# ADC Demo Design on CertusPro-NX Versa Board

# User Guide

FPGA-EB-02061-1.2

August 2025

## Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

## Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

# Contents

# Figures



# Tables

# Abbreviations in This Document

A list of abbreviations used in this document.

| Abbreviation | Definition |
|---|---|
| ADC | Analog to Digital Converter |
| BCD | Binary Coded Decimal |
| DIP | Dual Inline Package |
| DRC | Design Rule Checking |
| DTR | Digital Temperature Readout |
| EOC | End of Conversion |
| FSM | Finite State Machine |
| ILA | Internal Logic Analyzer |
| LLC | Lower Left Corner |
| LRC | Lower Right Corner |
| LSE | Lattice Synthesis Engine |
| MUX | Multiplexer |
| OSC | Oscillator |
| POT | Potentiometer |
| PMOD | Peripheral Module |
| PLL | Phase-Locked Loop |
| RTL | Register Transfer Level |
| SAR | Successive Approximation Register |
| ULC | Upper Left Corner |
| URC | Upper Right Corner |

# 1. Introduction

This demo design is one of hundreds of examples that could be implemented using the Analog to Digital Conversion (ADC) feature of Lattice Nexus family devices from Lattice. This demo design can be copied, modified, and included into your designs to simplify the ADC aspect. Alternatively, it can provide a reference from which other designs can be created. This demo design is targeted for the CertusPro™-NX Versa Evaluation Board.

The demo can control the ADC, convert the binary output and show the result on the 7-segment LED display D60. A state machine controls the ADC regarding power up calibration, multiplexer (MUX) setting, sampling the input, converting the analog signal, and storing the 12-bit outputs. The 12-bit binary output is converted to a voltage or temperature value based on the ADC MUX channel.

This demo shows how to process the ADC output when the analog input is from the internal digital temperature readout (DTR), internally divided down supply pins, or external voltage source provided by the on-board potentiometer (POT), R143.

Figure 2.1 shows the powered-on board with default jumpers. The ADC demo runs on the CertusPro-NX Versa Evaluation Board with the DIP switch selecting the adjustable POT as the input. The POT is set to 1.23 V.

# 2. Running the Demo

This demo is built using the following versions of Lattice software, IPs, and hardware components:

- Lattice Radiant Software and Programmer Ver. 2024.1
- ADC IP Ver. 1.5.0
- PLL IP Ver. 1.9.0
- Oscillator IP version 1.4.0
- CertusPro-NX Versa Board Rev. B

The ADC Demo Lattice Radiant project is archived on the ADC Demo Design web page. The contents of this archive are summarized below, starting at the top level. Unlike earlier versions, this demo uses System Verilog and must be synthesized using Synplify® in the Lattice Radiant software rather than Lattice Synthesis Engine (LSE).

- certuspro_nx_versa_v1.rdf – Lattice Radiant Project file
- source/impl_1 folder
  - adc_certuspro_nx_versa_v1.sv – Top level module of demo Verilog file
  - adc_temp.sv – Binary to Temperature conversion Verilog file
  - ADC_binary2volts.v – Binary to Voltage conversion Verilog file
  - hex2ssd.v – Hex to 7-Segment Verilog file
  - adc_certuspro_nx_versa.pdc – Contains pin mapping information for the design. Clock and timing information is not here as it is part of the oscillator and PLL IP.
- adc_pair folder – Contains generated files from the IP Wizard for ADC.
- pll_adc folder – Contains generated files from the IP Wizard for PLL. Note that the ADC needs the bottom right PLL CLKOS4.
- osc_50 – Contains generated files from the PLL IP Wizard for the 50 MHz on-die oscillator.

The following jumpers need to be installed on the CertusPro-NX Versa Board for this demo to operate in hardware.

- ADC_VREF0 – J28 pins 1-2, gets 1.8 V reference from reference IC on board
- ADC_VREF1 – J29 pins 1-2, gets 1.8 V reference from reference IC on board
- ADC_DP0 – J25 pins 1-2, dedicated ADC0 positive input to ADC_POT R143
- ADC_DN0 – J26 pins 1-2, dedicated ADC0 negative input to GND
- Bank 2 VCCIO – J57 pins 1-2, 3.3 V
- Bank 0 VCCIO – J58 pins 1-2, 3.3 V

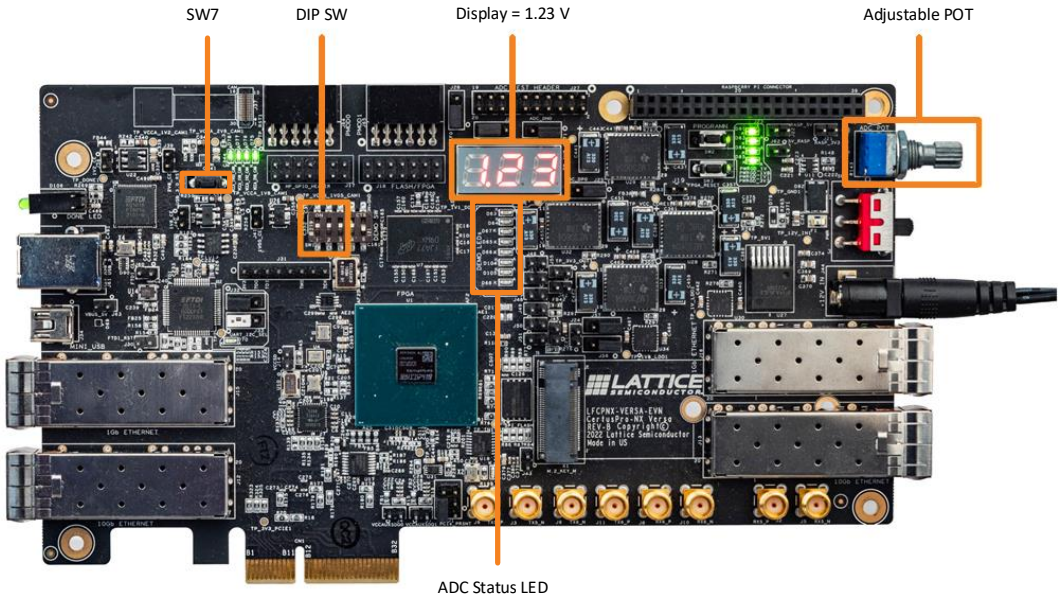


**Figure 2.1. CertusPro-NX Versa Board Powered on with Default Jumpers**

# 3. Testing and Debugging

## 3.1. Basic Testing

At power on reset, the 7-segment LED display shows the ADC0 input from pins DP0 and DN0 that are connected to the potentiometer. Vary the potentiometer to see voltages from 0 to 1.8 V on the 7-segment LED. All the LED indicators are off, as shown in the Status LEDs section. To see other inputs, use Reveal.

## 3.2. Using Reveal to Change the ADC Input

The ADC mux input is controlled by the Reveal Analyzer as a user control register. Connect a USB cable to the board to use Reveal. To open this tool, open the design using the Lattice Radiant software and open the debug file rvl_1_1_1.rva. Reveal has two sections, the logic analyzer and the controller. The logic analyzer shows waveforms inside the FPGA. The controller can implement virtual switches that change the status inside the FPGA, as well as virtual LEDs for static outputs from the FPGA.

When you open Reveal, do the following in the given order, as shown in Figure 3.1.

1. Using the drop-down menu on the top right, choose **top_Controller**.
2. Select an intermediate polling speed and start polling the virtual LEDs.
3. Select **Direct Mode** for virtual switches.

This should provide control of the ADC inputs.

- To see the ADC1 input DP1 and DN1 on the 7-segment, select **ADC_EXT_SEL0B1**. Then, you can see the GUI shown in Figure 3.1. LED 0 on the topmost is on and ADC1 is being displayed. ADC1 input comes from J27 pins 9 and 11.
- To see the voltage on the internal VCCM supply, select **SEL_VCCM**. Then, board LED1, which is second from top turns on. The 7-segment shows the VCCM voltage.
- To see VCCIO1 voltage, use **SEL_VCCIO1**. Note that LEDs on the board and virtual LEDs turn on for both VCCIO1 and **SEL_ADC1**, as ADC1 is used to read the VCCIO1 voltage.
- In this way, all voltages can be read on the 7-segment LED.
- To see the temperature, select **SEL_DTR**. This shows the temperature in Kelvin on the 7-segment LED. If you select **SEL_TEMP_C** or **SEL_TEMP_F**, you can see the temperature in Celsius or Fahrenheit. The LED pattern is explained in the Status LEDs section.
- Other buttons of the Reveal GUI are explained in respective sections.

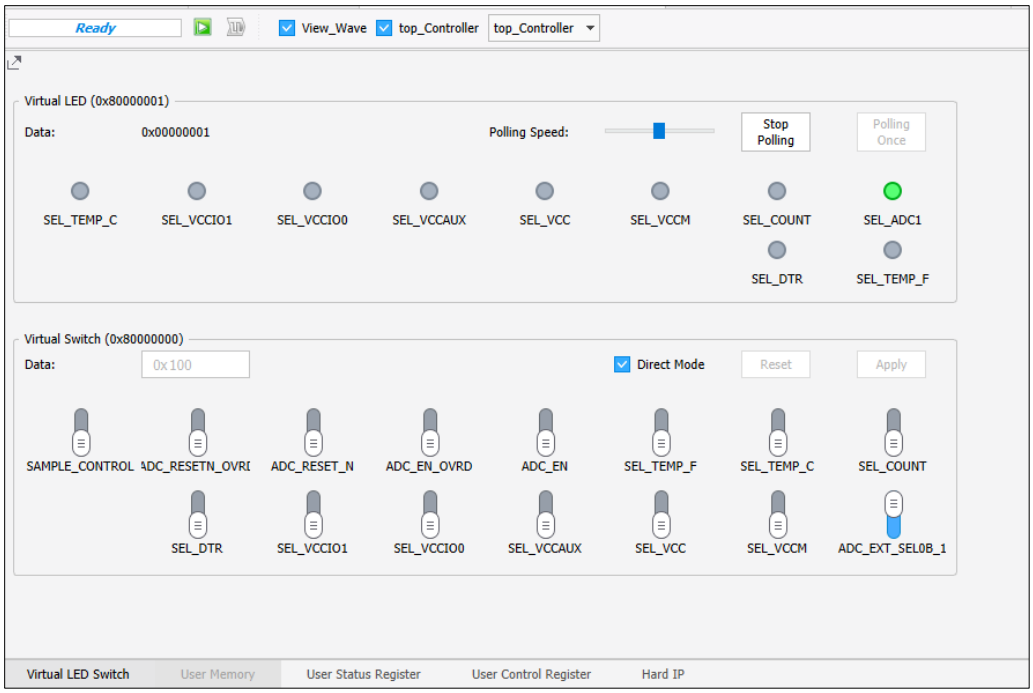


**Figure 3.1. Change ADC Selection in Reveal Controller, with ADC1 DP1 and DN1 Inputs Selected**

## 3.3. Conversion Stop

By pressing SW7, you give an adc_convstop_i signal to the ADC that stops the conversion. The ADC finite state machine (FSM) has been designed to take another sample as soon as the conversion stops so nothing can be visible on the user display. However, the conversion stop can be observed using the Reveal Logic Analyzer, which is shown in Figure 3.2. Notice that when adc_convstop becomes 1 at the cursor, the adc_eoc pulse does not occur but adc_cog becomes 0 to end the conversion. The FSM restarts by taking a new sample with adc_soc after that.

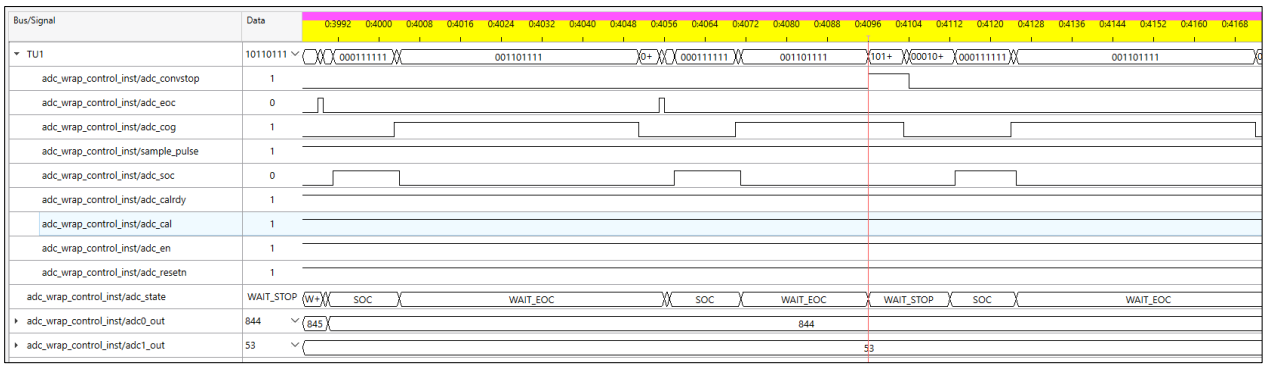


**Figure 3.2. Sample of Conversion Stop at the Cursor – Absence of End of Conversion (EOC)**

## 3.4. Board Testing and Debugging Features

### 3.4.1. Status LEDs

The demo LEDs from LED0 to LED7 under the 7-segment display can be used to debug the board if something does not work. LED0 is at the top, and LED7 is at the bottom.

| LED | Function |
|---|---|
| 0 | If LED0 is on, it indicates that ADC1 is displayed on the 7-segment. If off, it indicates ADC0 is displayed. The inputs the ADCs are sampling can be determined from other LEDs.<br>If no other LEDs are on while LED0 is off, then the 7-segment displays the DP0DN0 voltage from the potentiometer. This is true if the header on J25 is from pin 1 to pin 2 and header on J26 is from pin 1 to pin 2. See header J25 and J26 on sheet 11 of the schematic in CertusPro-NX Versa Board Evaluation Board User Guide (FPGA-EB-02053).<br>If no other LEDs are on while LED0 is on, then the 7-segment displays DP1DN1 voltage from J27 header pins 9 and 11. |
| 1 | If LED1 is on, the 7-segment displays the voltage of the internal supply VCCM. |
| 2 | If LED2 is on, the 7-segment displays the voltage of the internal supply VCC. |
| 3 | If LED3 is on, the 7-segment displays the voltage of the internal supply VCCAUX. |
| 4 | If LED4 is on, the 7-segment displays the voltage of internal supply VCCIO0. LED0 is also on as ADC1 is used to measure this VCCIO0. |
| 5 | If LED5 is on, the 7-segment displays the voltage of internal supply VCCIO1. LED0 is also on as ADC1 is used to measure this VCCIO1. |
| 6 | If LED6 is on, temperature is displayed on the 7-segment. LED1 is on as DTR is read by ADC1. If LED7 is off, then the temperature value is in Kelvin. |
| 7 | If LED7 is off and LED6 is on, the displayed temperature value is in Kelvin.<br>If LED7 is on, the temperature value is in Celsius.<br>If LED7 is flashing, the temperature value is in Fahrenheit. |

### 3.4.2. LED 7-Segment Display

The LED 7-segment display is used to display voltage and temperature. It is a common cathode display and is driven from the FPGA.

### 3.4.3. External Scope Signals

The PMOD0 J64 and PMOD1 J65 connectors are used as internal FPGA signals to headers for debug.

**Table 3.1. PMOD0 Header J64**

| Pin Name | Signal | Description |
|---|---|---|
| 1 | adc_resetn | ADC reset pin. This is different from the system reset. |
| 2 | adc_en | ADC enable. It turns on the internal ADC clock gating. |
| 3 | adc_cal | ADC calibration starts. |
| 4 | adc_calrdy | ADC calibration is done. |
| 5 | gnd | Board ground. |
| 6 | VCCIO2 | VCCIO2 is expected to be 3.3 V. |
| 7 | adc_soc | Start of conversion for both ADC0 and ADC1. |
| 8 | adc_cog | Ongoing conversion for both ADC0 and ADC1. |
| 9 | adc_eoc | End of conversion for both ADC0 and ADC1. |
| 10 | adc_convstop | Stops conversion on both ADCs. |

**Table 3.2. PMOD1 Header J65**

| Pin Name | Signal | Description |
|---|---|---|
| 1 | fab_clk | Fabric clock. |
| 2 | adc_clk | ADC clock. |
| 3 | sample_pulse | Pulses when the SOC signal goes low. If it is constantly 1, it indicates that the SOC signal goes 0 as soon as COG becomes 1. |
| 4 | strobe_1Hz | 1 Hz strobe signal. It indicates the clock is working. |
| 5 | gnd | Board ground. |
| 6 | VCCIO2 | VCCIO2 is expected to be 3.3 V. |
| 7 | strobe_7seg | Strobe signal for the 7-segment display. It is about 610 Hz. |

| Pin Name | Signal | Description |
|---|---|---|
| 8 | sel_adc0b_1 | 0-ADC0 is being displayed.<br>1-ADC1 is being displayed. |
| 9 | 0 | Always tied to logic 0. |
| 10 | ila_trigger_out | Reveal LA is triggered. |

### 3.4.4. Reveal Controls

The following sections are visible if the top controller is selected in Reveal, as shown in Figure 3.3.

