Cryptographic Engine (CRE) Demo for CertusPro-NX

User Guide

FPGA-UG-02193-0.80

September 2023

Disclaimers

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AES	Advanced Encryption Standard
APB	Advanced Peripheral Bus
CRE	Cryptographic Engine
LMMI	Lattice Memory Mapped Interface
SHA	Secure Hash Algorithm
SoC	System on a Chip
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

1. Introduction

This document provides instructions on how to set up and use the CRE Module IP with CertusPro™-NX devices. This document includes the following topics:

- Setting up the demo
- Programming the board
- Testing the design on the board with Lattice Propel™ software

CRE stands for Cryptographic Engine, which is a module built on the Security Hard IP. This module includes the following features:

- Supports the following user mode security features:
  - High throughput Secure Hash Algorithm – 256 bits (SHA 256)
  - Hash Based Message Authentication Code – HMAC-SHA
  - High throughput Advance Encryption Standard – 128/256 bits (AES-128/256)
- Supports True Random Number Generation
- Supports multiple bus interfaces:
  - Lattice Memory Mapped Interface (LMMI)
  - LMMI + FIFO (High-speed SHA/AES)
  - Advanced High-Performance Bus (AHB) – Lite
  - Advanced Peripheral Bus (APB)

1.1. Validated Configuration

- Board: This module functions on the CertusPro-NX Evaluation Board and is also compatible with the CertusPro-NX Versa Board and CertusPro-NX PCIE Bridge Board.
- Lattice Radiant™ Version: 2022.1.1.289.4
- Lattice Propel Version: 2022.1.2211240753

Table 1.1. IP Version

IP Name	Version
CRE	1.2.0
OSC for CRE	1.4.0
PLL	1.7.0
RISC-V MC	2.3.0
AHB Lite Interconnect	1.3.0
AHB Lite to APB Bridge	1.1.0
APB Interconnect	1.2.0
System_Memory	1.1.2
GPIO	1.6.0, replaceable with 1.6.1
APB Feedthrough	1.0.0
UART	1.3.0

2. Features

- Refer to CRE Module IP User Guide (FPGA-IPUG-02067) for supported features.
- Demonstrate the usage of the Secure Hash Algorithm – 256 bits (SHA-256).
- Demonstrate the usage of the Advance Encryption Standard – 128 Bits (AES-128).

3. Demo Requirements

3.1. Hardware Requirements

Hardware requirements for the demonstration:

- CertusPro-NX Evaluation Board or CertusPro-NX Versa Board
- USB connection for device programming
- USB connection for UART communication

3.2. Software Requirements

The following software are required for the steps outlined in this guide. Install the software in the order listed below:

1. Lattice Radiant v2022.1

https://www.latticesemi.com/Support/SoftwareArchive?ActiveTab=Downloadable+Software#_695D9FA3604D4CDA96EB88D98ADD31E2

2. Lattice Radiant v2022.1.1 – Service Pack

https://www.latticesemi.com/Support/SoftwareArchive?ActiveTab=Downloadable+Software#_695D9FA3604D4CDA96EB88D98ADD31E2

3. Lattice Radiant v2022.1.1 Control Pack CRE

Contact Technical Support for the control pack

4. Lattice Propel v2022.1

https://www.latticesemi.com/Support/SoftwareArchive?ActiveTab=Downloadable+Software#_695D9FA3604D4CDA96EB88D98ADD31E2

4. Setting Up Projects

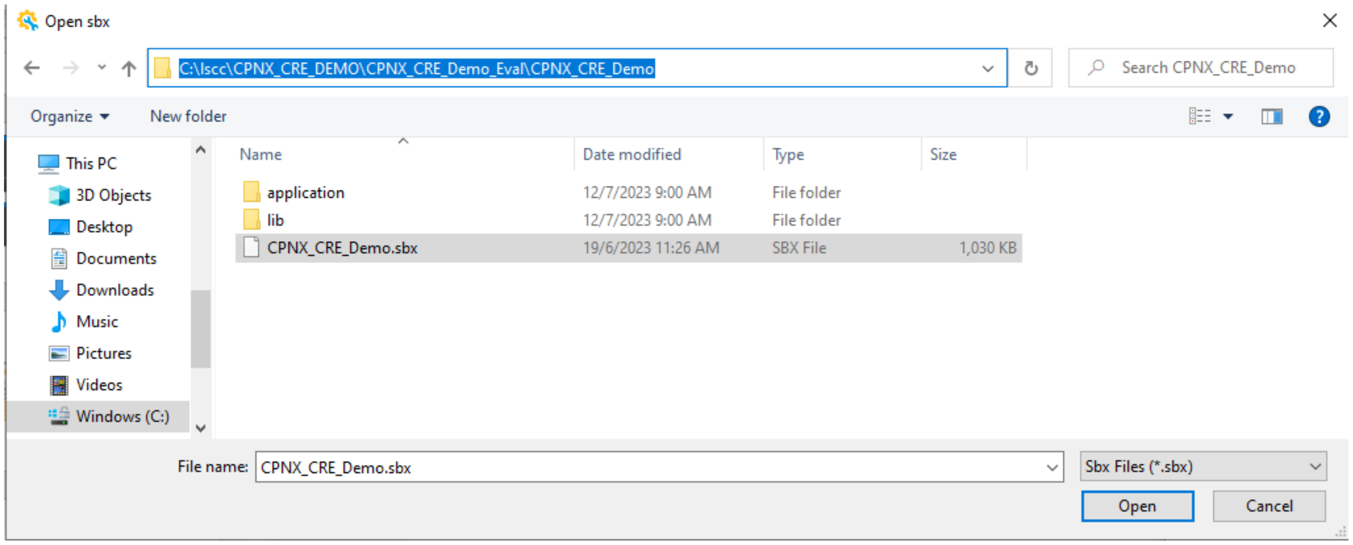
This section provides instructions on how to set up Lattice Propel software and Lattice Radiant software for Lattice CRE Demo for CertusPro-NX devices. A complete project contains three parts:

- Lattice Propel SoC project. The hardware portion of the project builds and generates the RISC-V SoC system that contains a CPU and peripherals that are instanced in the FPGA portion of CertusPro-NX using the Lattice Propel Builder.
- Lattice Radiant project. This portion of the project instantiates the CRE Module IP (only supported in Lattice Radiant software) and interfaces the CRE Module IP to the SoC system generated from Lattice Propel Builder. It is also used to synthesize, place and route, and generate the programming file to be loaded into the FPGA portion of CertusPro-NX devices.
- Lattice Propel RISC-V project. The software portion of the project is used to develop and test software codes running on the RISC-V processor using Lattice Propel software.

4.1. Importing an existing Lattice Propel SoC Project

To import the CRE Demo SoC project:

1. Unzip CPNX_CRE_DEMO.zip to C:\lscc.
2. In Lattice Propel Builder, import Demo SoC Project by clicking **File > Open Design > Open C:\lscc\CPNX_CRE_DEMO\CPNX_CRE_Demo_Eval\CPNX_CRE_Demo\CPNX_CRE_Demo.sbx.**



Name	Date modified	Type	Size
application	12/7/2023 9:00 AM	File folder	
lib	12/7/2023 9:00 AM	File folder	
CPNX_CRE_Demo.sbx	19/6/2023 11:26 AM	SBX File	1,030 KB

File name: CPNX_CRE_Demo.sbx
Sbx Files (*.sbx)

Figure 4.1. Importing a Lattice Propel SoC Project

This SoC project uses RISC-V MC "Hello World" template with an APB Feedthrough IP to export out the APB interface ports as CRE Module IP is only available in the Lattice Radiant software. Refer to Figure 4.2 and Table 4.1 for signals exported out from Lattice Propel SoC Project.

