert.bat** to convert *golden_top.sof* to *golden_top.jic*.
4. Confirm that the MSEL switch on the board is set to AS Mode.
5. Connect the Type-C USB cable to USB blaster III connector of the board.
6. Execute **flash_program.bat** to program the *golden_top.jic* file into flash.
7. Turn the power off and on of the board again to check whether your design is running on the FPGA.

Chapter 5

Additional Information

5.1 Getting Help

Here are the addresses where you can get help if you encounter problems:

■ Terasic Technologies

No.80, Fenggong Rd., Hukou Township, Hsinchu County 303035. Taiwan

Email: support@terasic.com

Web: www.terasic.com

DE25-Nano Development Kit Web: <http://de25-nano.terasic.com/>

■ Revision History

Date	Version	Changes
2025.09	First publication	