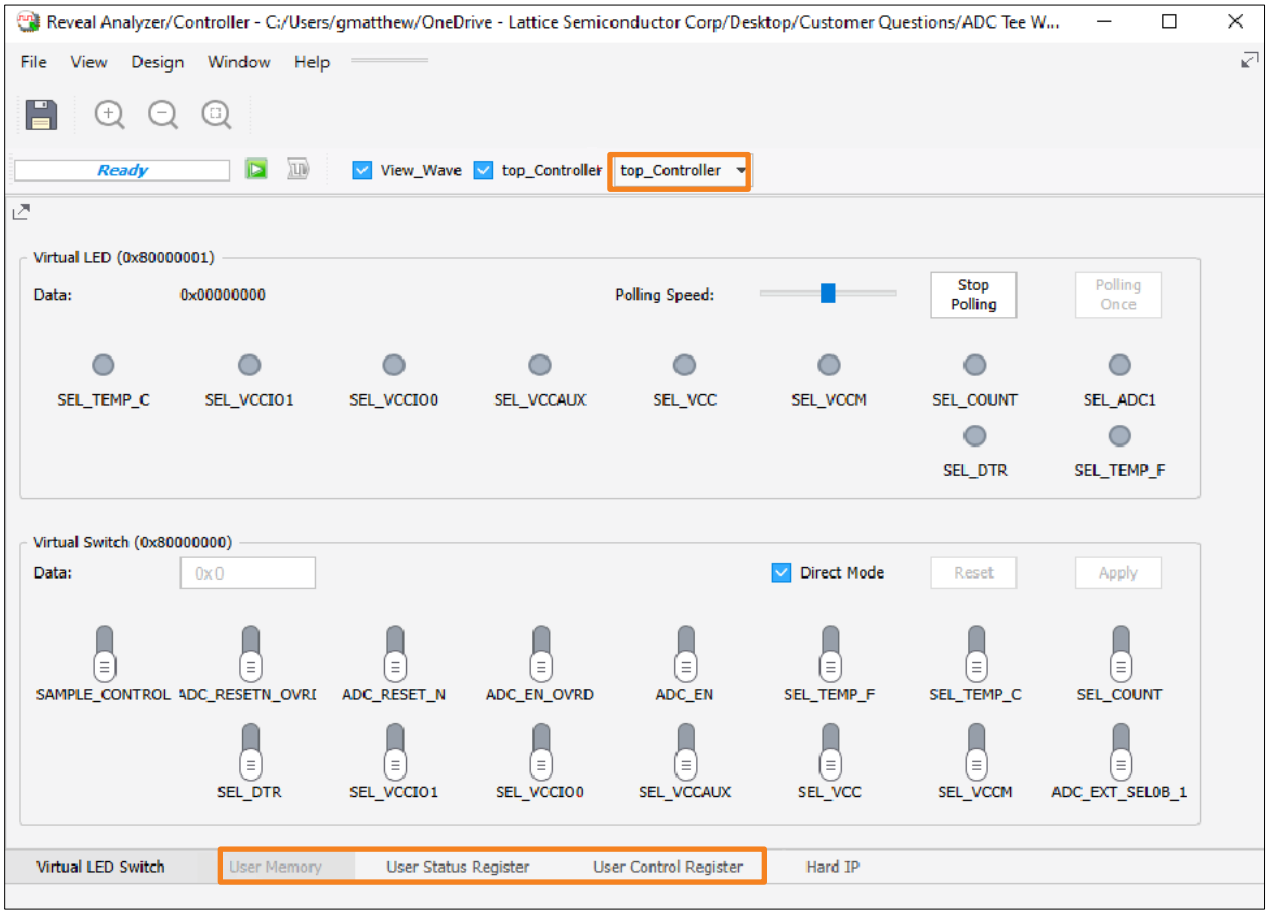


**Figure 3.3. Top Controller Selected in the Reveal GUI**

The top controller has virtual switches that replace DIP switches on the board. These virtual switches are summarized in Table 3.3.

**Table 3.3. Reveal Switches**

| Switch | Function |
|---|---|
| SAMPLE_CONTROL | If 1, takes one sample at the frequency of fab_clk/{SAMPLE_H,SAMPLE_L} in the user control register of Reveal.<br>If 0, samples are taken as soon as EOC pulse is done. |
| ADC_RESETN_OVRD | If 0, ADC comes out of reset as soon as the system comes out of reset as controlled by ADC wrap FSM.<br>If 1, ADC reset is controlled by the ADC_RESET_N switch. |
| ADC_RESET_N | Value of adc_resetn if ADC_RESETN_OVRD =1. |
| ADC_EN_OVRD | If 0, ADC gets enabled as soon as ADC comes out of reset as controlled by ADC wrap FSM.<br>If 1, ADC reset is controlled by the ADC_EN_N switch. |

| Switch | Function |
|---|---|
| ADC_EN | Value of adc_en if ADC_EN_OVRD =1. |
| SEL_TEMP_F | If SEL_DTR=1, makes temperature display in Fahrenheit.<br>If SEL_DTR=0, ignored. |
| SEL_TEMP_C | If SEL_DTR=1, makes temperature display in Celsius.<br>If SEL_DTR=0, ignored. |
| SEL_COUNT | If 1, displays counter on LEDs. The counter runs from 0.00 to 9.99. |
| SEL_DTR | Displays temperature on 7-segment.<br>If SEL_TEMP_F and SEL_TEMP_C are both 0, then temperature is in Kelvin. |
| SEL_VCCIO1 | Internal voltage of VCCIO1 supply. |
| SEL_VCCIO0 | Internal voltage of VCCIO0 supply. |
| SEL_VCCAUX | Internal voltage of VCCAUX supply. |
| SEL_VCC | Internal voltage of VCC supply. |
| SEL_VCCM | Internal voltage of VCCM supply. |
| ADC_EXT_SEL0B_1 | If 1, ADC1 input from ADC header is displayed.<br>If 0, ADC0 output from POT is displayed. |

**Note:** It is possible to select contradictory inputs, for example, selecting SEL_VCCIO0 and SEL_VCCIO1. The Behavior of the switches in this case is undefined.

There are virtual LEDs that are used to display the internal status of the ADC logic (Table 3.4).

**Table 3.4. Switches for Displaying Internal Status of the ADC Logic**

| LED | Meaning |
|---|---|
| SEL_TEMP_C | Indicates temperature is in Celsius. If it is off and SEL_TEMP_F is also off and SEL_DTR is on, the temperature is in Kelvin. |
| SEL_TEMP_F | Indicates temperature is in Fahrenheit. If it is off and SEL_TEMP_C is also off and SEL_DTR is on, the temperature is in Kelvin. |
| SEL_VCCIO1 | Indicates display is showing VCCIO1 voltage. When it is only, SEL_ADC1 is also on as ADC1 is used to measure VCCIO1. |
| SEL_VCCIO0 | Indicates display is showing VCCIO0 voltage. When it is only, SEL_ADC1 is also on as ADC1 is used to measure VCCIO0. |
| SEL_VCCAUX | Indicates display is showing VCCAUX voltage. |
| SEL_VCC | VCC voltage is being displayed. |
| SEL_VCCM | VCCM voltage is being displayed. |
| SEL_COUNT | The counter is being displayed. |
| SEL_ADC1 | ADC1 output is being displayed. |
| SEL_DTR | The temperature is being displayed. Unit depends on SEL_TEMP_C and SEL_TEMP_F. If both are off, the temperature is in Kelvin. |

The user status register can be selected using tabs shown in Figure 3.3. It shows the number of fab_clk cycles from cal_start to calrdy. It consists of CAL_RUN_TIME_H, which is bits 15:8, and CAL_RUN_TIME_L, which is bits 7:0. It is recommended to read the register in hex, as shown in Figure 3.4.

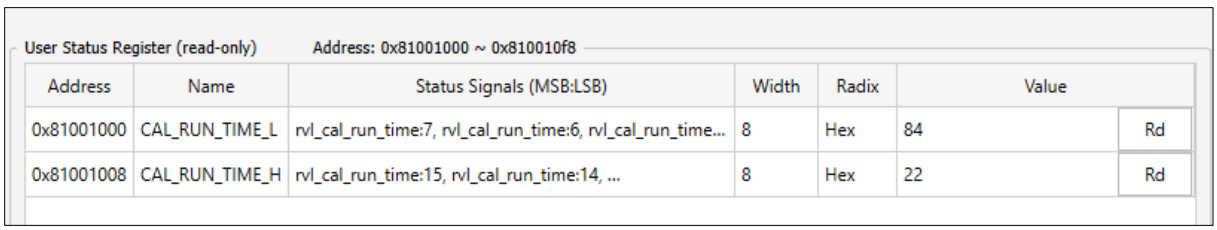


**Figure 3.4. User Status Register Showing CAL Run Time**

The user control register in Figure 3.5 shows the sample timer. If sample enable is 1, then a sample is taken once every {SAMPEL_H, SAMPLE_L} fab_clk cycles. This gives a consistent sampling rate, but the sample rate should be chosen so that each sample pulse appears only when the SOC signal is high and the COG signal is high. Use Reveal Logic Analyzer to analyze the waveform and fix the sample rate.

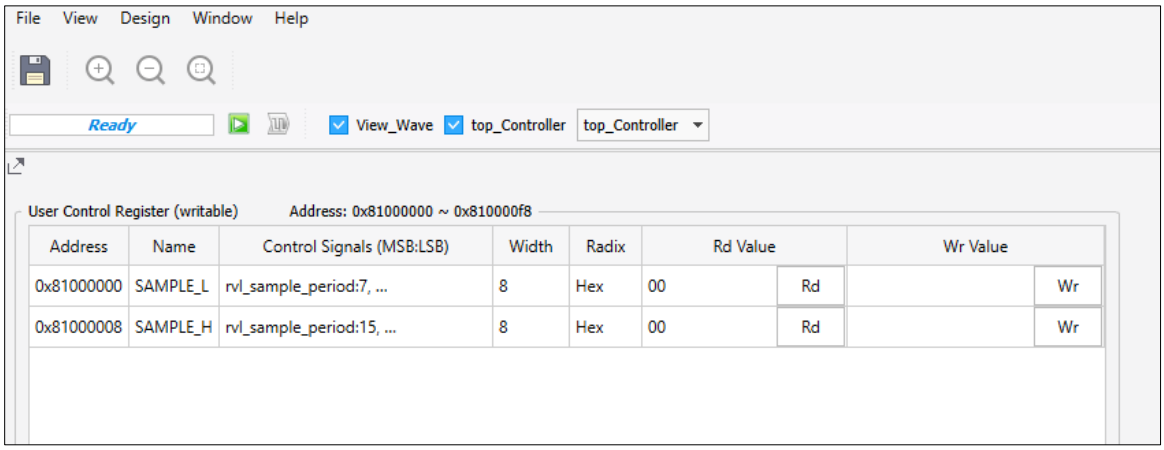


**Figure 3.5. User Control Register**

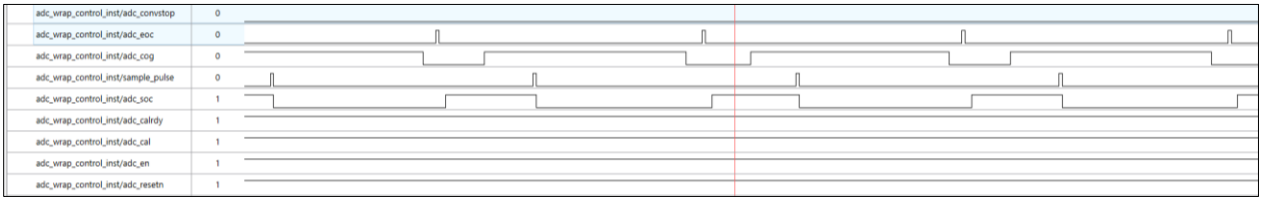


**Figure 3.6. Correct Way to Set Sample Pulse**

**Note:** The pulse comes only when both SOC and COG are 1.

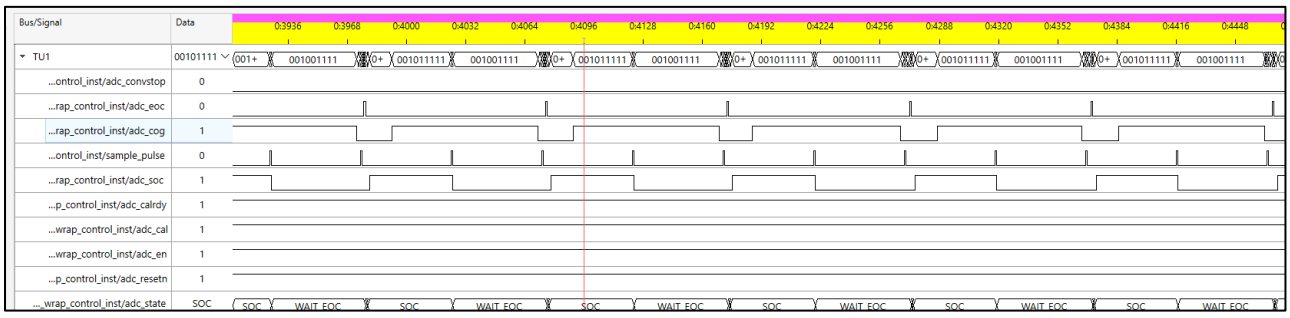


**Figure 3.7. Wrong Way to Set Sample Pulse**

## 3.4.5. Board Level Analog Input

The Input to ADC0 comes from the potentiometer on the board R143, which gives an output voltage from 1.8 V to 0 V to the ADC DP0 and DN0 if the jumper connects pins of J25. The CertusPro-NX board is shipped in this way. ADC1 input comes from pins 9 and 11 of J27. See the schematic shown in Figure 3.8 excerpted from CertusPro-NX Versa Board Evaluation Board User Guide (FPGA-EB-02053).

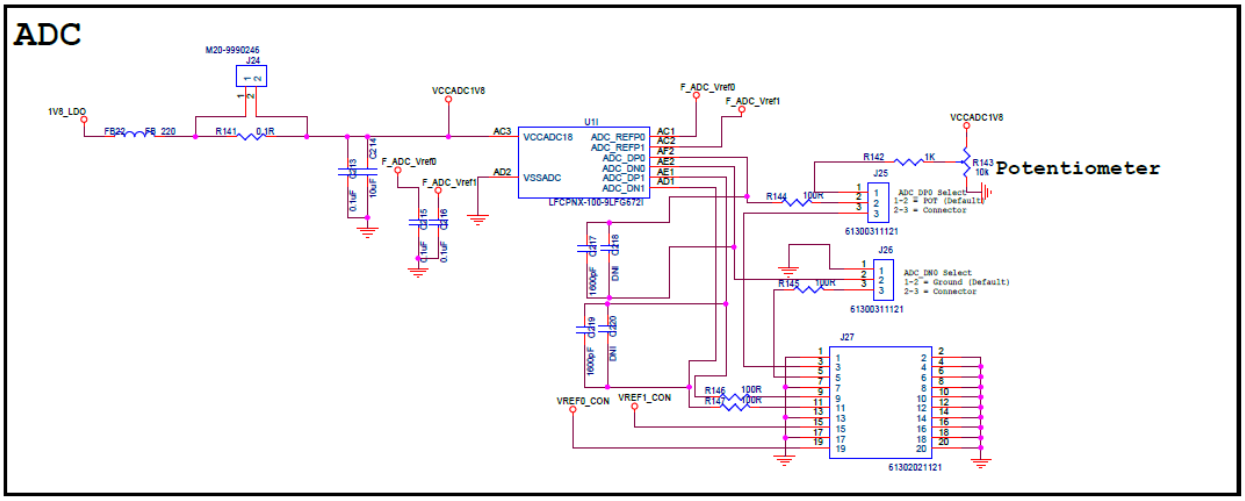


**Figure 3.8. ADC Inputs**

# 4. Design Overview

This demo design consists of several modules that are shown in the block diagram of Figure 4.1. The light grey background is the FPGA, and the rest are peripherals on the board.