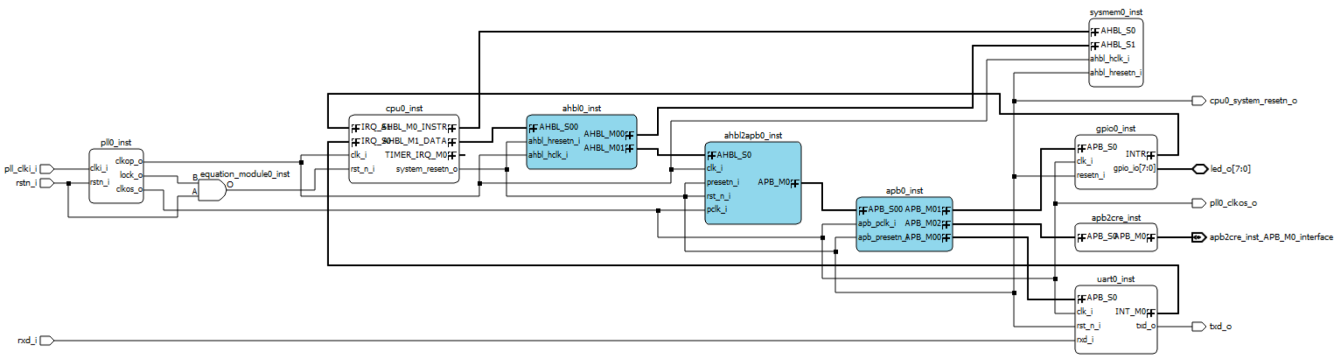
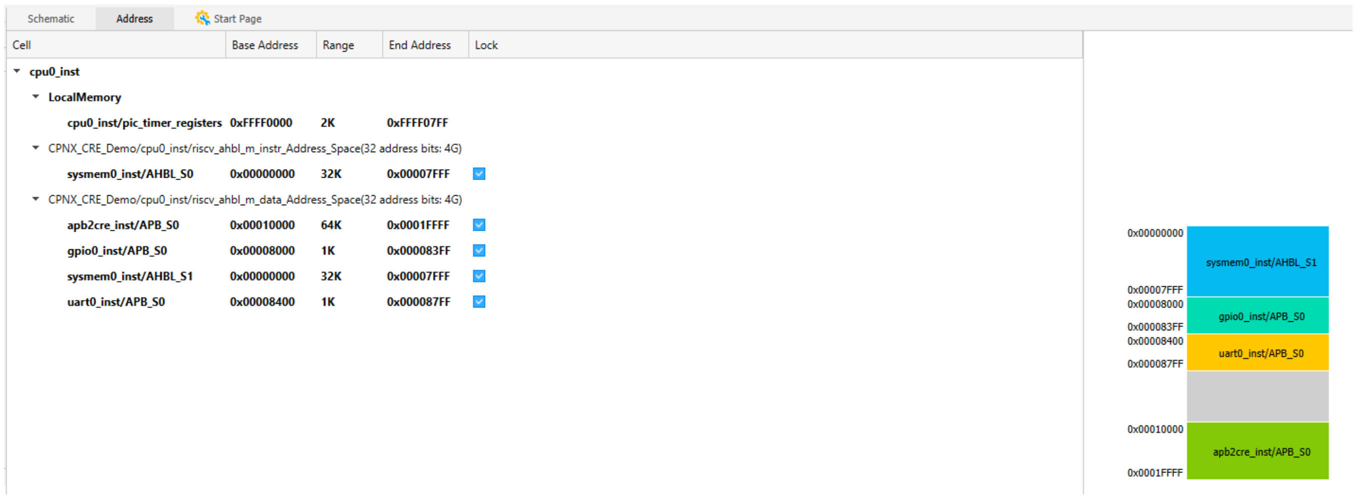


Figure 4.2. CRE Demo SoC Project Schematics

Table 4.1. Exported Signals

IP Name → Signal Name	Exported Signal Name	Direction	Description
PLL → clki_i	pll_clki_i	Input	Export signal using Create Port feature, to be connected to **OSC for CRE IP** in Lattice Radiant software.
PLL → clkos_o	pll0_clkos_o	Output	Export signal using Create Port feature, to be connected to **CRE Module IP** in Lattice Radiant software.
RISC-V MC → system_resetn_o	cpu0_system_resetn_o	Output	Export signal using Create Port feature, to be connected to **CRE Module IP** in Lattice Radiant software.
APB Feedthrough → APB_M0	apb2cre_inst_APB_M0_interface	Inout	Export APB interface using Export Interface feature, to be connected to **CRE Module IP**'s APB interface. Refer to Lattice-Propel-2022-1-Builder User-Guide (FPGA-UG-02177) for more information on the Export Interface.



Cell	Base Address	Range	End Address	Lock
cpu0_inst				
LocalMemory				
cpu0_inst/pic_timer_registers	0xFFFF0000	2K	0xFFFF07FF	
CPNX_CRE_Demo/cpu0_inst/riscv_ahbl_m_instr_Address_Space(32 address bits: 4G)				
sysmem0_inst/AHBL_S0	0x00000000	32K	0x00007FFF	✓
CPNX_CRE_Demo/cpu0_inst/riscv_ahbl_m_data_Address_Space(32 address bits: 4G)				
apb2cre_inst/APB_S0	0x00010000	64K	0x0001FFFF	✓
gpio0_inst/APB_S0	0x00008000	1K	0x000083FF	✓
sysmem0_inst/AHBL_S1	0x00000000	32K	0x00007FFF	✓
uart0_inst/APB_S0	0x00008400	1K	0x000087FF	✓

Figure 4.3. CRE Demo SoC Project Memory Address Mapping

The CRE Module IP in this demo SoC project is configured and accessed using the APB Interface. Figure 4.3 shows the memory address mapping for the SoC project, with the APB interface designated for controlling and accessing the CRE Module IP starts at address 0x10000.

3. Generate a Lattice Radiant Project. Refer to the Generating a Lattice Radiant Project section.

4.2. Generating a Lattice Radiant Project

To generate a Lattice Radiant project from the CRE Demo SoC Project:

1. Launch Lattice Radiant software from Lattice Propel Builder by clicking **Design > Run Radiant** or clicking the icon.