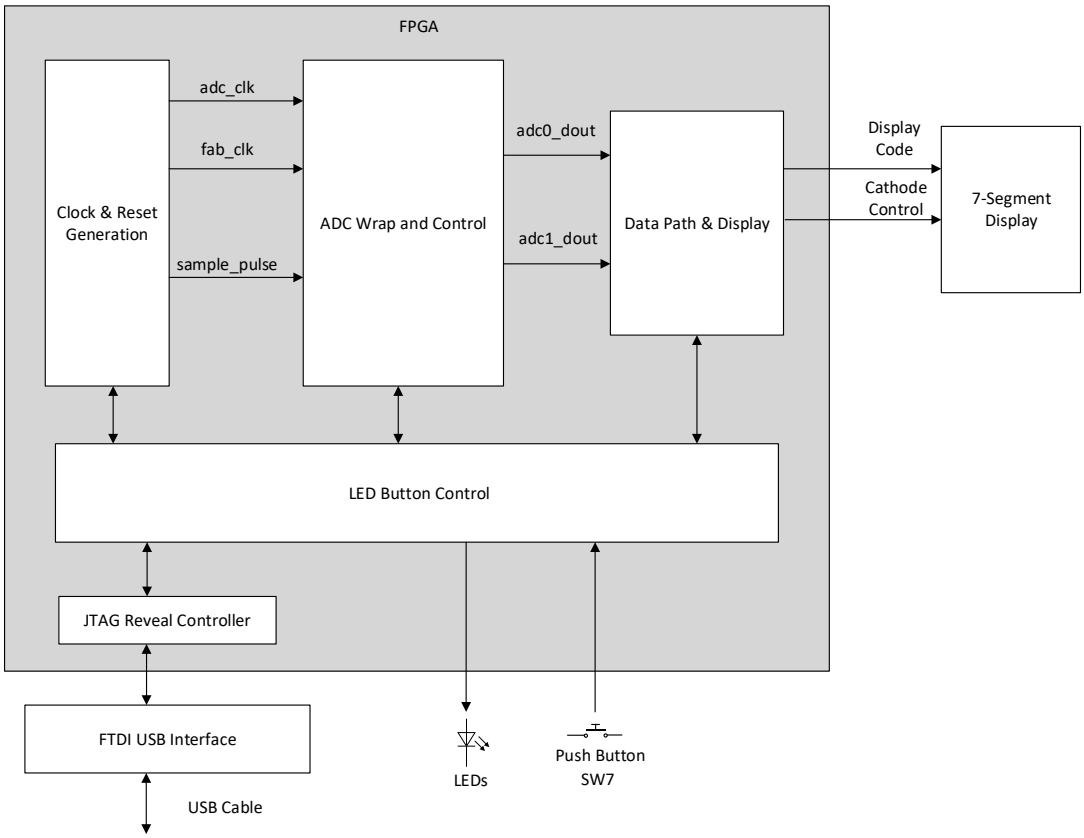


**Figure 4.1. ADC Demo Design Block Diagram**

The functions of the blocks are:

- Clock and Reset – Generates fab_clk and adc_clk from a PLL which gets a reference clock from an oscillator. This block also generates sample_pulse if fixed frequency sampling is required.
- ADC Wrap – Instantiates the ADC and the control FSM.
- Data Path and Display – Scales the ADC output into volts or degrees of Kelvin, Fahrenheit, or Celsius.
- LED Button Control – Debounces external switches, encodes Reveal top controller output into control signals, and drives LEDs.

## 4.1. Hard IP Blocks Used

The FPGA has several hard IP blocks, such as ADC, PLL, and Oscillator. You can instantiate them from the IP catalog in the Lattice Radiant software. In the Lattice Radiant software, select **Tools** > **IP catalog**. Double click the target module from the IP catalog to generate it.

### 4.1.1. Oscillator

An on-die oscillator is used to generate the clock for the ADC and control logic. There is no external oscillator used on the board. Make the following configurations and generate the oscillator module, as shown in Figure 4.2.

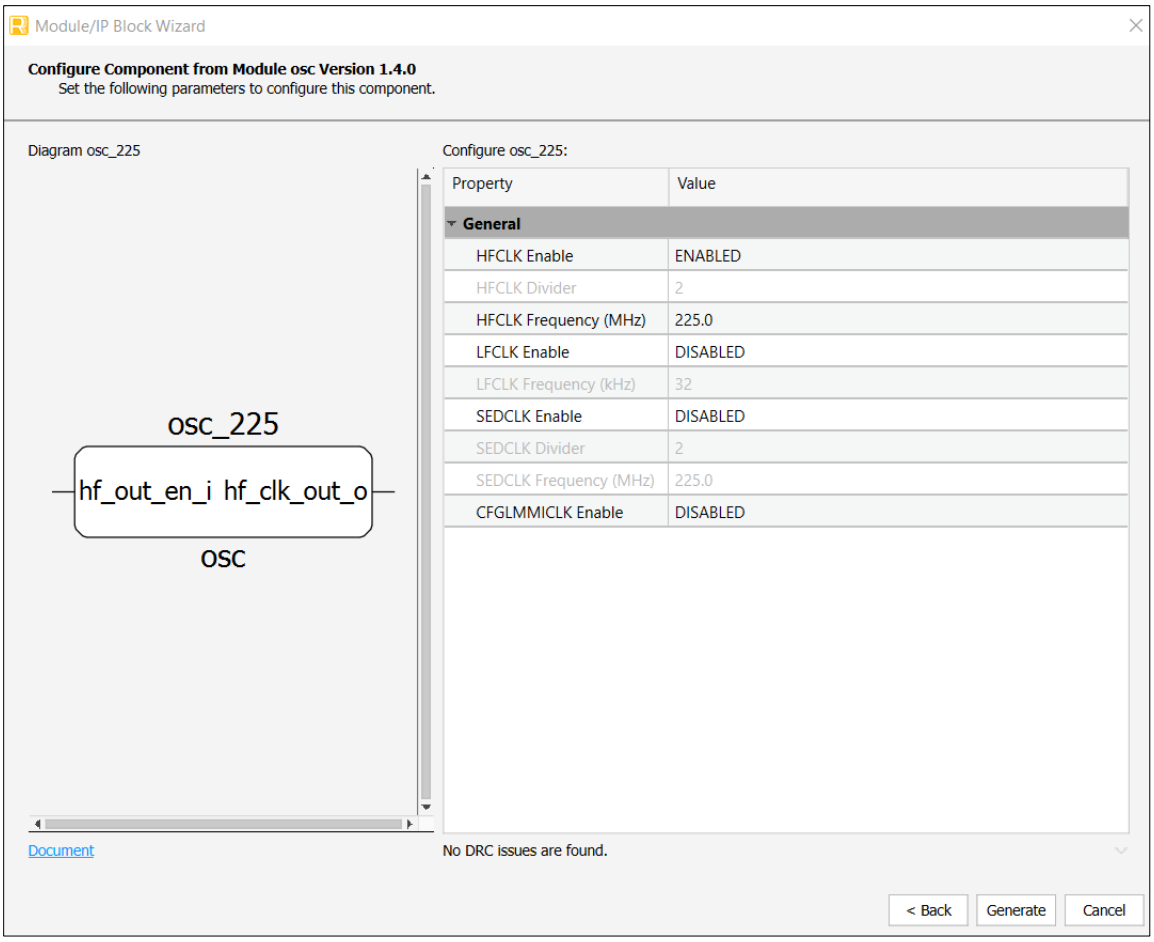


**Figure 4.2. Oscillator Configuration GUI**

After the oscillator is generated, it can be instantiated into the RTL, as shown in Figure 4.3.

```
osc_225 osc_225_inst (
    .hf_out_en_i  (1'b1    ),
    .hf_clk_out_o (clk_225 )
    ) ;
```

**Figure 4.3. Oscillator Instance**

### 4.1.2. PLL

The ADC converter clock, adc_clk, is hardwired on the die to the 4th secondary clock output of the lower right PLL. However, the design must specify the settings and connections for the proper operation. The input clock to PLL is the 225 MHz clock from the oscillator. In this design, the fab_clk comes from the same PLL, but it can come from any PLL. The output clocks from the PLL are:

- 40 MHz clock for the fabric clock. This is used for the ADC control interface, and all ADC I/O are controlled by this fabric clock. This is the fastest possible frequency for the fabric clock.
- 50 MHz ADC clock used for the analog logic of the ADC. Hardwired to the 4th secondary clock on the lower right PLL. This is the max frequency for adc_clk.

The external reset input from the board goes to the PLL reset. The PLL lock is used as the reset for the RTL, so that the logic of the ADC works only when the PLL is stable.

The primary and secondary clocks can be configured using the IP catalog GUI. Then, use the **Calculate** button to confirm the clock frequencies and validity. Then, click the **Generate** button (Figure 4.4).

Initially, the design can run through Place and Route without specifying the PLL location. There are four PLLs available, one in each corner of the device, URC (Upper Right Corner), ULC (Upper Left Corner), LRC (Lower Right Corner), and LLC (Lower Left Corner). Normally, the LRC PLL gets automatically connected to the ADC. However, if you get an error, you can manually configure the placement logic. Near the bottom of the Signal or Pad Report, the locate statement for the PLL can be copied and pasted into the Post-Synthesis Constraint .pdc file. The location within the braces can be edited to read {PLL_LRC}. Then, when the tools are re-run, the settings and connections are assigned to the correct PLL for the ADC.

After the IP is added to the project, the ports are copied from the generated wrapper module to the demo's top-level module and connected to the inputs and wires, as shown in Figure 4.5.

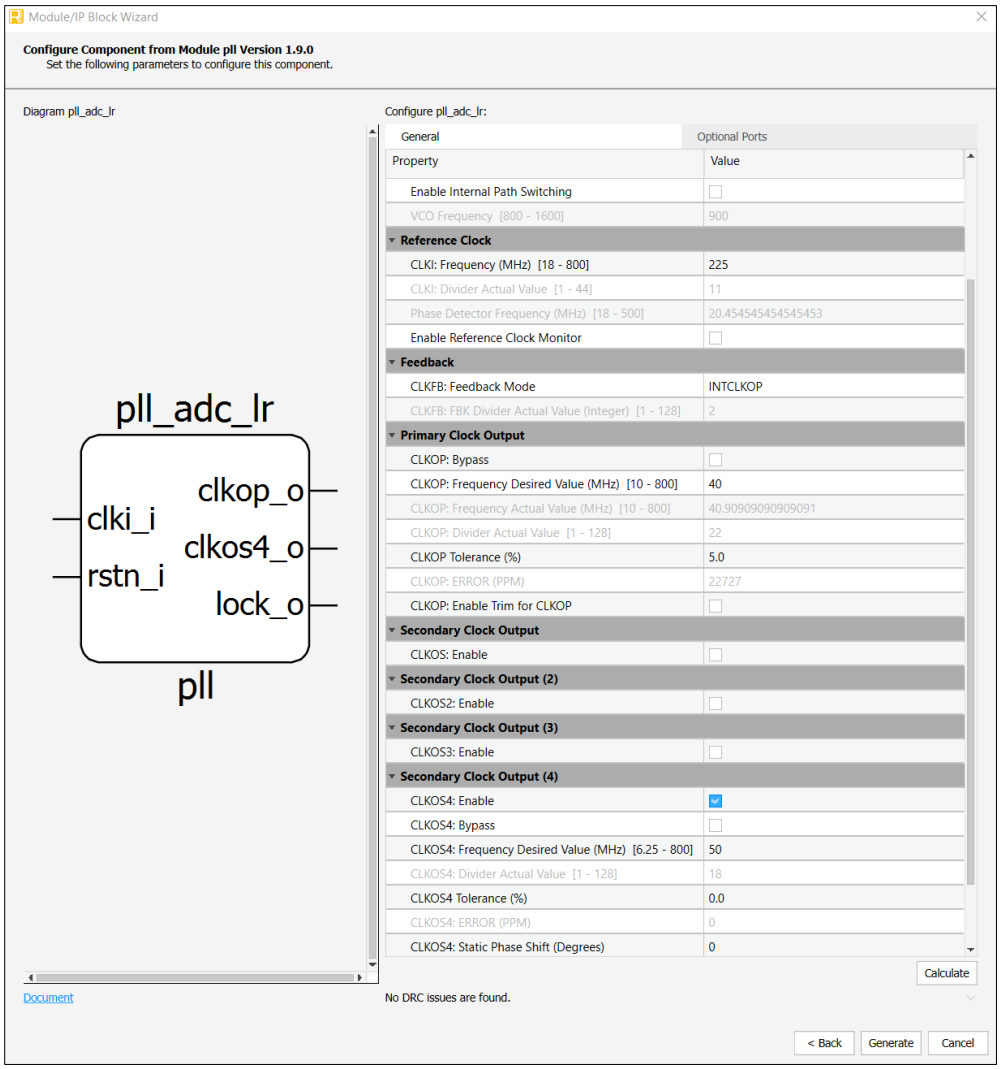


**Figure 4.4. PLL Configuration GUI**

### 4.1.3. ADC

The ADC Core module is generated from the user input to the Module/IP Block Wizard. This IP is located in the Architecture_Modules folder and make sure not to select the ADC Sequencer module. Configure the ADC Core module, as shown in Figure 4.5. Do not enable any comparators or additional ADC channels, as they are not used in this design. The additional ADC channels get their input voltage from pins on the FPGA that have multiple usages. On the CertusPro-NX board, these pins are used for DDR and other uses. refer to ADC User Guide for Nexus Platform (FPGA-TN-02129). If the additional I/O or comparators are enabled, the Module/IP Block Wizard would add them to the generated wrapper module.

The ADC should also be instantiated like the PLL and oscillator. It is instantiated in the adc_wrap block.

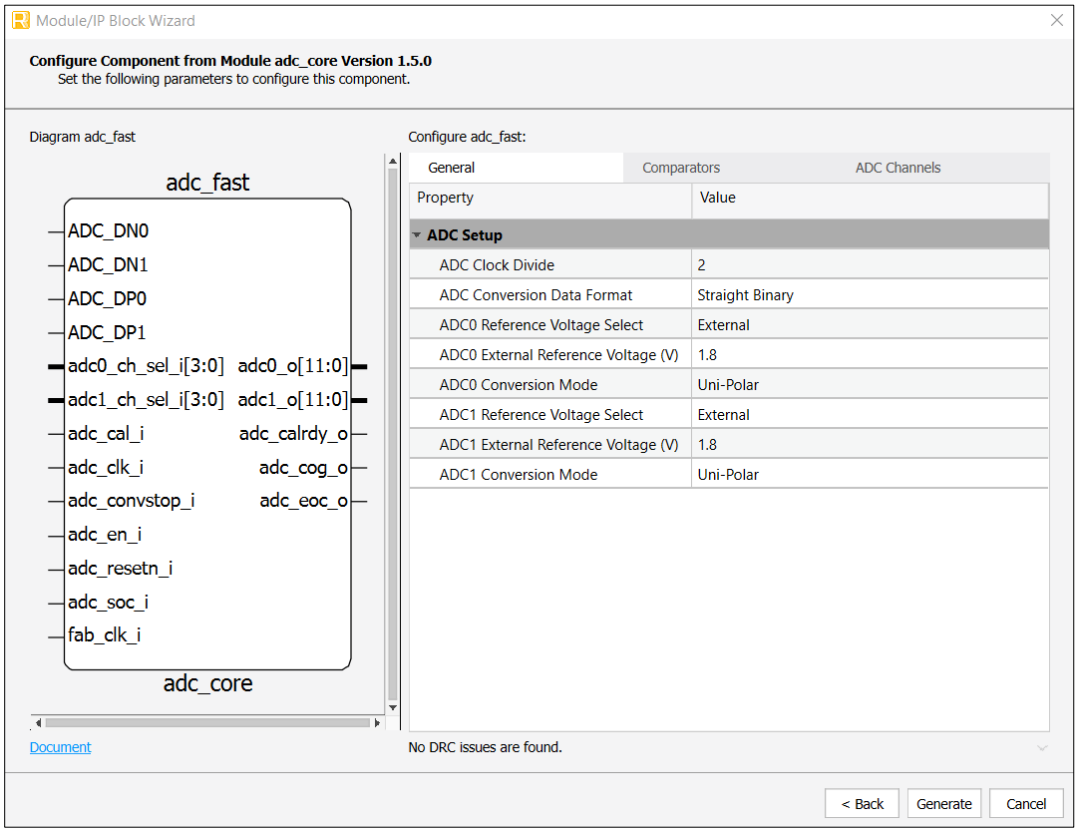


**Figure 4.5. ADC Configuration GUI**

## 4.2. ADC Control

The ADCs have selectable inputs, as they can measure DP or DN pins or internal voltages or temperatures. For details on selectable inputs, refer to ADC User Guide for Nexus Platform (FPGA-TN-02129). The control FSM provides adc_soc for conversion start, and monitors adc_cog for conversion ongoing, and adc_eoc for end of conversion. The timing and pulse width of the adc_soc pulse should be sufficient so that the clock domain crossing logic inside the ADC IP works properly.

It is recommended that you use the FSM in the RTL described in Figure 4.6, as this provides the highest sampling rate along and is guaranteed to work for all combinations of adc_clk, fab_clk, and internal ADC clock divider. It has the following inputs.

- adc_resetn – This signal goes to both ADC IP and the control FSM. If 0, the ADC is internally reset and loses its calibration information. The control FSM goes to the reset state. If adc_resetn=0, the ADC is in a low power state.
- adc_en – This signal goes to both ADC IP and the control FSM. If 0, the ADC is internally clock-gated and takes slightly less power. It cannot give any output. However, calibration information is not lost. The control FSM goes into a state where it waits for the enable to become 1.
- sample_pulse – Before making SOC=0, the control FSM waits for sample_pulse to become 1 after COG becomes 1. This allows a programmable sampling frequency. If it is always kept high, a new sample is taken as soon as the EOC pulse appears. This is the max sampling frequency for the given adc_clk and fab_clk frequency, but the sampling period can change slightly due to CDC synchronizers inside the IP taking between 2 and 3 clock cycles.

At power on reset or any time that adc_resetn has a rising edge, the ADC needs to be calibrated. The FSM handles this process. Calibration runs for a period of about 2700 ADC internal clock cycles, which is adc_clk divided by the clock divider value. However, the duration also depends on the fab_clk frequency. The FSM measures the calibration run time in fab_clk cycles and stores it in rvl_cal_run_time registers that can be accessed in Reveal (Figure 3.4).

The FSM also has a timeout mechanism that gives a new adc_soc_i pulse in case the adc_eoc_o pulse does not come at the expected time, due to timing errors in the ADC hard IP. It can be enabled by making rvl_eoc_timeout_enable=1. The timeout period is given by the EOC_TIMEOUT_PERIOD parameter, which should be set at least 1.5 times the expected time at which the adc_eoc_o pulse is supposed to appear. To find this period, use the formula below.

$$EOC\_TIMEOUT\_PERIOD = 66(T_{ADCCLK} * CLKDIV / T_{FABCLK}) + 35 \qquad \textbf{Eq 1}$$

Round up in the division of equation 1 if you get a fraction. If a timeout error occurs, eoc_timeout_error is 1, as this indicates the ADC output value is outdated and comes from the previous completed conversion.
The eoc_timeout_error_sticky bit remains set on an EOC timeout condition until the ADC is reset. EOC timeouts are not expected to happen very frequently, but this is a failsafe mechanism. Normal operation is shown in Figure 4.7, where the timer runs but never reaches its terminal value as the adc_eoc_o pulse appears. A timeout example is shown in Figure 4.8. This timeout case is not expected to happen, and the picture is taken only by making the EOC_TIMEOUT_PERIOD too low.

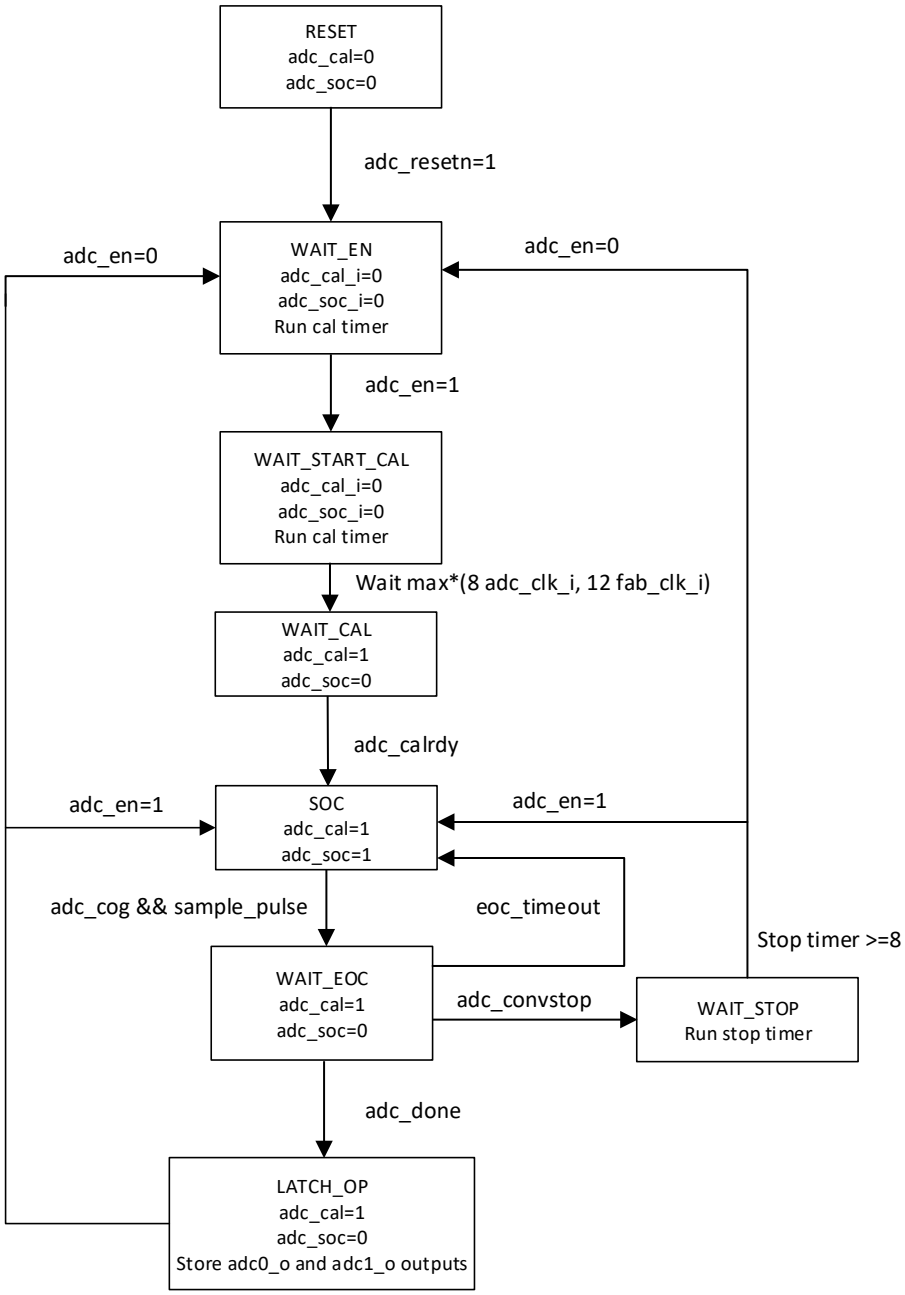


**Figure 4.6. ADC Control FSM**

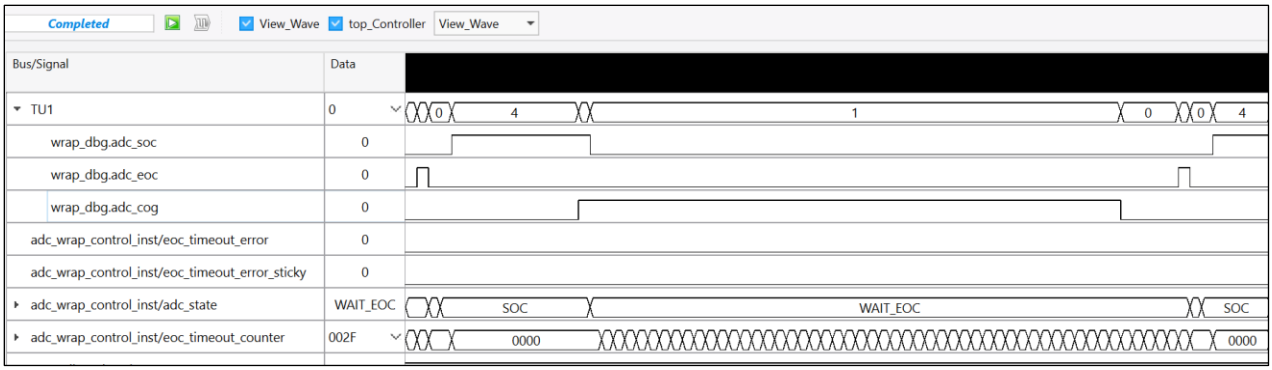


**Figure 4.7. End Of Conversion Timer Running with No Error, Expected Behavior**

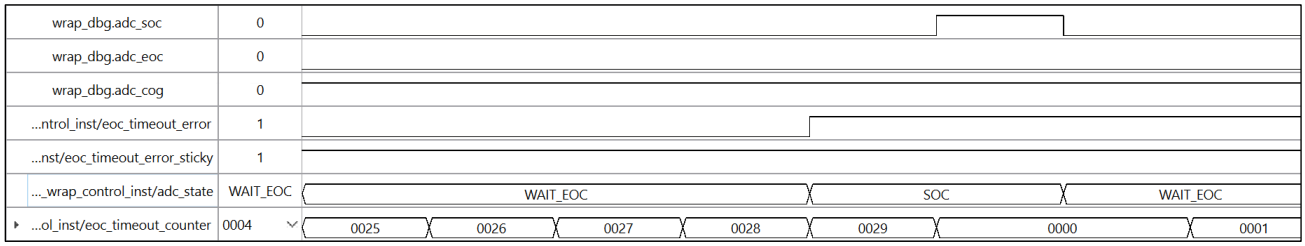


**Figure 4.8. End Of Conversion Timer Timeout and Giving Fresh SOC, Not Expected to Happen**

# 5. Data Path and Display Logic

Logic in this section converts the raw ADC output into a form that can be displayed on the 7-segment LED.

## 5.1. ADC Binary to Bolts Module

This module takes the 12-bit ADC result as an input and converts it into a 3-digit Binary Coded Decimal (BCD) Voltage output. This module is based on a state machine that sequentially subtracts a binary value from the ADC result while simultaneously adding to the decimal result. A separate state is used for each of the three digits. This module accepts two real parameters, ADC_VREF and ADC_VSCALE.

ADC_VREF is used in the module to compute the binary parameter values of 1.0 V, 0.1 V, and 0.01 V that are used in converting from binary to BCD. For the CertusPro-NX Versa board used in this demo, the value is 1.805 V.

ADC_VSCALE tells the module what scaling value to use. The Nexus ADC hardware includes a 2/5 voltage divider between the supply pins and the ADC input. So, for this design, a scaling value of 2.5 is applied to the ADC result, so that the BCD value represents the voltage at the pin rather than the raw ADC input. The ADC_VSCALE is applied when the adc_scale_i input is set and should only be used when measuring internal voltage rail channels.

Note that the $floor function is not yet supported by Synplify. Therefore, you must hardcode values so that the input parameters are not affected.

```
localparam real ADC_LSB_VOLT = ADC_VREF / 4096.0 ;//0.000439453125

//Multiply up by factor of 16 to push round-off errors into micro-volts
localparam real ADC_VOLT_1V00 = 1.00 * 16.0 / ADC_LSB_VOLT;//36408.8888
localparam real ADC_VOLT_0V10 = 0.10 * 16.0 / ADC_LSB_VOLT;//3640.88888
localparam real ADC_VOLT_0V01 = 0.01 * 16.0 / ADC_LSB_VOLT;//364.08888
// Convert real to integer
localparam [15:0] ADC_BIN_VOLT_1V00 = ADC_VOLT_1V00;//'h8E38  //We want to use $floor(ADC_VOLT_1V00), but as Syn tool does not handle it take only the int part
localparam [15:0] ADC_BIN_VOLT_0V10 = ADC_VOLT_0V10;//'hE38   //We want to use $floor(ADC_VOLT_0V10), but as Syn tool does not handle it take only the int part
localparam [15:0] ADC_BIN_VOLT_0V01 = ADC_VOLT_0V01;//'h16C   //We want to use $floor(ADC_VOLT_0V01), but as Syn tool does not handle it take only the int part

// Scale the values for VCC monitors
localparam real ADC_VCC_VOLT_1V00 = ADC_VOLT_1V00 / ADC_VSCALE;
localparam real ADC_VCC_VOLT_0V10 = ADC_VOLT_0V10 / ADC_VSCALE;
localparam real ADC_VCC_VOLT_0V01 = ADC_VOLT_0V01 / ADC_VSCALE;
// Convert real to integer
localparam [15:0] ADC_BIN_VCC_VOLT_1V00 = ADC_VCC_VOLT_1V00;//$floor(ADC_VCC_VOLT_1V00);
localparam [15:0] ADC_BIN_VCC_VOLT_0V10 = ADC_VCC_VOLT_0V10;//$floor(ADC_VCC_VOLT_0V10);
localparam [15:0] ADC_BIN_VCC_VOLT_0V01 = ADC_VCC_VOLT_0V01;//$floor(ADC_VCC_VOLT_0V01);
```

**Figure 5.1. ADC to BCD Scaling Parameters**

## 5.2. ADC Binary to Temperature Module

Similar to the ADC_binary2volts module, this module uses the ADC result from the DTR channel and converts it into a 3-digit BCD temperature output. This module also implements a state machine to sequentially subtract a binary value from the ADC result while simultaneously adding to the decimal result. A separate state is used for each of the three digits. It currently works only for VREF=1.8 V. It gives the temperature in Kelvin. See the DTR section of the ADC User Guide for Nexus Platform (FPGA-TN-02129) for more information.

## 5.3. Display

The BCD value calculated from the binary to volts or binary to temperature is converted to a 7-segment code for display. The display also has a multiplexing logic, as the 7-segment display is a common cathode display.

# 6. Reveal

This demo makes use of Reveal Logic Analyzer and Top Controller, the latter has already been discussed in Testing and Debugging. The Reveal Logic Analyzer can be used to see internal waveforms of the control logic. The Reveal Logic Analyzer runs on a 100 MHz clock that is synchronous to the fab_clk so that all signal changes can be seen.

To open Reveal Logic Analyzer from the Lattice Radiant software, use the file rvl_1_1_1.rva and then select the logic analyzer from the drop-down menu, as shown in Figure 6.1.

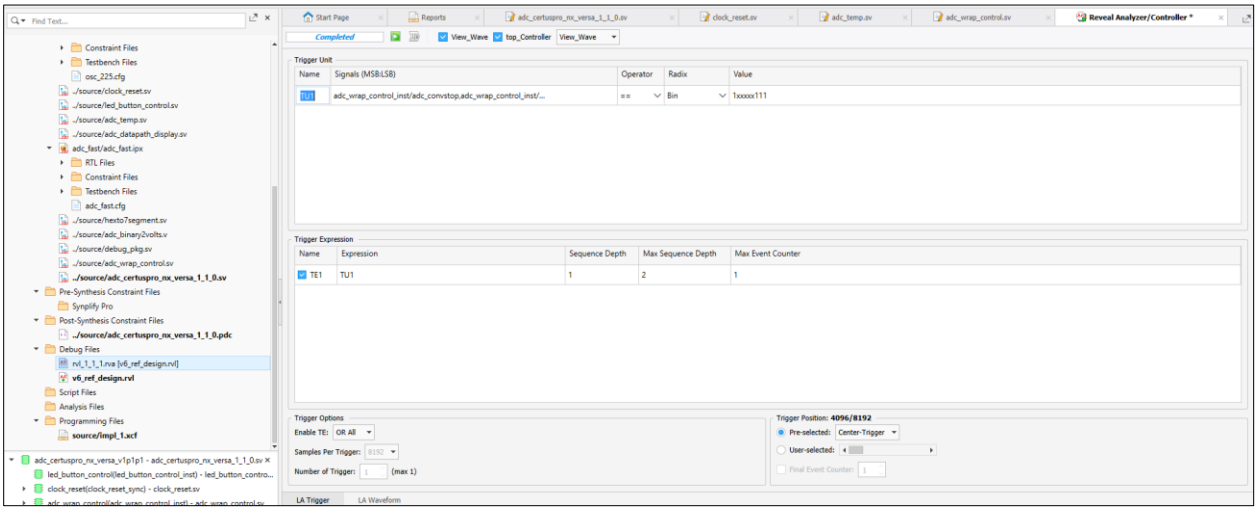

**Figure 6.1. Reveal Logic Analyzer Trigger Setup**

Once the logic analyzer is open, you can select the trigger in the GUI. You can also select the trigger position in the waveform using the bottom right drop-down menu (Figure 6.1).

Using the trigger, you can capture a waveform of the ADC working, as shown in Figure 6.2.