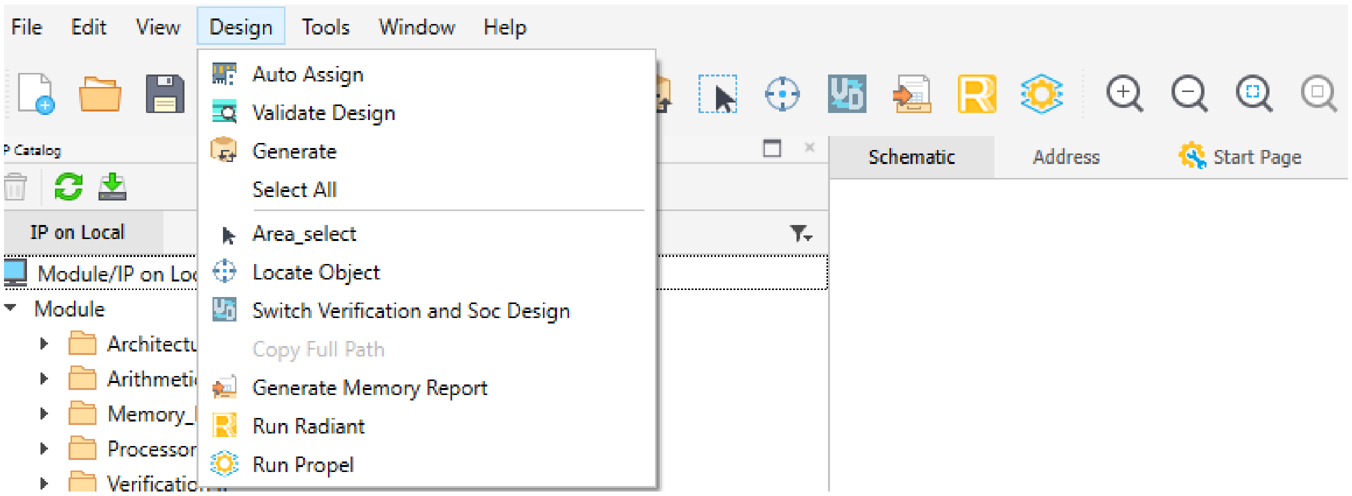


Figure 4.4. Launch Lattice Radiant from Lattice Propel Builder

2. In Lattice Radiant software, Navigate to File List Hierarchy > right-click on the original top-level unit **CPNX_CRE_Demo – CPNX_CRE_Demo.v** > click **Unset Top-Level Unit** > right-click **cre_wrapper.v** > click **Set as Top-Level Unit**.

Note: **cre_wrapper.v** is a module that is used to instantiate **OSC for CRE IP and CRE Module IP** that is only supported in Lattice Radiant software. Its purpose is to provide an interface to the original module generated by Lattice Propel Builder without requiring any modifications to the original module.

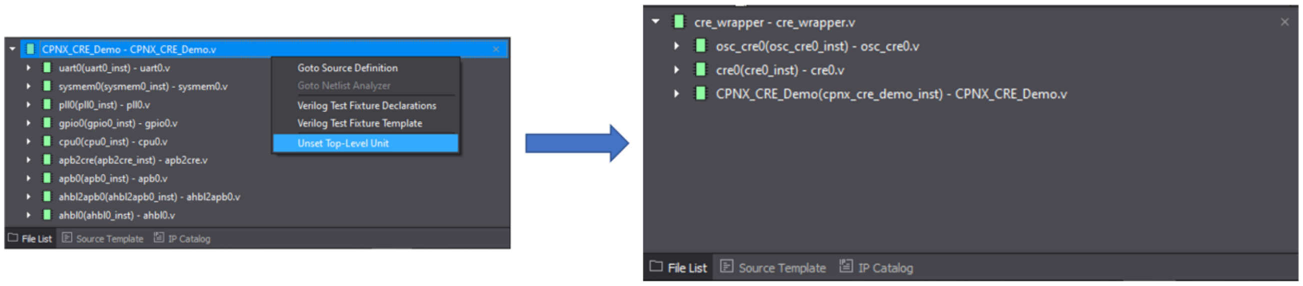

CPNX_CRE_Demo - CPNX_CRE_Demo.v
uart0(uart0_inst) - uart0.v
sysmem0(sysmem0_inst) - sysmem0.v
pll0(pll0_inst) - pll0.v
gpio0(gpio0_inst) - gpio0.v
cpu0(cpu0_inst) - cpu0.v
apb2cre(apb2cre_inst) - apb2cre.v
apb0(apb0_inst) - apb0.v
ahbl2apb0(ahbl2apb0_inst) - ahbl2apb0.v
ahbl0(ahbl0_inst) - ahbl0.v
Goto Source Definition
Goto Netlist Analyzer
Verilog Test Fixture Declarations
Verilog Test Fixture Template
Unset Top-Level Unit
File List
Source Template
IP Catalog
cre_wrapper - cre_wrapper.v
osc_cre0(osc_cre0_inst) - osc_cre0.v
cre0(cre0_inst) - cre0.v
CPNX_CRE_Demo(cpnx_cre_demo_inst) - CPNX_CRE_Demo.v
File List
Source Template
IP Catalog

Figure 4.5. Setting Top-Level Unit to "cre_wrapper.v"

3. Re-run synthesize flow.

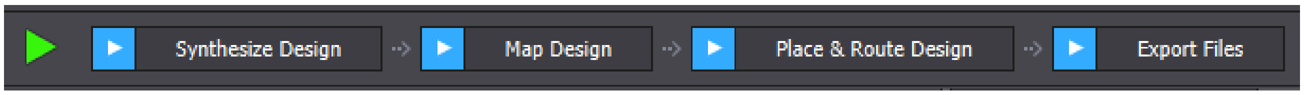


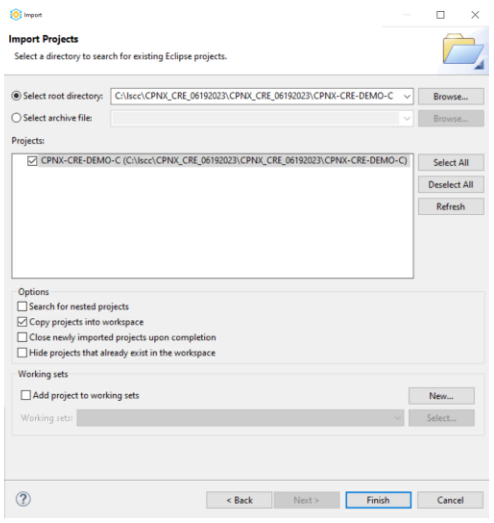
Figure 4.6. Lattice Radiant Project Compilation

Note: Repeat steps 2 and 3 if the top-level unit is not **cre_wrapper.v** after any subsequent launch of Lattice Radiant software from Lattice Propel Builder. To prevent the top-level unit from being overwritten, launch Lattice Radiant software separately.

4.3. Importing a Lattice Propel RISC-V Project

To import the CRE Demo RISC-V software project:

1. In Lattice Propel workspace, click File > Import.
2. Select Existing Projects into Workspace under the General group.
3. Click Next.
4. In **Import Projects** page, choose **Select Root Directory**, click **Browse** and select the folder **C:\lscc\CPNX_CRE_DEMO\CPNX-CRE-DEMO-C**. as shown in Figure 4.7.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: C:\lscc\CPNX_CRE_06192023\CPNX_CRE_06192023\CPNX-CRE-DEMO-C

( ) Select archive file:

Projects:

☑ CPNX-CRE-DEMO-C (C:\lscc\CPNX_CRE_06192023\CPNX_CRE_06192023\CPNX-CRE-DEMO-C)

Options

☐ Search for nested projects

☑ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

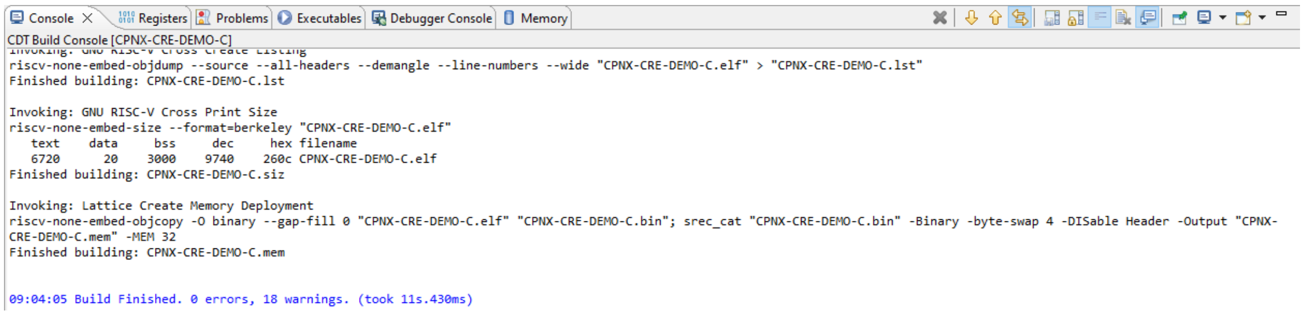
Working sets

☐ Add project to working sets

Working sets:

Figure 4.7. Importing a Lattice Propel Project

5. Under **Options**, select **Copy projects into workspace**. This provides a new copy of the source files that can be modified.
6. Click **Finish**. This Lattice Propel project contains the CRE Demo RISC-V software project, which includes the main test program, a simple driver and header file for CRE.
7. Right-click the RISC-V software project **CPNX-CRE-DEMO-C** in the Project Explorer tab and select Build Project.
   The console window at the bottom of the screen displays the output as shown in Figure 4.8. This indicates a successful compilation. There should be no error in the console window and warnings can be safely ignored.



CDT Build Console [CPNX-CRE-DEMO-C]

```
Invoking: GNU RISC-V Cross Create Listing
riscv-none-embed-objdump --source --all-headers --demangle --line-numbers --wide "CPNX-CRE-DEMO-C.elf" > "CPNX-CRE-DEMO-C.lst"
Finished building: CPNX-CRE-DEMO-C.lst

Invoking: GNU RISC-V Cross Print Size
riscv-none-embed-size --format=berkeley "CPNX-CRE-DEMO-C.elf"
   text    data     bss     dec     hex filename
   6720      20    3000    9740    260c CPNX-CRE-DEMO-C.elf
Finished building: CPNX-CRE-DEMO-C.siz

Invoking: Lattice Create Memory Deployment
riscv-none-embed-objcopy -O binary --gap-fill 0 "CPNX-CRE-DEMO-C.elf" "CPNX-CRE-DEMO-C.bin"; srec_cat "CPNX-CRE-DEMO-C.bin" -Binary -byte-swap 4 -DISable Header -Output "CPNX-
CRE-DEMO-C.mem" -MEM 32
Finished building: CPNX-CRE-DEMO-C.mem


09:04:05 Build Finished. 0 errors, 18 warnings. (took 11s.430ms)
```

Figure 4.8. Successful Project Compilation Message

8. Make sure the executable file **CPNX-CRE-DEMO-C.elf** is generated in **C:\lscc\CPNX_CRE_DEMO\CPNX-CRE-DEMO-C\Debug** to be used in Testing the design using Lattice Propel Debugger section. Debugger downloads the executable file (.elf) into the target processor for software execution.

4.4. Programming the Board

This section provides instructions on how to program the board with the .bit files generated from Lattice Radiant software.

1. Launch Programmer from Lattice Radiant software by clicking **Tools > Programmer**.

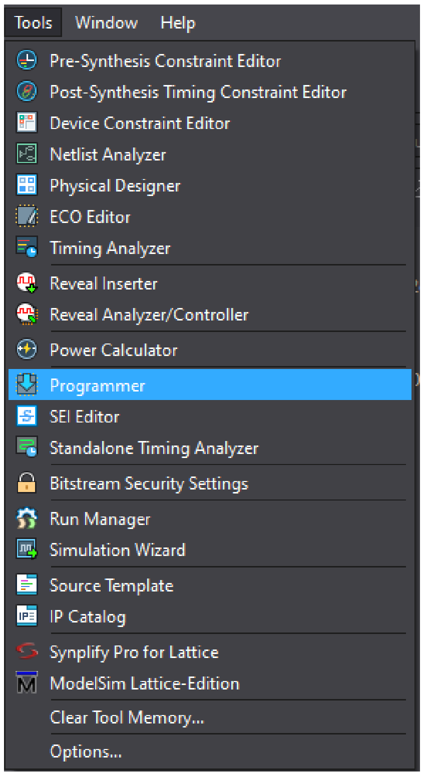
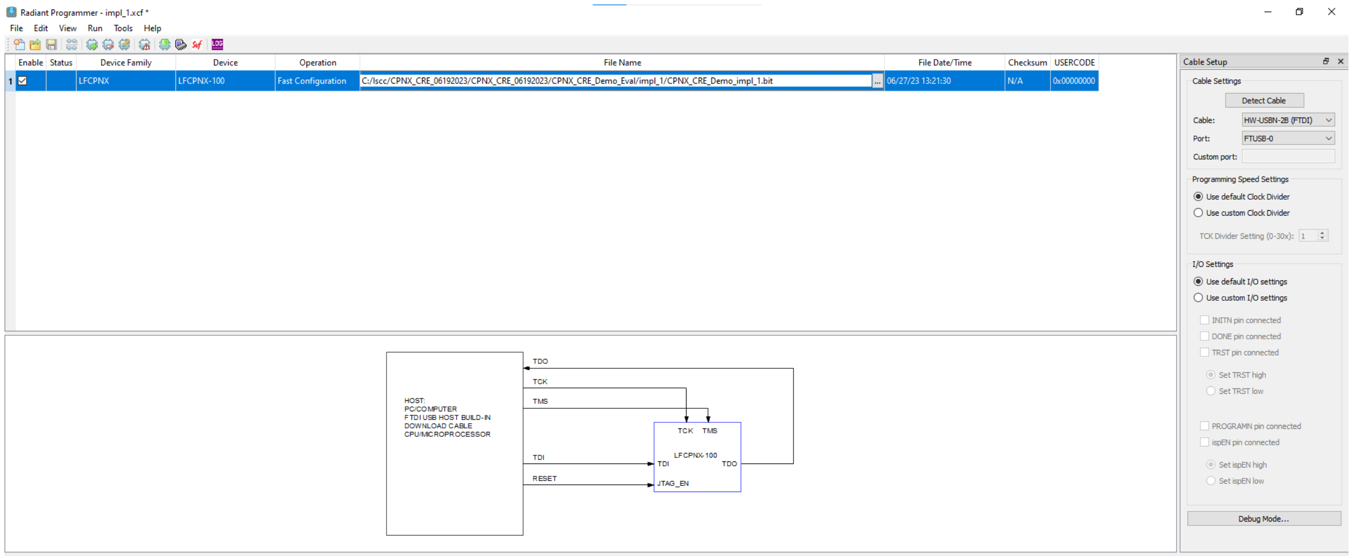


Figure 4.9. Launch Programmer

2. In the **Radiant Programmer** window, ensure the **Device Family**, **Device** and **Operation** matches the configuration shown in Figure 4.10. Select the .bit file generated from the Generating a Lattice Radiant Project section.