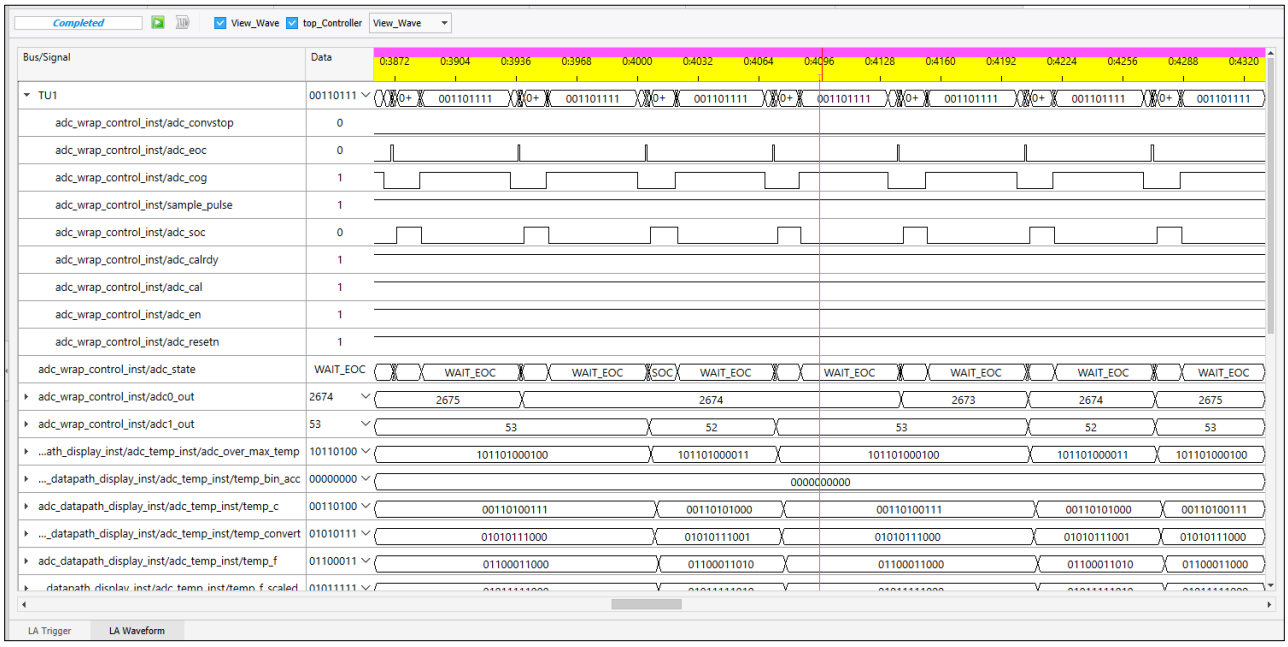


**Figure 6.2. ADC Signals Captured by Reveal Logic Analyzer**

# 7. Creating the Demo Design

The procedure below describes how to create the demo design on the CertusPro-NX Versa Board. The procedure for other Nexus devices is similar.

## 7.1. Required Hardware and Software

The FPGA board used in the demo is the CertusPro-NX board. The ordering part number of the board is LFCPNX-VERSA-EVN. The Lattice Radiant 2024.1 software is used. Check for all updates, as shown in Figure 7.1.

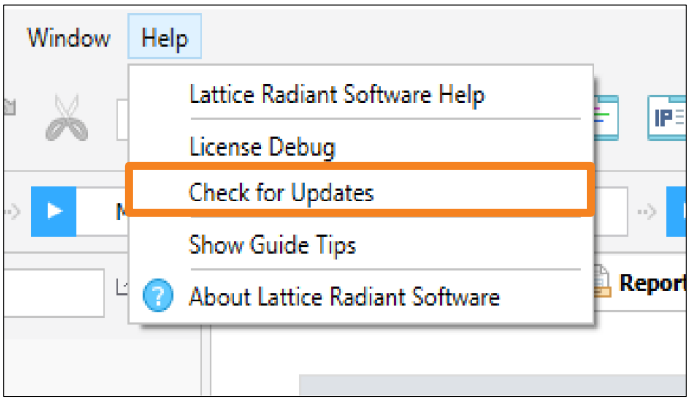


**Figure 7.1. Check for Updates**

## 7.2. Creating a New Project

1. Select **File** > **New** > **Project** to create a new project.
2. Choose the targeted FPGA family, device, and package to match the board, as shown in Figure 7.2. Make sure these selections are correct. As for the operating condition and performance grade, they do not directly affect the bitstream. They are only used for timing and performance analysis.

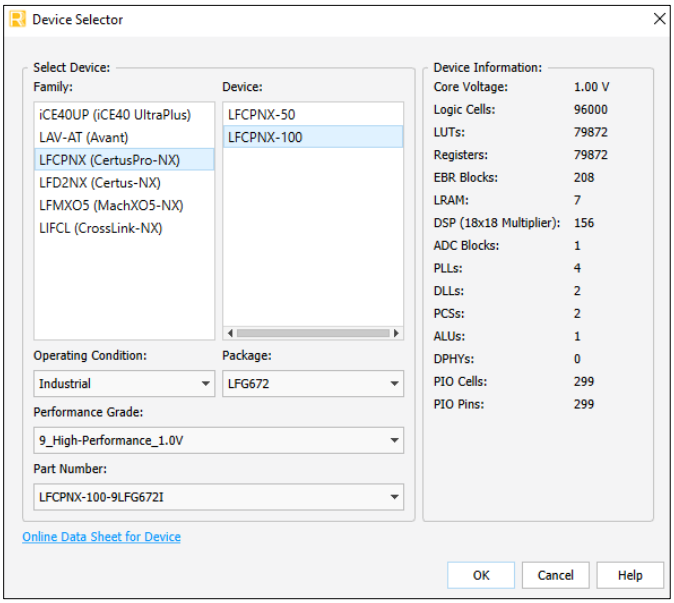


**Figure 7.2. Select Targeted FPGA**

3. Click **OK** and save the project.

## 7.3. Selecting IP blocks

The FPGA has several hard IP blocks that perform analog functions and can be instantiated. The ADC is one of them, which internally contains two Successive Approximation Register (SAR) ADCs. The ADC needs a clock that comes from a PLL which needs an input oscillator. All these need to be added.

In the bottom left of the Lattice Radiant software GUI, choose the IP Catalog tab and from that instantiate the following blocks. The version number should be equal to or higher than the number shown in Figure 7.3.

- ADC core
- Oscillator
- PLL

### 7.3.1. Oscillator

To instantiate the oscillator, select it and double click (Figure 7.3).

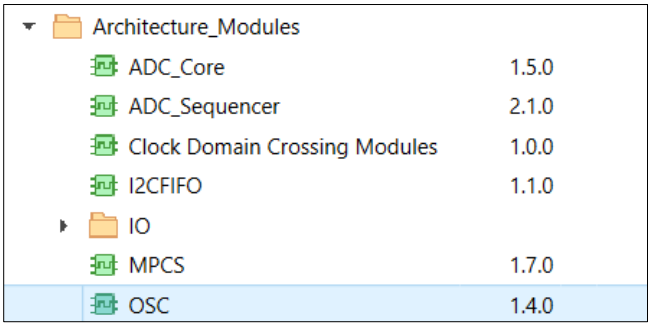


**Figure 7.3. Instantiate the Oscillator**

In the oscillator configuration, enable the HF clock and set its output frequency to 225 MHz. Disable the LF clock, as shown in Figure 7.4. After completing the configuration, click **Generate**.

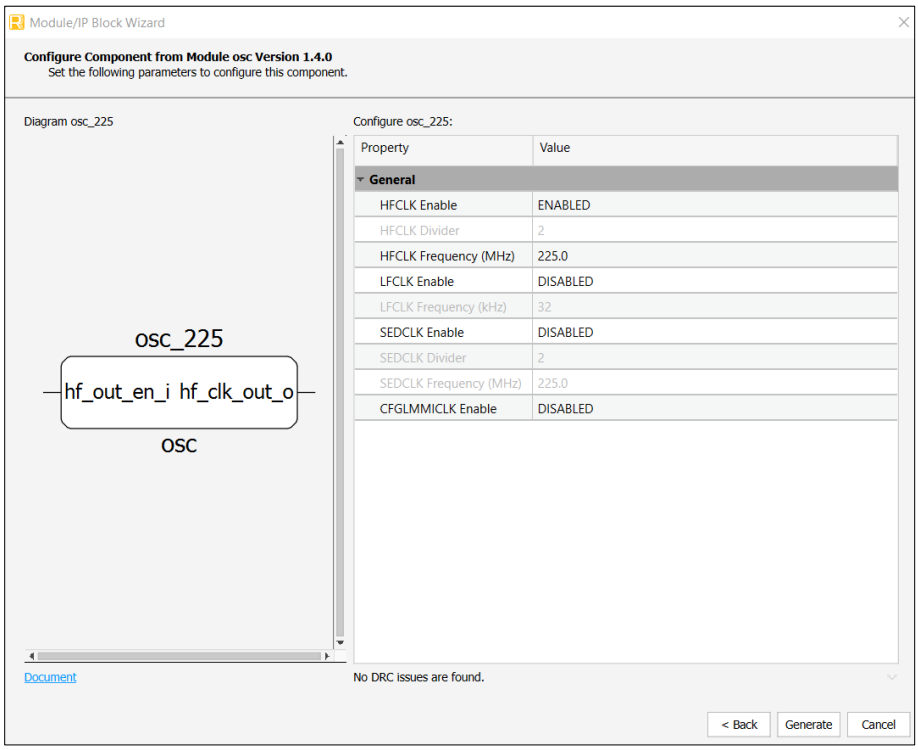


**Figure 7.4. Configure the Oscillator**

### 7.3.2. PLL

To instantiate the PLL, select it from the IP catalog, as shown in Figure 7.5. The FPGA device has four PLLs, one in each corner. The lower right PLL has its secondary output 4 hard wired to the clock input adc_clk for the ADC. The ADC fab_clk input can be connected to any PLL. In view of this, you need to instantiate the lower right PLL. The Lattice Radiant software does this automatically, unless you write constraints overriding it. Therefore, you should be careful while writing constraints. As this demo uses only one PLL and does not involve writing any constraints, the Lattice Radiant software connects it automatically.

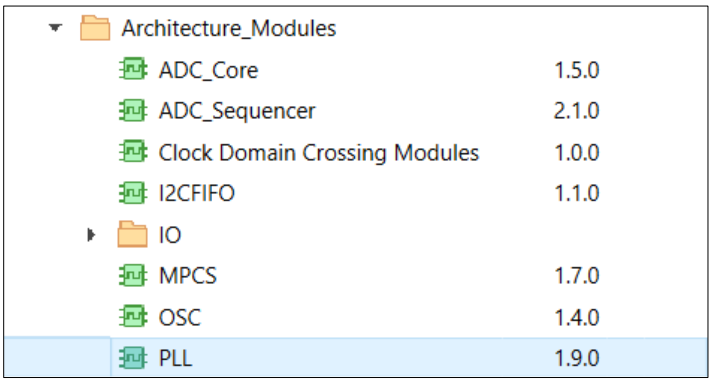


**Figure 7.5. Instantiate the PLL**

Make the following configurations for the PLL module, as shown in Figure 7.6.

- CLKI Frequency: The CLKI Frequency needs to match the oscillator frequency, which is 225 MHz here.
- CLKOP Frequency Desired Value: CLKOP is the fab_clk_i input to the ADC. The CLKOP Frequency Desired Value of 40 MHz is the fastest frequency that meets timing.
- CLKOS4 Frequency Desired Value: CLKOS4 is the adc_clk_i input to the ADC and the CLKOS4 Frequency Desired value of 50 MHz is the fastest frequency that meets timing. The adc_clk_i input to the ADC is hardwired to the lower right PLL secondary output 4. Therefore, you need to select CLKOS4 from the PLL configuration and the Lattice Radiant software assigns this PLL instance to the lower right PLL.

After making the above configurations, click **Calculate** to check if there are Design Rule Checking (DRC) issues. There should be no DRC issue if you use the values mentioned above. Then, Click **Generate**.

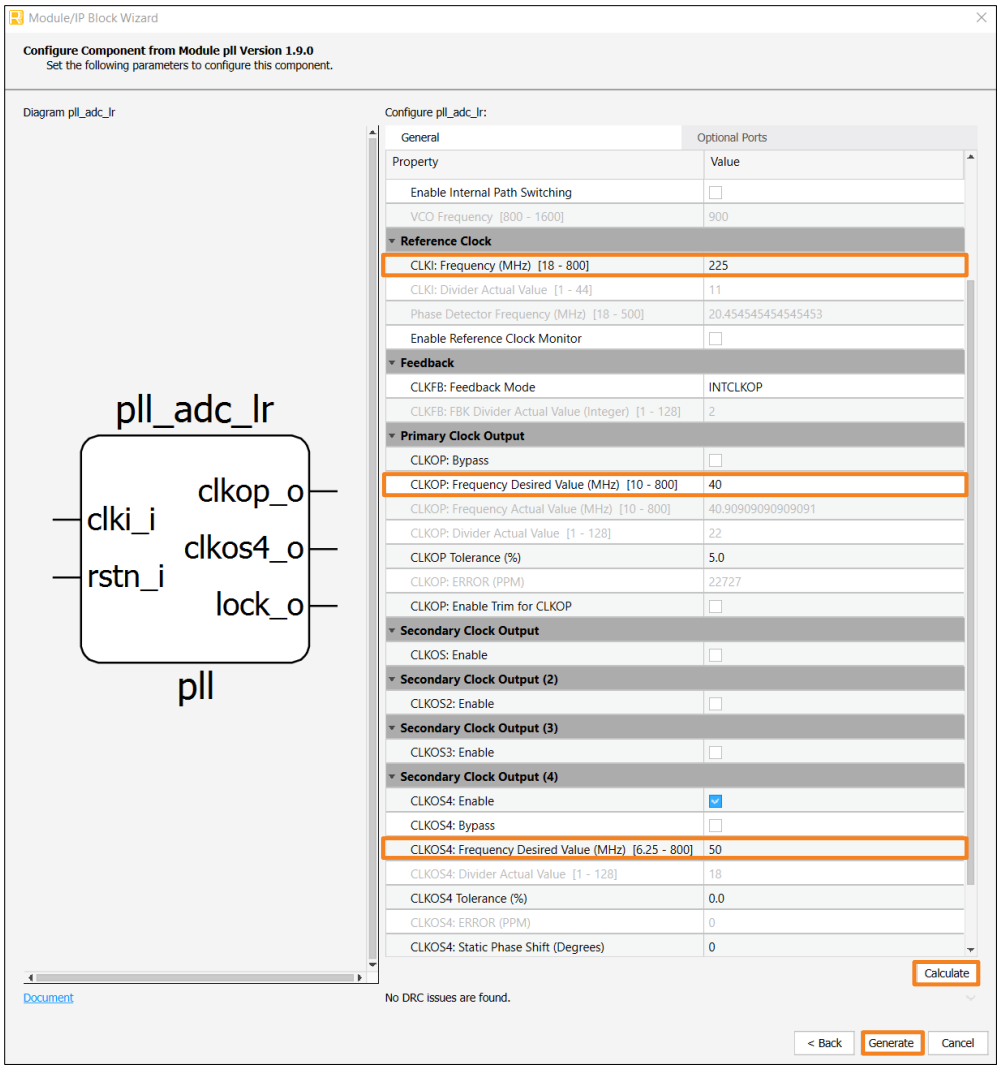


**Figure 7.6. Configure the PLL**

### 7.3.3. ADC

Select the ADC Core module from the IP Catalog and double click, as shown in Figure 7.7. Do not select the ADC Sequencer module.

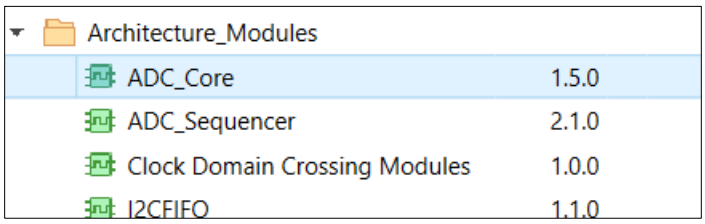


**Figure 7.7. Instantiate the ADC Core**

Make the following configurations for the ADC Core, as shown in Figure 7.8. Keep the default settings for the Comparators and ADC Channels tabs.

- ADC Clock Divide: adc_clk_i is divided by the number of this attribute. Select the smallest value of 2 for the highest sampling frequency.

- ADC External Reference Voltage: The external VREF of 1.8 V is provided on board. Connect Jumper J28 pin 1 and pin 2. Connect Jumper J29 pin 1 and pin 2.
- ADC Conversion Mode: Here the value of Uni-polar means the output goes from 0x0 for input voltage of 0 to 0x3FF for input voltage of VREF. If bi-polar is selected, VREF/2 would output 0.