Radiant Programmer - impl_1.xcf *

	Enable	Status	Device Family	Device	Operation	File Name	File Date/Time	Checksum	USERCODE
1	✓		LFCPNX	LFCPNX-100	Fast Configuration	C:/lscc/CPNX_CRE_06192023/CPNX_CRE_06192023/CPNX_CRE_Demo_Eval/impl_1/CPNX_CRE_Demo_impl_1.bit	06/27/23 13:21:30	N/A	0x00000000

Cable Setup

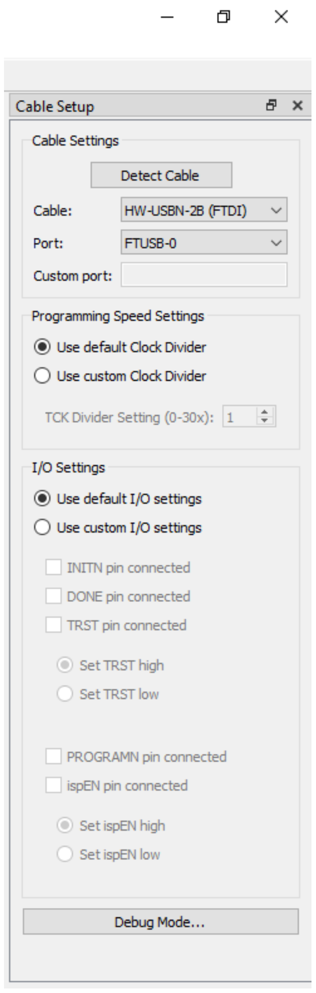
Cable Settings
Cable: HW-USBN-2B (FTDI)
Port: FTUSB-0
Custom port:

Programming Speed Settings
[✓ Use default Clock Divider] Use custom Clock Divider
TCK Divider Setting (0-30x): 1

I/O Settings
[✓ Use default I/O settings] Use custom I/O settings
INITN pin connected
DONE pin connected
TRST pin connected
[✓ Set TRST high] Set TRST low
PROGRAMN pin connected
ispEN pin connected
[✓ Set ispEN high] Set ispEN low

Figure 4.10. Radiant Programmer Configuration

3. In the **Cable Setup** window, click **Detect Cable**.



Cable Setup

Cable Settings

Cable: HW-USBN-2B (FTDI)

Port: FTUSB-0

Custom port:

Programming Speed Settings

(●) Use default Clock Divider
( ) Use custom Clock Divider

TCK Divider Setting (0-30x): 1

I/O Settings

(●) Use default I/O settings
( ) Use custom I/O settings

☐ INITN pin connected
☐ DONE pin connected
☐ TRST pin connected

(●) Set TRST high
( ) Set TRST low

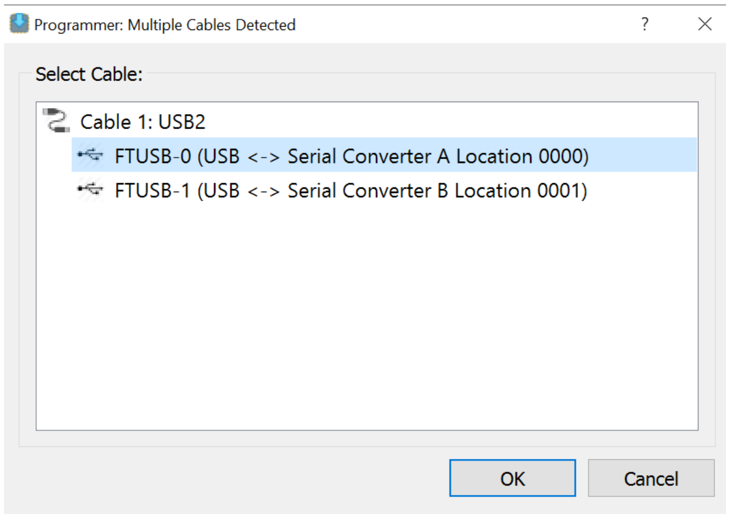
☐ PROGRAMN pin connected
☐ ispEN pin connected

(●) Set ispEN high
( ) Set ispEN low

Figure 4.11. Cable Setup

4. In the **Select Cable** window, select the **FTUSB-0** option as shown in Figure 4.12 and click **OK**.

Note: For CertusPro-NX boards, FTUSB-0 is connected to JTAG port. Refer to CertusPro-NX board schematic or diagram for more information.



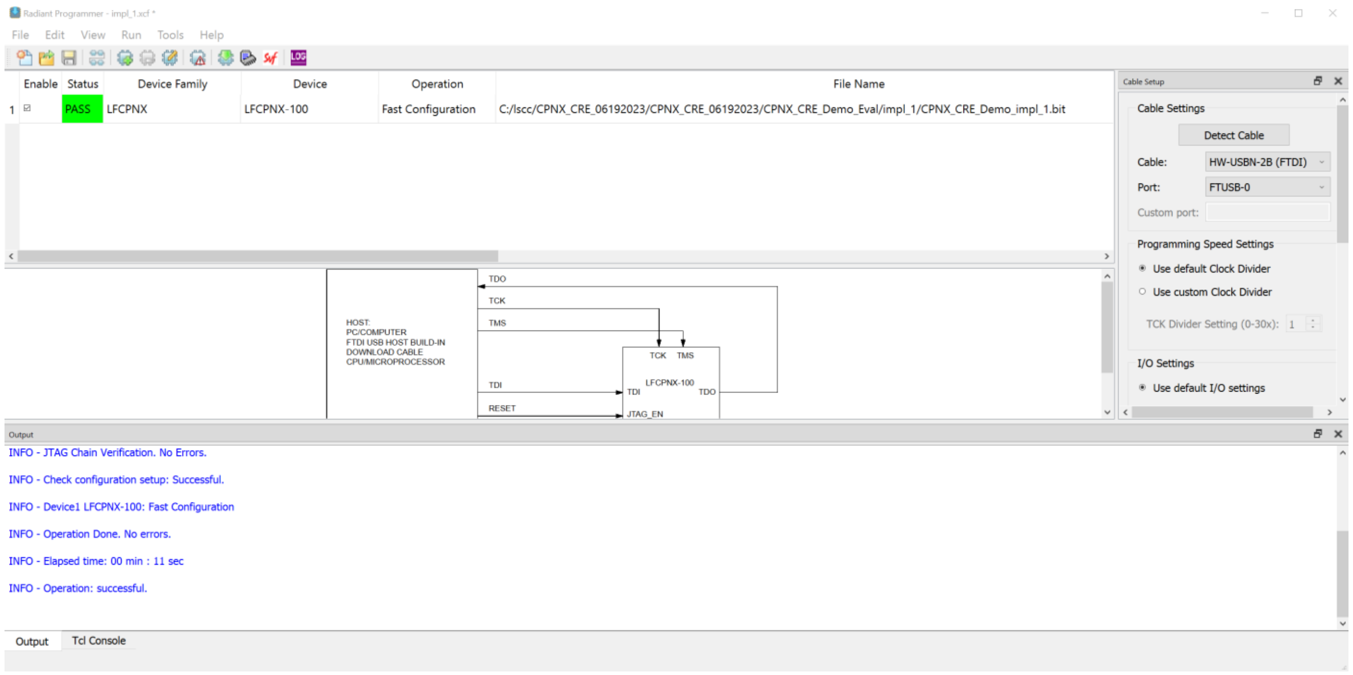
Programmer: Multiple Cables Detected

Select Cable:

Cable 1: USB2
FTUSB-0 (USB <-> Serial Converter A Location 0000)
FTUSB-1 (USB <-> Serial Converter B Location 0001)

Figure 4.12. Selecting Cable

5. Click Program Device. **Status** should display **PASS** and the console window at the bottom of the screen should display the logs as shown in Figure 4.13. This indicates that the configuration is successful.


Radiant Programmer - impl_1.xcf *

	Enable	Status	Device Family	Device	Operation	File Name
1	☑	PASS	LFCPNX	LFCPNX-100	Fast Configuration	C:/lscc/CPNX_CRE_06192023/CPNX_CRE_06192023/CPNX_CRE_Demo_Eval/impl_1/CPNX_CRE_Demo_impl_1.bit

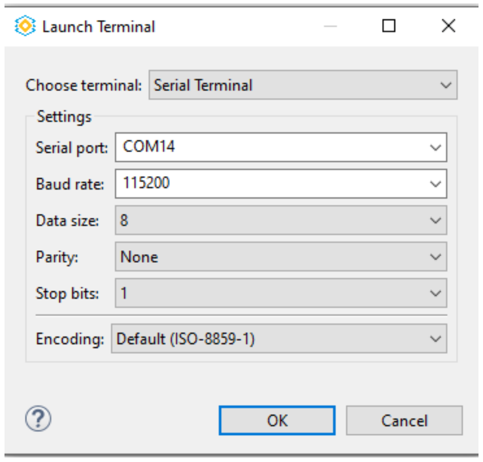
Cable Setup
Cable Settings
Cable: HW-USBN-2B (FTDI)
Port: FTUSB-0
Custom port:
Programming Speed Settings
● Use default Clock Divider
○ Use custom Clock Divider
TCK Divider Setting (0-30x): 1
I/O Settings
● Use default I/O settings

```
INFO - JTAG Chain Verification. No Errors.
INFO - Check configuration setup: Successful.
INFO - Device1 LFCPNX-100: Fast Configuration
INFO - Operation Done. No errors.
INFO - Elapsed time: 00 min : 11 sec
INFO - Operation: successful.
```

Figure 4.13. Programming the Device

4.5. Testing the design using Lattice Propel Debugger

1. In the **Terminal** window, click **Open a Terminal**, this opens the **Launch Terminal** window.
2. In the Launch Terminal window, make the following selections.
    a. Choose terminal: **Serial Terminal**
    b. Serial Port: **COMXX**, choose the larger number. For example, if you see COM1 and COM2, choose COM2.
    Note: **COMXX** is the port on the laptop that represent UART. Refer to CertusPro-NX board schematic or diagram for more information.
    c. Baud rate: **115200**


Launch Terminal

Choose terminal: Serial Terminal
Settings
Serial port: COM14
Baud rate: 115200
Data size: 8
Parity: None
Stop bits: 1
Encoding: Default (ISO-8859-1)

Figure 4.14. Launch Terminal Window

3. Click **OK.**
4. In Lattice Propel software, click **Run > Debug Configurations**

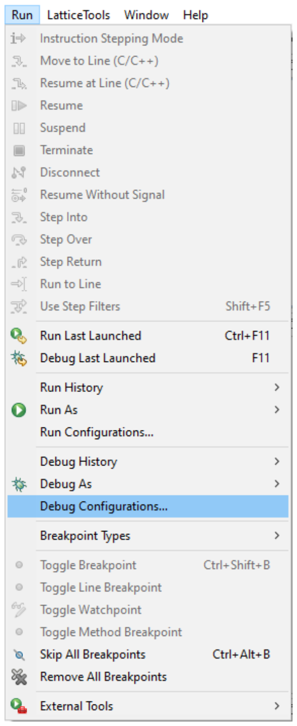
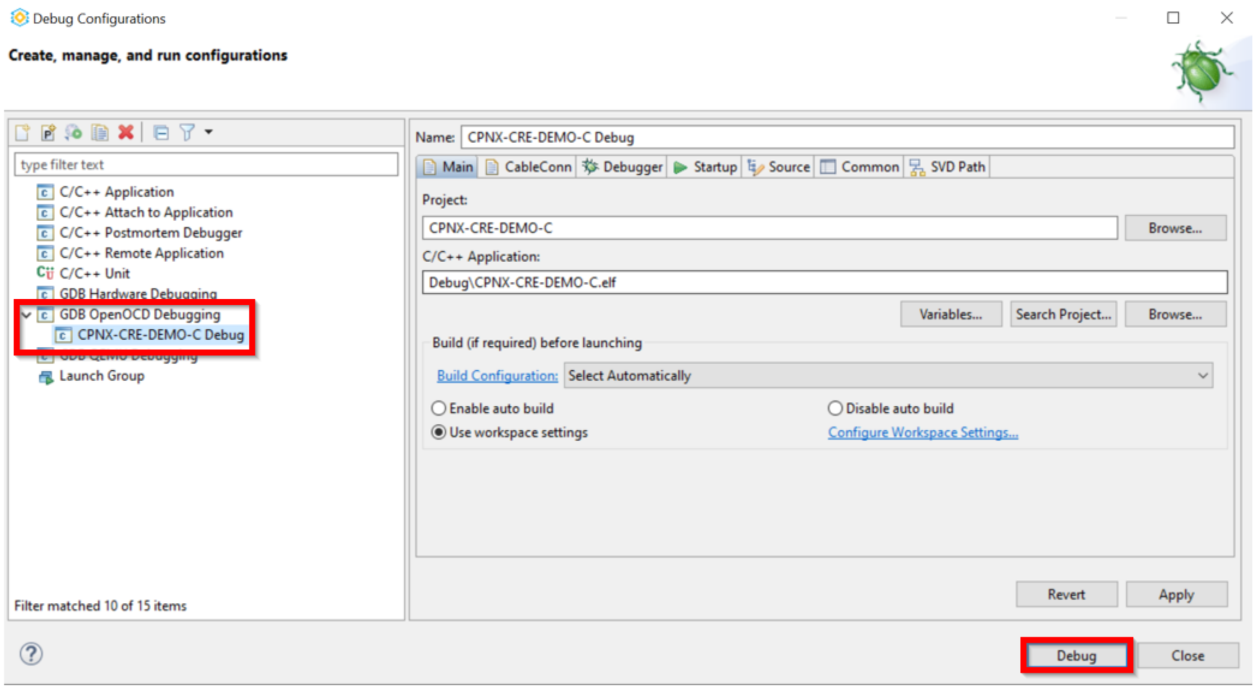


Figure 4.15. Launching Debug Configurations

5. In the **Debug Configurations** window, select and double-click GDB OpenOCD Debugging. A window should appear on the right that automatically populates the fields as shown in Figure 4.15.


Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- C/C++ Unit
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - CPNX-CRE-DEMO-C Debug
- GDB QEMU Debugging
- Launch Group

Filter matched 10 of 15 items

Name: CPNX-CRE-DEMO-C Debug

Project: CPNX-CRE-DEMO-C

C/C++ Application: Debug\CPNX-CRE-DEMO-C.elf

Build (if required) before launching

Build Configuration: Select Automatically

Enable auto build

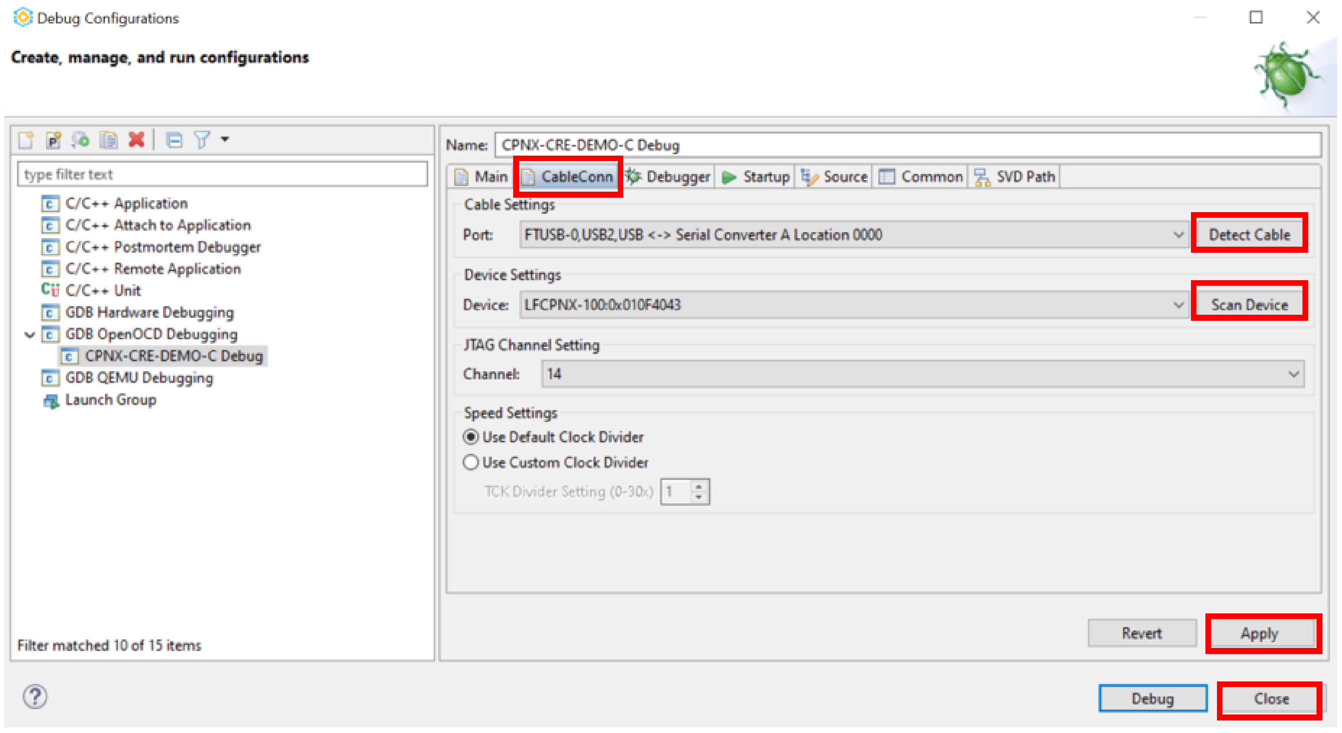
Disable auto build

Use workspace settings [✓]

Configure Workspace Settings...

Figure 4.16. GDB OpenOCD Debugging Window

6. Clickon the **CableConn** tab.
7. Click **Detect Cable** and **Scan Device**. This automatically detects the FTUSB cable port and device. If not, check to ensure the connections and configurations of the board are set up properly and retry.


Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- C/C++ Unit
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - CPNX-CRE-DEMO-C Debug
- GDB QEMU Debugging
- Launch Group

Filter matched 10 of 15 items

Name: CPNX-CRE-DEMO-C Debug

Cable Settings

Port: FTUSB-0,USB2,USB <-> Serial Converter A Location 0000

Device Settings

Device: LFCPNX-100:0x010F4043

JTAG Channel Setting

Channel: 14

Speed Settings

Use Default Clock Divider [✓]