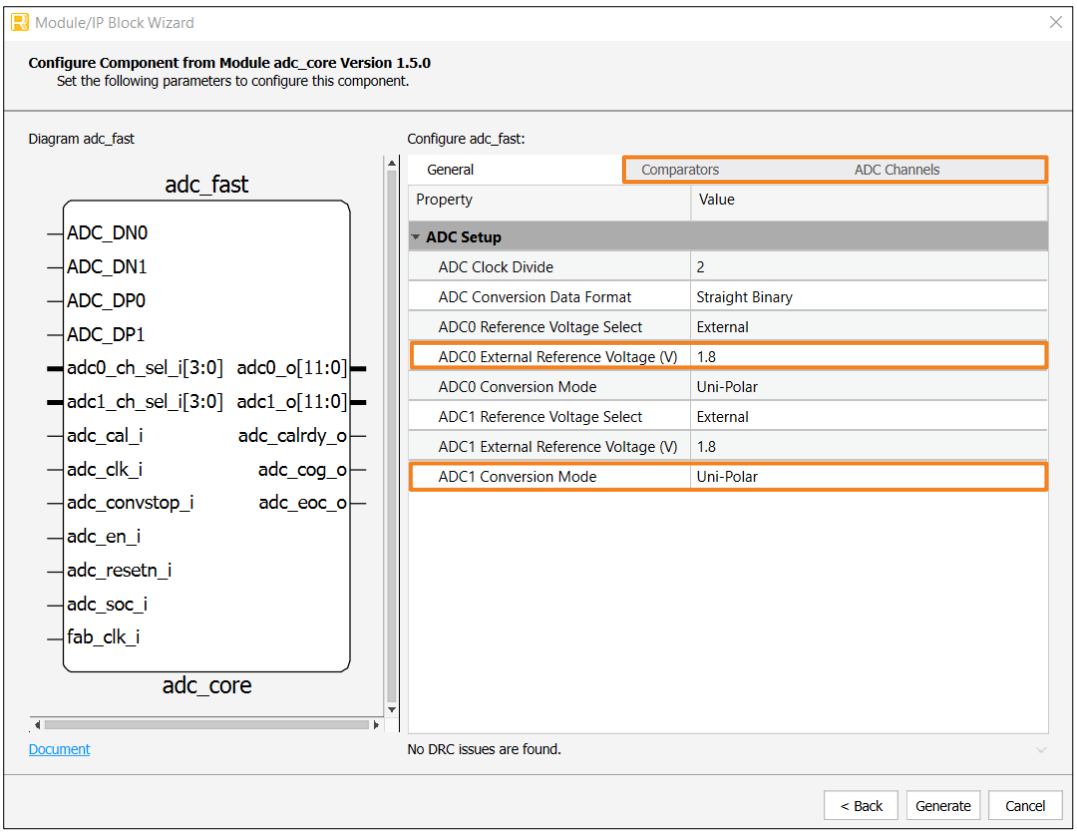


**Figure 7.8. Configure the ADC**

## 7.4. System Verilog and Synthesis

Add the following source files, as shown in Figure 7.9. The list of files to be added are:

- adc_certuspro_nx_versa_1_1_0.sv
  Top level wrapper for the whole design.
- adc_wrap_control.sv
  Instantiate the ADC core generated in the ADC section and its control FSM. This is the file that you need to reuse in your own design.
- adc_temp.sv
  Converts ADC output into a temperature in Kelvin, Celsius, or Fahrenheit in BCD. A value of 0xA displays a negative sign.
- adc_binary2volts.v
  Converts ADC output to a BCD voltage.
- clock_reset.sv
  Instantiates the PLL and generates the clocks used in the system. It also generates strobe signals for the 7-segment display multiplexing.
- led_button_control.sv
  Decodes the Reveal controller and physical push buttons.
- hexto7segment.sv
  Converts BCD to the 7-segment input. An input of 0xA is a negative sign.

- adc_datapath_display.sv
  Wrapper around adc_temp.sv, adc_binary2volts.sv, and hexto7segment.sv.
- debug_pkg.sv
  Creates a System Verilog package for the debug bus datatype.

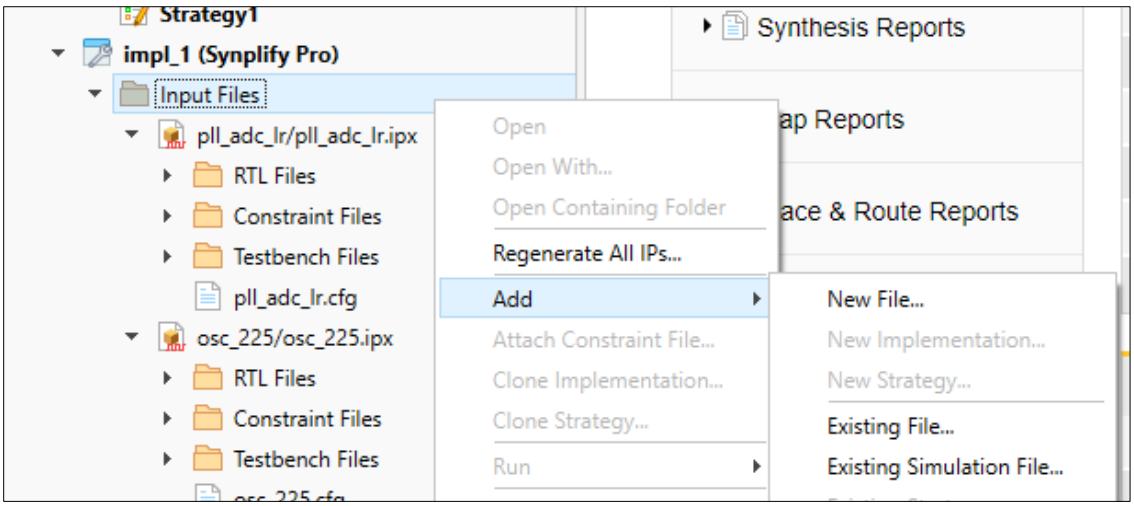


**Figure 7.9. Add Source Code**

After adding the source code, set the top-level unit to the module name of the wrapper, as shown in Figure 7.10. The synthesis tool should be Syplify Pro as LSE does not support System Verilog.

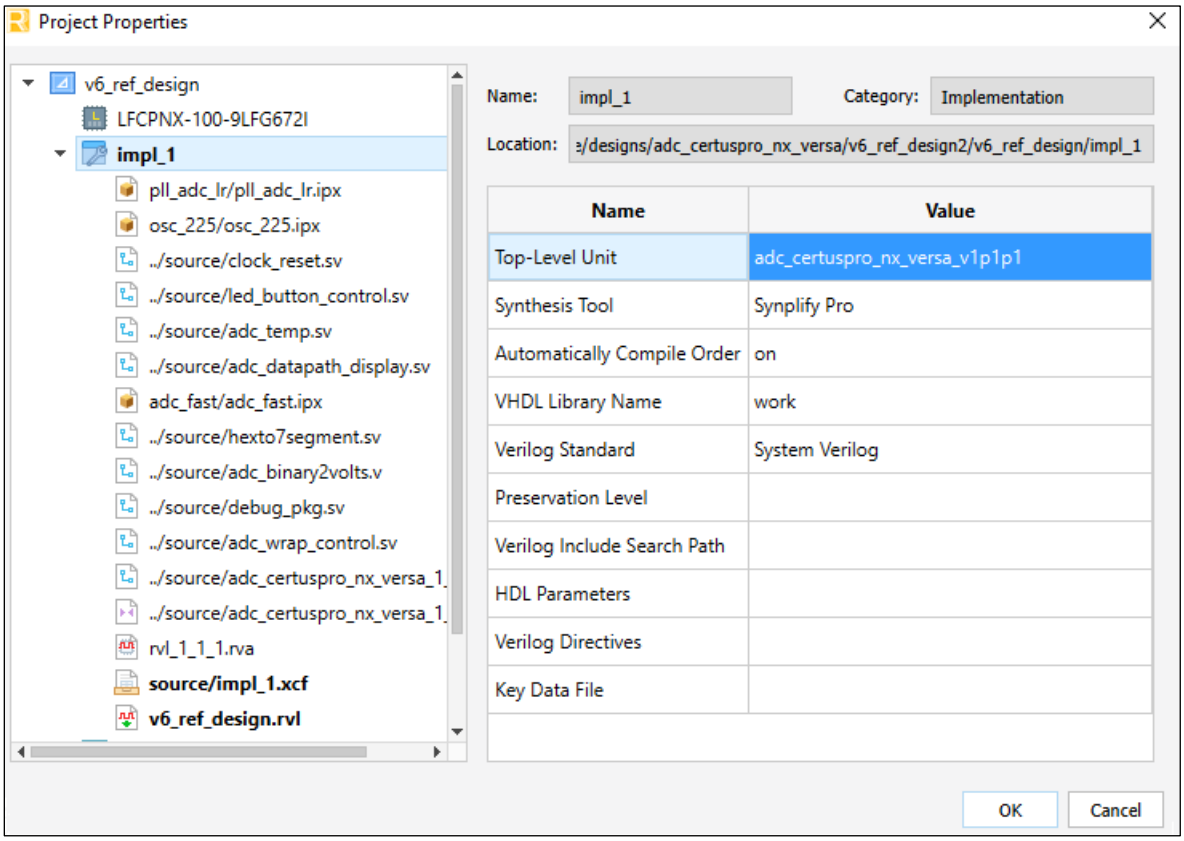


**Figure 7.10. Verilog Configuration**

## 7.5. Post Synthesis and Pin Constraints

The digital I/O need to be assigned to pins through the .pdc file as shown in Figure 7.11. Analog inputs such as DP0, DN0, DP1, DN1 and the voltage reference pins are dedicated inputs and do not need to be added here.

Refer to CertusPro-NX Versa Board Evaluation Board User Guide (FPGA-EB-02053) for the schematic of the board.

The PMOD connector is used as a set of external debug pins to which a scope can be connected. Push button SW7 is used as a stop conversion input.

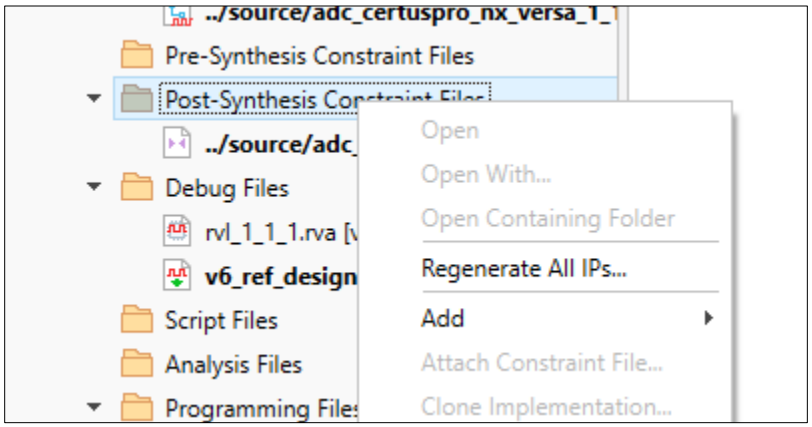


**Figure 7.11. Add Post Synthesis Constraints Using a .pdc File**

## 7.6. Debug and Control

Reveal allows the insertion of a soft logic analyzer that can show internal FPGA signals. It also provides a top controller that provides virtual I/O, such as LED and push button.

### 7.6.1. Logic Analyzer

1. In the Lattice Radiant software, select **Tools** > **Reveal Inserter**.
2. After the Reveal Inserter opens, click **Add core…** > **Add Logic Analyzer** to add a new logic analyzer (Figure 7.12).
3. Rename the Logic analyzer as View _Wave.
4. Drag the fab_clk signal from the design tree pane to the **Sample Clock** box.
5. Check the checkbox for **Include trigger signals in trace data**.

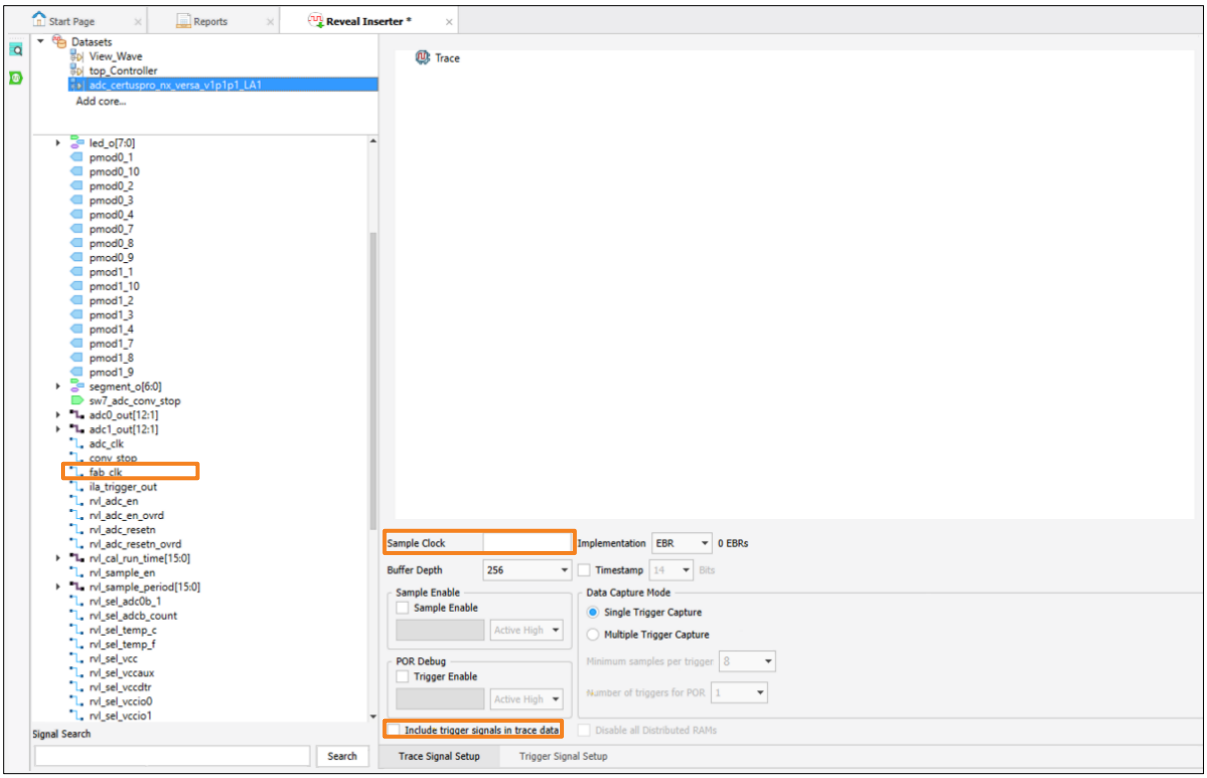


**Figure 7.12. Logic Analyzer Initial View**

6. Click the **Trigger Signal Setup** tab.
7. Add a new trigger unit and add signals to it by dragging the desired signals from the design tree pane to the **Signals (MSB:LSB)** box in the Trigger Unit section (Figure 7.13). If you double click on the names of the signals, as highlighted in Figure 7.13, the **TU Signals** dialog box opens (Figure 7.14).
8. You can reorder these signals using the up and down buttons. Click **Ok** after configuring the signals. Check the checkbox for **Enable Trigger Out** in the Trigger Out section and assign it to ila_trigger_out NET. The ila_trigger_out NET is not visible in the Verilog code but is assigned to a PMOD pin and enables the synchronization of an external scope to the Internal Logic Analyzer (ILA).
9. Once the trigger signals are added, switch to the **Trace Signal Setup** tab and add trace signals. These trace signals cannot be used as part of the trigger expression, but they are displayed in the captured waveform. Check the check box for **Include trigger signals to trace data** (Figure 7.15).