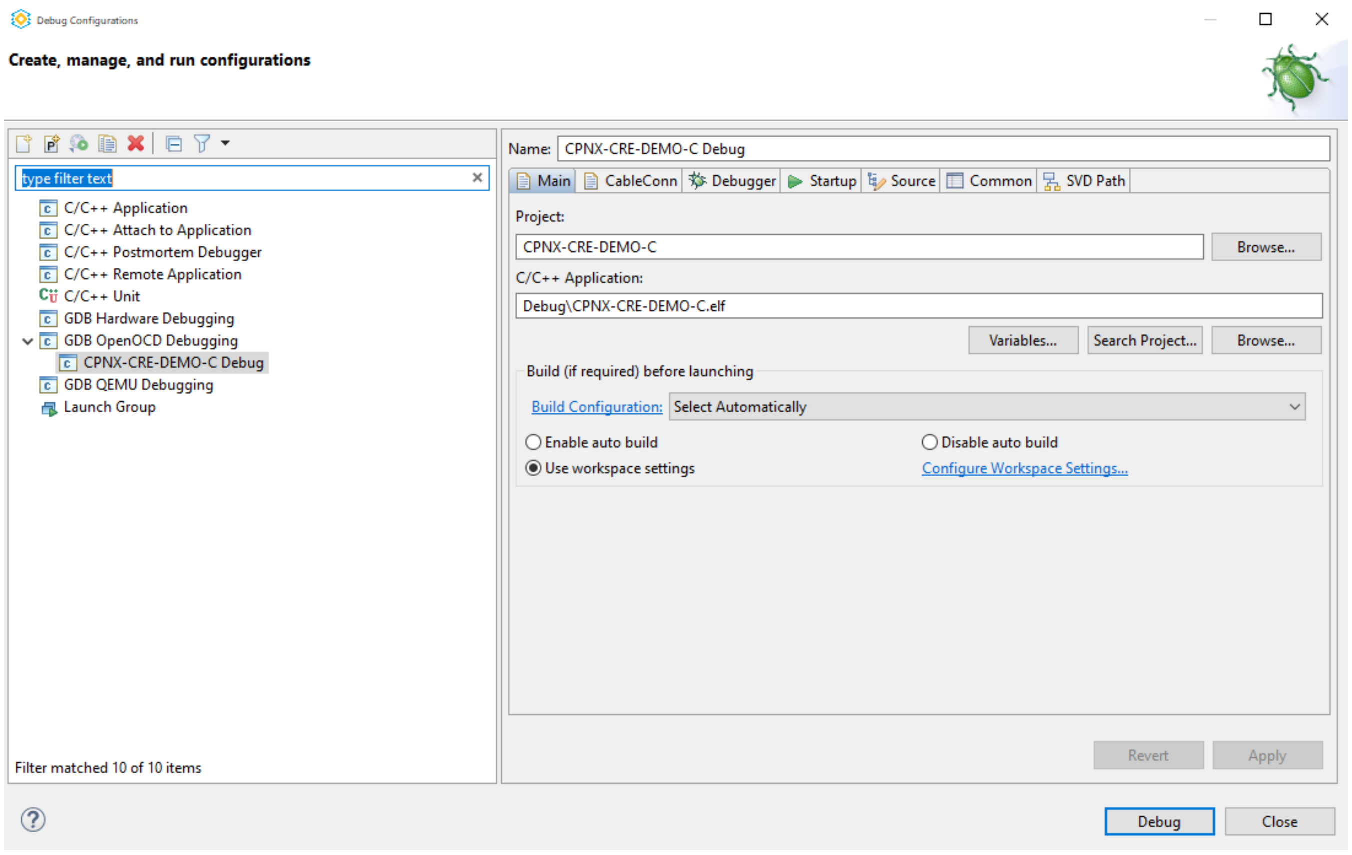
Use Custom Clock Divider

TCK Divider Setting (0-30x) 1

Figure 4.17. Debugger Cable and Device Settings

8. Click **Apply** > **Close**. This creates a debug configuration for the user which can be used to launch the program.

9. In the Propel main window, look for **Terminal** window. If not available, click **Window > Show View > Terminal**. This displays the output of the program through UART.
10. In the **Debug Configuration** window, select **CPNX-CRE-DEMO-C Debug** and click **Debug**.



Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- C/C++ Unit
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - CPNX-CRE-DEMO-C Debug
- GDB QEMU Debugging
- Launch Group

Filter matched 10 of 10 items

Name: CPNX-CRE-DEMO-C Debug

Project: CPNX-CRE-DEMO-C

C/C++ Application: Debug\CPNX-CRE-DEMO-C.elf

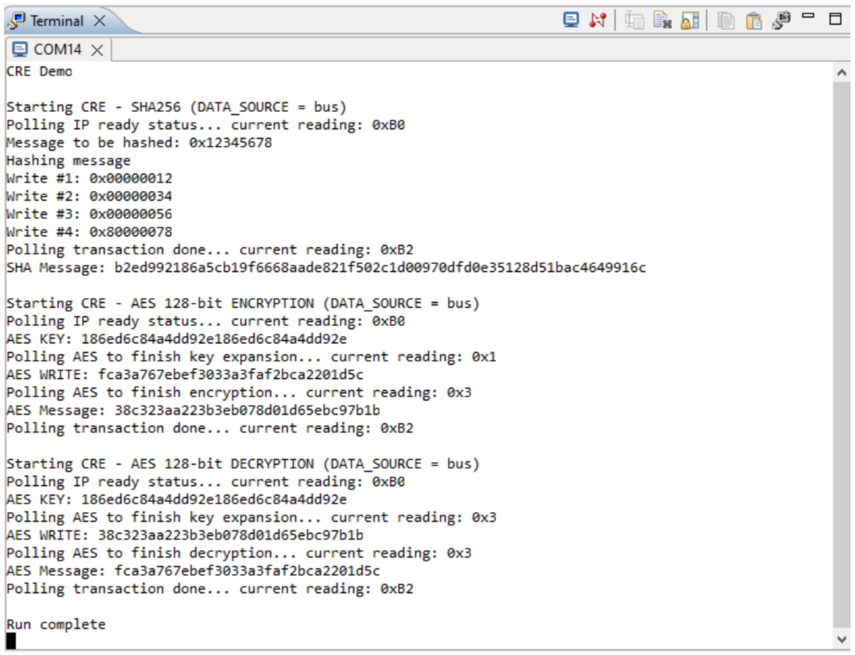
Build (if required) before launching

Build Configuration: Select Automatically

Enable auto build / [✓ Use workspace settings] / Disable auto build

Figure 4.18. Run Debugger

11. In the Propel SDK window, you can see messages from the demo design in the Serial Terminal.



```
CRE Demo

Starting CRE - SHA256 (DATA_SOURCE = bus)
Polling IP ready status... current reading: 0xB0
Message to be hashed: 0x12345678
Hashing message
Write #1: 0x00000012
Write #2: 0x00000034
Write #3: 0x00000056
Write #4: 0x80000078
Polling transaction done... current reading: 0xB2
SHA Message: b2ed992186a5cb19f6668aade821f502c1d00970dfd0e35128d51bac4649916c

Starting CRE - AES 128-bit ENCRYPTION (DATA_SOURCE = bus)
Polling IP ready status... current reading: 0xB0
AES KEY: 186ed6c84a4dd92e186ed6c84a4dd92e
Polling AES to finish key expansion... current reading: 0x1
AES WRITE: fca3a767ebef3033a3faf2bca2201d5c
Polling AES to finish encryption... current reading: 0x3
AES Message: 38c323aa223b3eb078d01d65ebc97b1b
Polling transaction done... current reading: 0xB2

Starting CRE - AES 128-bit DECRYPTION (DATA_SOURCE = bus)
Polling IP ready status... current reading: 0xB0
AES KEY: 186ed6c84a4dd92e186ed6c84a4dd92e
Polling AES to finish key expansion... current reading: 0x3
AES WRITE: 38c323aa223b3eb078d01d65ebc97b1b
Polling AES to finish decryption... current reading: 0x3
AES Message: fca3a767ebef3033a3faf2bca2201d5c
Polling transaction done... current reading: 0xB2

Run complete
```

Figure 4.19. Demo Design Message

These messages show the demo design flow, demonstrating the usage of the SHA-256 encryption, AES-128 encryption, and AES-128 decryption. The demo software does not print messages for all the procedures involved to serial terminal. For the complete procedure and description, refer to CRE Module IP User Guide (FPGA-IPUG-02067).

For SHA-256 encryption, the demo starts by polling the CRE IP ready status, then the SHA data source is configured to be from APB bus before initializing and starting SHA engine. Message to be hashed with SHA-256 will then be written 1B at a time. The CRE IP will then be polled for the transaction done status before the encrypted SHA message is read and displayed the in serial terminal.

For AES-128 encryption, the demo starts by polling the CRE IP ready status, then AES data source is configured to be from APB bus. Next, the CRE IP is configured with the AES encryption mode and the AES key size is set to 128 bits. The AES key is then written to memory address (0x2810) before the demo polls AES to finish key expansion and start the AES engine. The AES message to be encrypted will then be written into the IP. Afterward, the CRE IP will be polled for the AES finish encryption status. When encryption is done, the encrypted message will be read and displayed in the serial terminal.

For AES-128 decryption, the demo starts by polling the CRE IP ready status. The AES data source is then configured to be from APB bus. Next, the CRE IP is configured with the AES decryption mode and the AES key size is set to 128 bits. The AES key is then written to memory address (0x2810) before the demo polls AES to finish key expansion and start the AES engine.

Note that if AES encryption/decryption is being run back-to-back, value in memory address for AES STAT [1] will still be 1 and will not reset, hence the poll value will be 0x3. The AES message to be decrypted will then be written into the IP. Afterwards, the CRE IP will be polled for AES finish decryption status. Once decryption is done, the message is then read and displayed in the serial terminal.

References

For more information, refer to:

- CRE Module IP User Guide (FPGA-IPUG-02067)
- CertusPro-NX FPGA website
- Lattice Propel Design Environment website
- Lattice Radiant FPGA design software
- Lattice Diamond FPGA design software
- 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 1.0, September 2023

Section	Change Summary
All	Initial release.