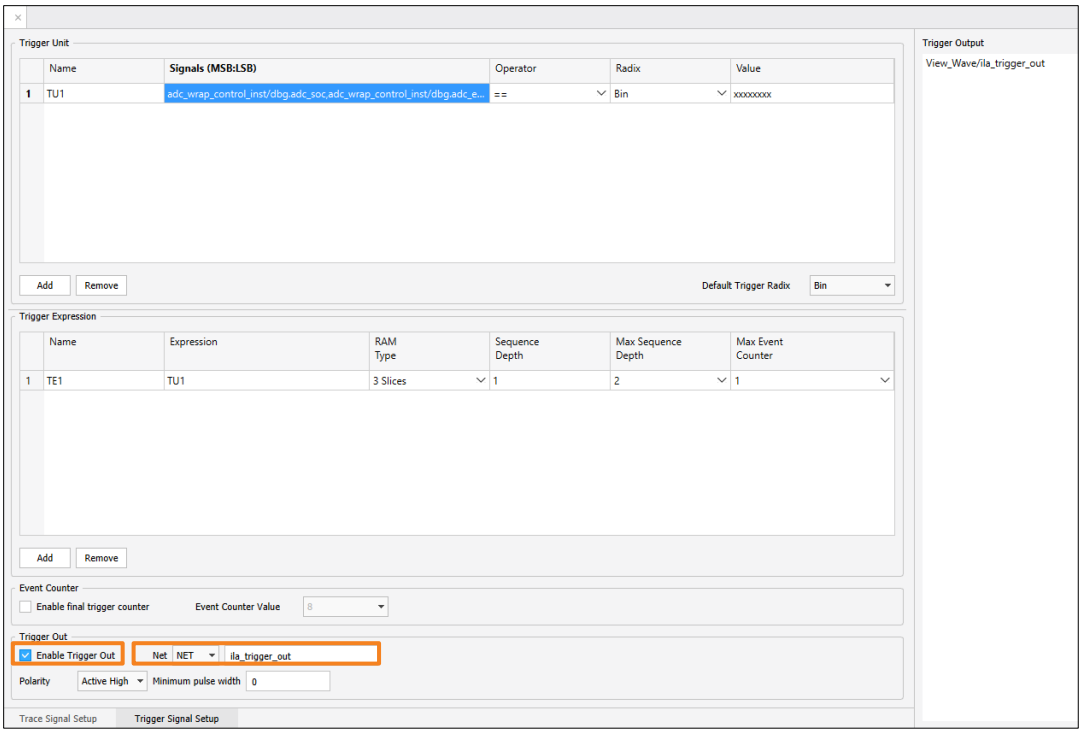


**Figure 7.13. Add a Trigger Unit**

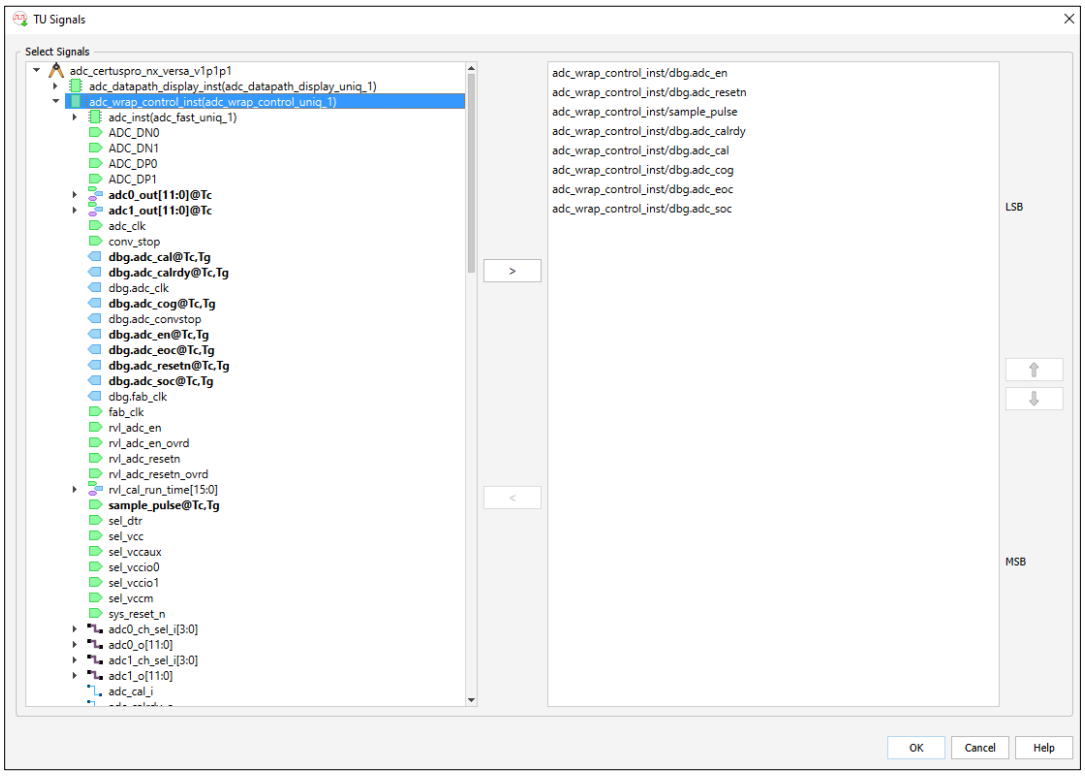


**Figure 7.14. Add Signals to the Trigger Unit**

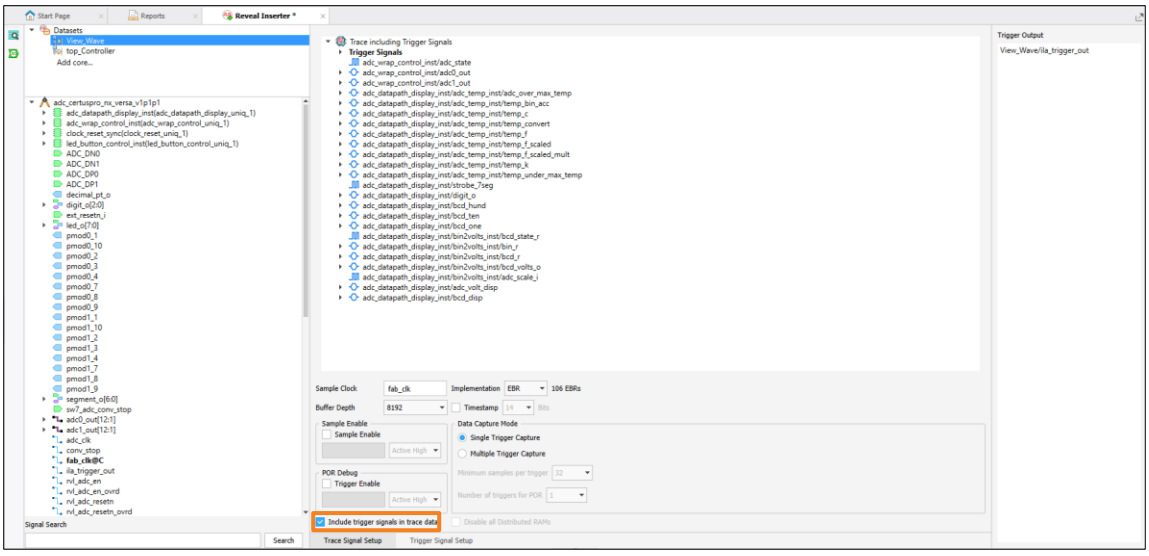

**Figure 7.15. Logic Analyzer Appearance after Adding Trace Signals**

## 7.6.2. Top Controller

From Reveal Inserter, add a top controller by selecting **Add core…** > **Add Controller**. In the top controller, assign virtual switches and LEDs, as shown in Figure 7.16. Then, connect status registers to the user status registers and user control registers, as shown in Figure 7.17 and Figure 7.18.

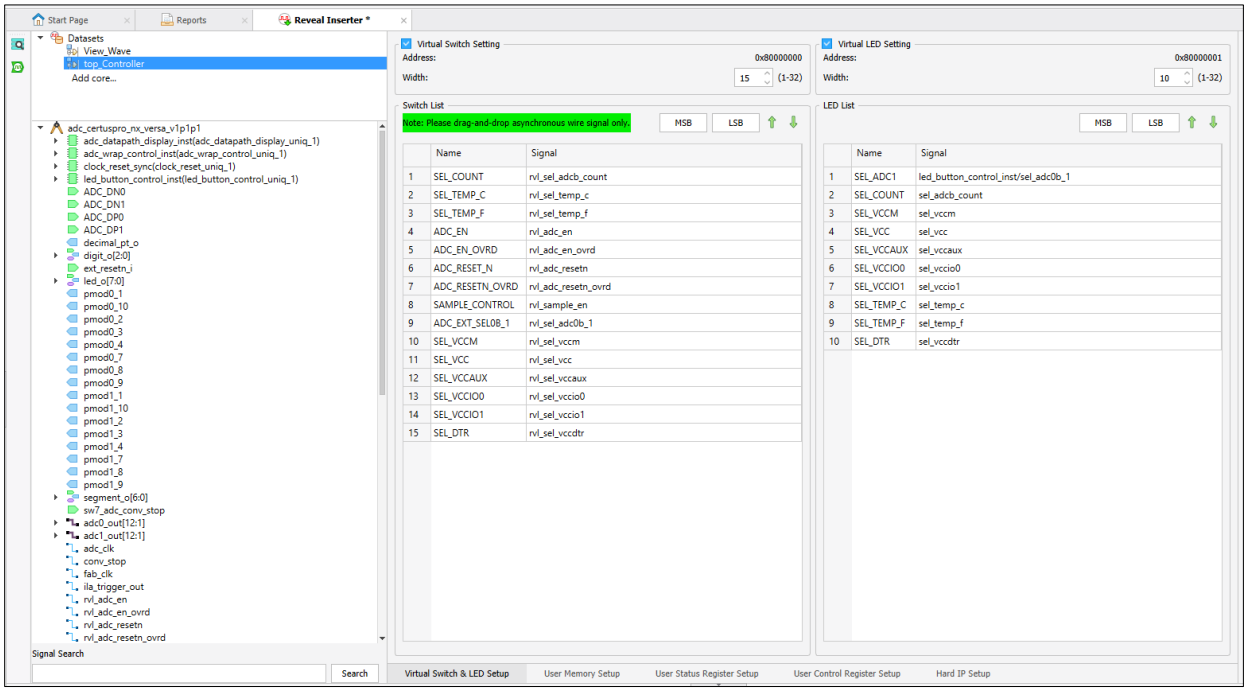

**Figure 7.16. Virtual Switches and LEDs**

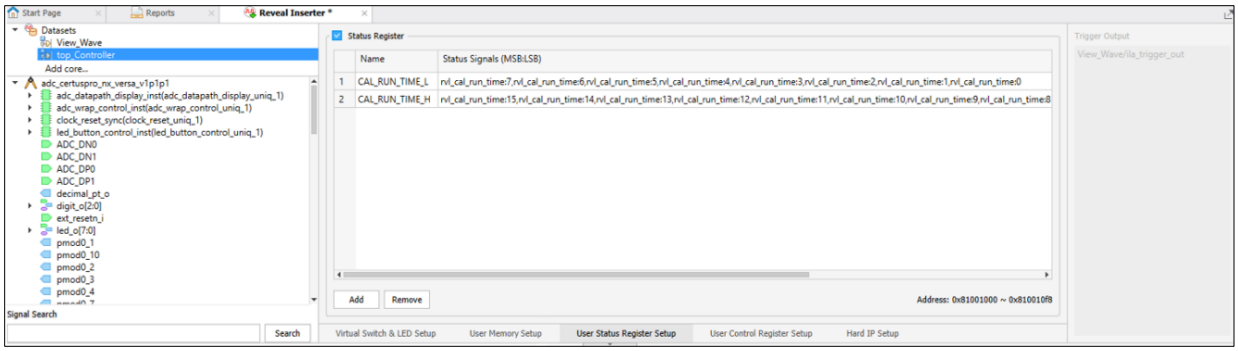


**Figure 7.17. User Status Register Setup**

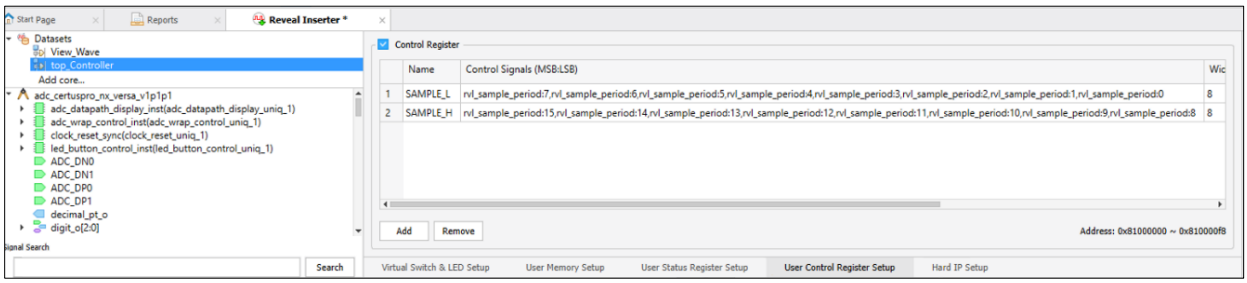


**Figure 7.18. User Control Register Setup**

## 7.7. Running Synthesis and Programming.

In the Lattice Radiant software, Click the **Export Files** button on the top left. It starts from synthesis, runs map, place and route, and generates the bit file.

To open the Programmer tool, select **Tools** > **Programmer** and the Radiant Programmer opens, as shown in Figure 7.19.

Connect a mini USB cable from the mini USB port on the FPGA board to a USB2 port on the laptop. Do not use the USB3 type B port on the board.

1. Set up the cable by clicking the **Detect Cable** button on the top right. If multiple cables are detected, use the lower numbered option, usually location 0. Set the TCK Clock Divider to 3 or higher. This reduces the chance of bit errors, even though it makes programming slower.
2. Point to the .bit file in the GUI. Select the operation as **Fast Configuration** and click the program button. This loads the .bit file into the FPGA configuration RAM but does not alter the flash. When programming is done, the 7-segment display displays a voltage that is the output from the potentiometer. Turning the potentiometer causes the voltage to vary from 0 to 1.8 V.

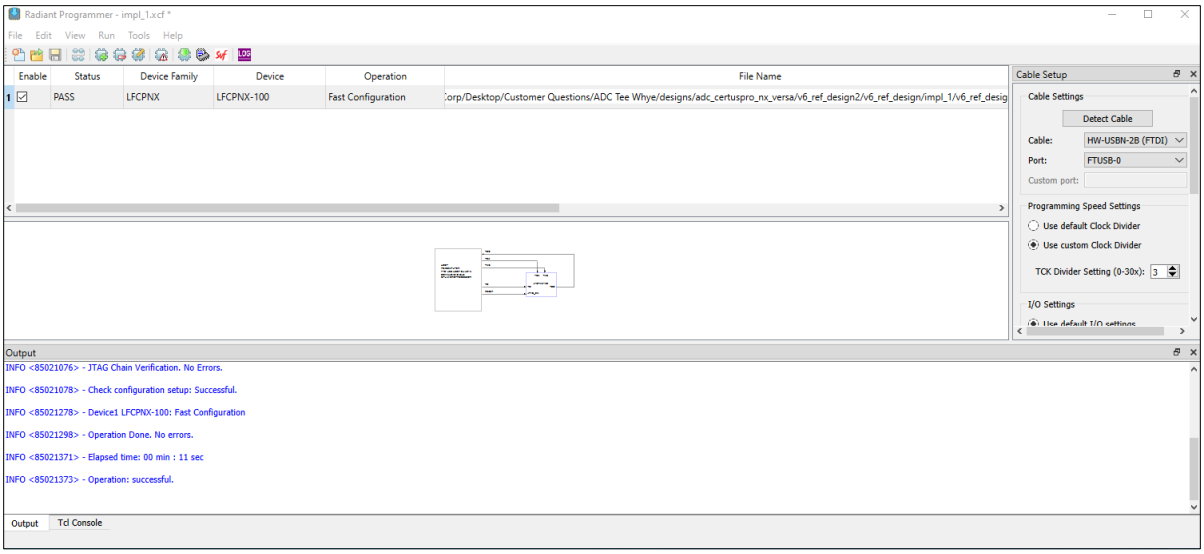


**Figure 7.19. Programmer GUI**

## 7.8. Debug GUI

To open the logic analyzer and controller, click **Tools** > **Reveal Analyzer/Controller** in the Lattice Radiant software.

### 7.8.1. Logic Analyzer

Choose the target logic analyzer from the drop-down menu on the top, as shown in Figure 7.20.


**Figure 7.20. Open the Logic Analyzer**

Set up the trigger on any signal that occurs in the design. The example shown in Figure 7.21 triggers on the SOC signal. Press the run button and the waveform is captured, as shown in Figure 7.22.

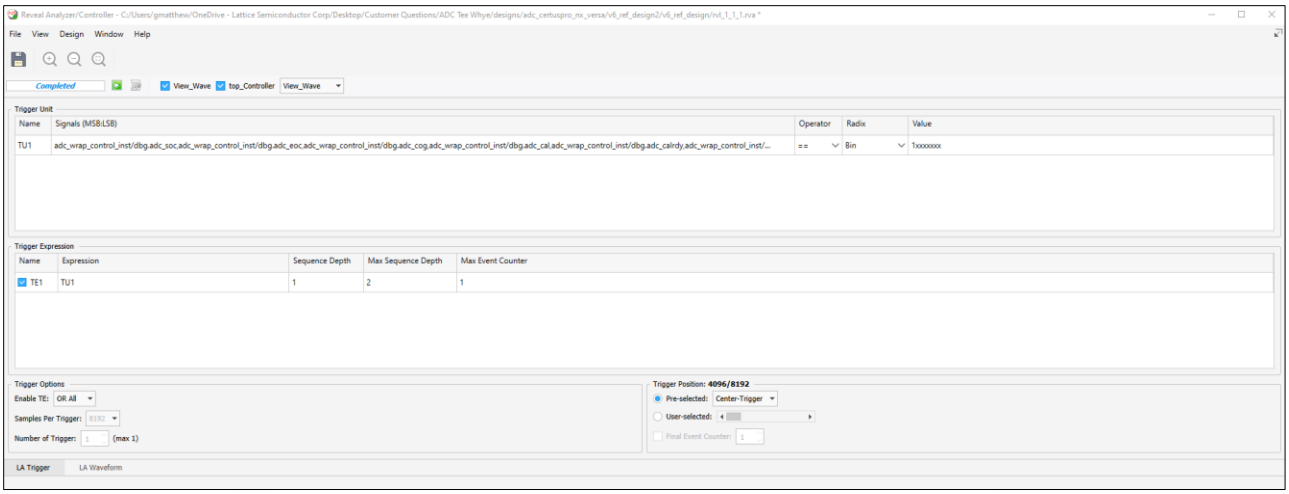

**Figure 7.21. Set up the Trigger**

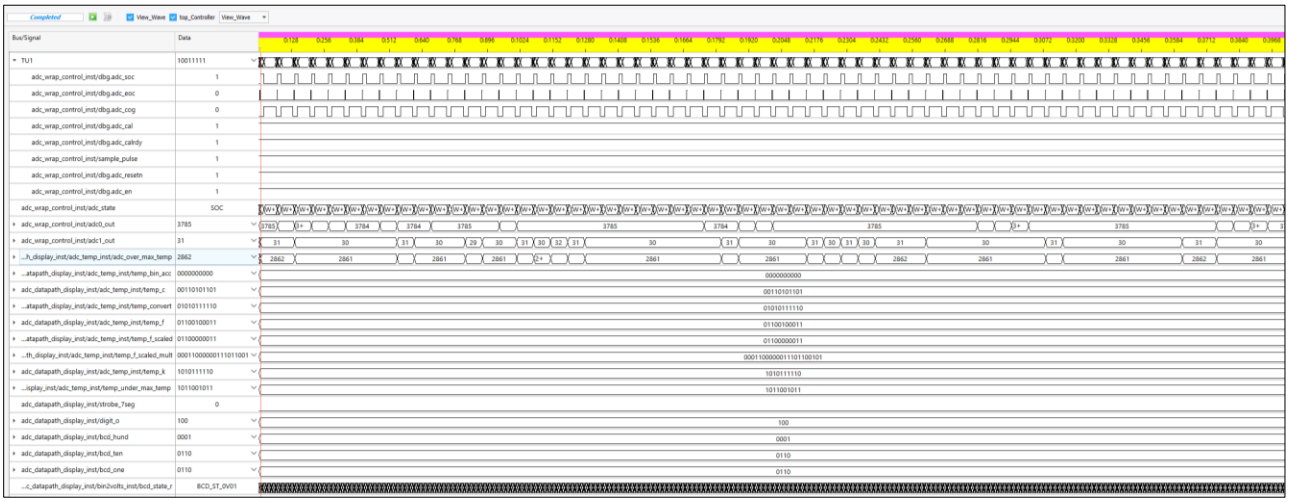

**Figure 7.22. Logic Analyzer Waveform**

## 7.8.2. Top Controller

Select the top controller instead of the logic analyzer from the drop-down menu. The top controller GUI shows (Figure 7.23). The upper section is for virtual LEDs, which are similar to board LEDs. Polling refers to how often the GUI checks the FPGA to find the current status of the LEDs. Move the slider to the middle and click Start Polling. By default, all LEDs are off.

The virtual switches are the equivalent of DIP switches on the board. Select **Direct Mode** so that when you move a switch, it is immediately sent to the board. If Direct Mode is not selected, you need to move the switches and then click the **Apply** button.

You can monitor the input voltages. For example, you can monitor VCC by using the SEL_VCC switch. The SEL_VCC LED illuminates and the 7-segment display displays about 1.0 V, as shown in Figure 7.24.

To measure temperature, turn off all other switches and turn on SEL_DTR. The LED SEL_DTR illuminates, indicating temperature measurement. The SEL_ADC1 LED also illuminates, as ADC1 is used to read the DTR. The board display gives a die temperature in Kelvin. Use the SEL_TEMP_C and SEL_TEMP_F to get temperatures in Celsius or Fahrenheit.

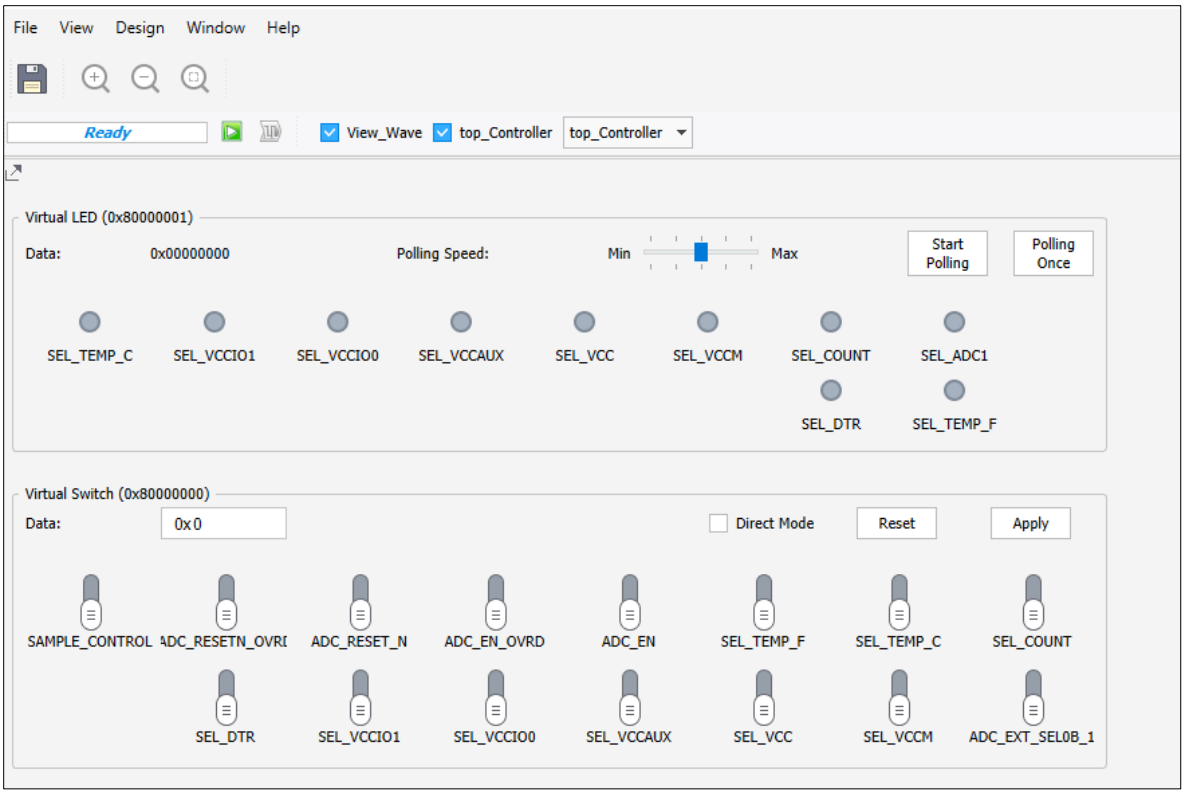


**Figure 7.23. Top Controller GUI**

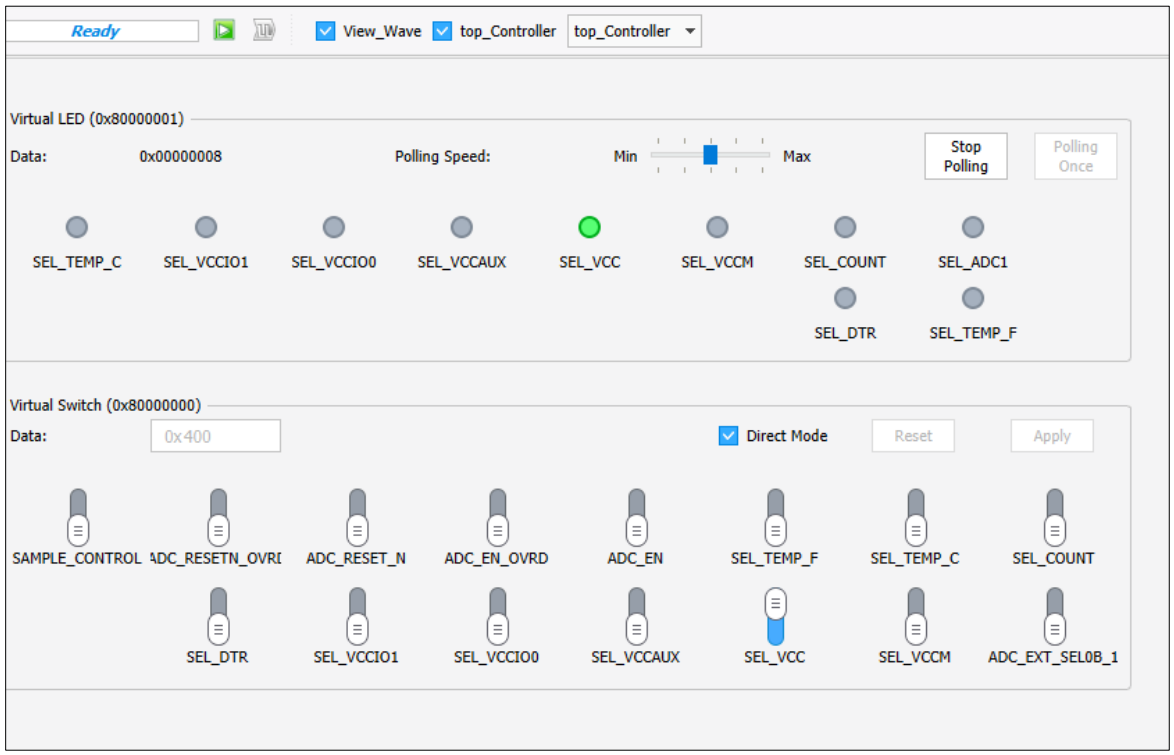


**Figure 7.24. Measure VCC**

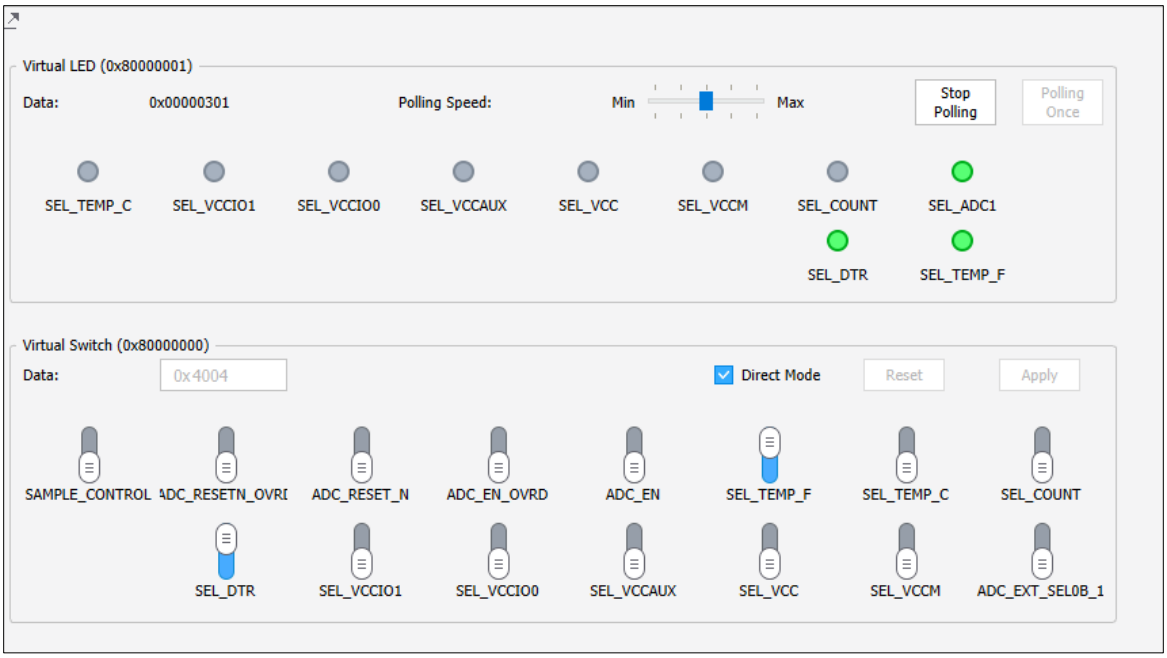


**Figure 7.25. Read Fahrenheit Temperature**

# 8. Summary

This basic demo design shows how to connect and sequence the ADC Core IP and how to process the results on the CertusPro-NX Versa Evaluation Board. The design includes the following input signals: a raw voltage on the dedicated input pins connected to a POT, internally divided down VCC voltages, and internal temperature values displayed in Kelvin, Celsius, or Fahrenheit. There are many variables in a design such as this, and the values used in this demo are just one example of many that could be used with the Nexus ADC IP. Different clock frequencies for fabric and ADC and delays in the sequencer can be verified in hardware by modifying this basic design.

# References

- ADC Demo Design web page
- ADC User Guide for Nexus Devices (FPGA-TN-02129)
- ADC Core Module User Guide (FPGA-IPUG-02168)
- CertusPro-NX Versa Board User Guide (FPGA-EB-02053)
- CertusPro-NX Data Sheet (FPGA-DS-02086)
- CertusPro-NX FPGA web page
- Lattice Radiant Software FPGA web page
- ADC Demo Design on CertusPro-NX Versa Board web page
- Debugging with Reveal Usage Guidelines and Tips (FPGA-AN-02060-1.1)
- Lattice Insights for Lattice Semiconductor training courses and learning plans

# Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

# Revision History

**Revision 1.2, August 2025**

| Section | Change Summary |
|---|---|
| Design Overview | In the ADC Control section:<br>• added description for the timeout mechanism;<br>• updated Figure 4.6. ADC Control FSM;<br>• added Figure 4.7. End Of Conversion Timer Running with No Error, Expected Behavior and Figure 4.8. End Of Conversion Timer Timeout and Giving Fresh SOC, Not Expected to Happen.<br>• for the sample_pulse description, changed *This is the max sampling frequency for the given adc_clk and fab_clk frequency but the sampling period can change slightly due to CDC synchronizers inside the IP taking between 3 and 3 clock cycles* to *This is the max sampling frequency for the given adc_clk and fab_clk frequency but the sampling period can change slightly due to CDC synchronizers inside the IP taking between 2 and 3 clock cycles*. |
| References | Added reference to the ADC Demo Design web page. |

**Revision 1.1, December 2024**

| Section | Change Summary |
|---|---|
| Inclusive Language | Added this section. |
| Introduction | Removed the original Figure 1.1 ADC Demo Running on CertusPro-NX Versa Evaluation Board. |
| Running the Demo | • Added *unlike earlier versions, this demo uses System Verilog and must be synthesized using Synplify under Radiant, not Lattice Synthesis Engine (LSE)*.<br>• Updated the summary of the contents of the ADC Demo project.<br>• Updated the description for J28 pins 1-2 from *external 1.8 V reference* to get *1.8 V reference from reference IC on board*.<br>• Updated the description for J29 pins 1-2 from *external 1.8 V reference* to *get 1.8 V reference from reference IC on board*.<br>• Updated Figure 2.1. CertusPro-NX Versa Board Powered on with Default Jumpers.<br>• Removed the original Table 2.1. DIP Switch Selection and ADC Results and related description. |
| Testing and Debugging | Added this section. |
| Design Overview | • Updated the section title from *Overview* to *Design Overview*.<br>• Updated Figure 4.1. ADC Demo Design Block Diagram.<br>• Added *The functions of the blocks are:*<br>*Clock and Reset —Generates fab_clk and adc_clk from a PLL which gets a reference clock from an oscillator. This block also generates sample_pulse if fixed frequency sampling is required.*<br>*ADC Wrap —Instantiates the ADC and the control FSM.*<br>*Data Path and Display —Scales the ADC output into volts or degrees of Kelvin, Fahrenheit, or Celsius.*<br>*LED Button Control —Debouces external switches, encodes Reveal top controller output into control signals, and drives LEDs.*<br>• Added the Hard IP Blocks Used and ADC Control sections. |
| Phase-Locked Loop IP | Removed this section. |
| ADC Core IP | Removed this section. |
| Data Path and Display Logic | Added this section. |
| Hex to Seven Segment Module | Removed this section. |
| ADC Binary to Volts Module | Removed this section. |
| ADC Binary to Temperature Module | Removed this section. |
| Connecting the Modules | Removed this section. |

| Section | Change Summary |
|---|---|
| ADC State Machine | Removed this section. |
| DIP Switch Debounce and Decode | Removed this section. |
| Reveal | Added this section. |
| References | Added the reference to *Debugging with Reveal Usage Guidelines and Tips (FPGA-AN-02060-1.1)* and *ADC Demo Design on CertusPro-NX Versa Board* web page. |

**Revision 1.0, November 2023**

| Section | Change Summary |
|---|---|
| All | Initial release. |


www.latticesemi.com