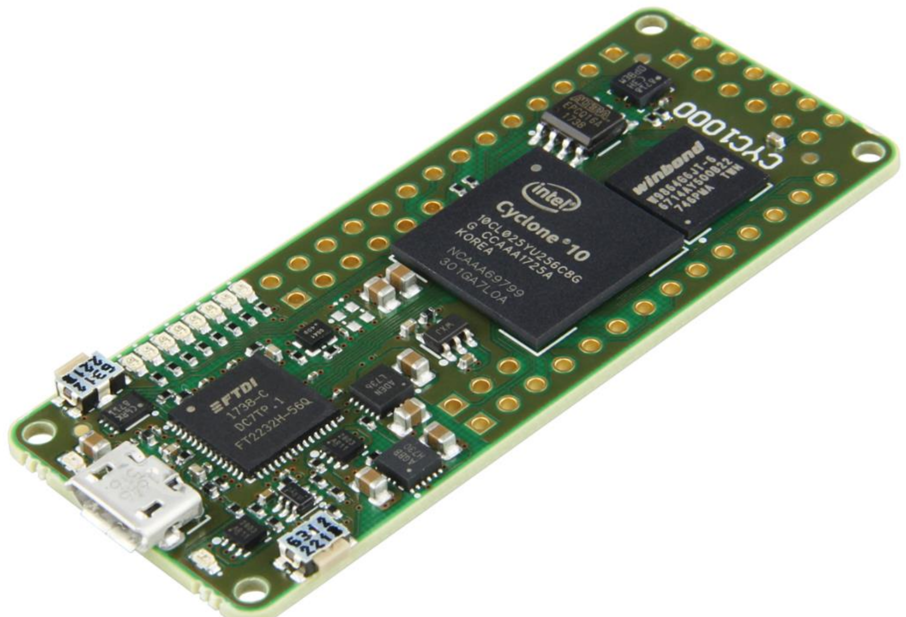
CYC1000 User Guide



Please read the legal disclaimer at the end of this document.

Revision 1.0

Table of Contents

Table of Figures

Chapter 1 - CYC1000 IoT / Maker Board

1.1 About Arrow CYC1000 Board

The CYC1000 is a customizable IoT / Maker Board ready for evaluation, development and/or use in a product. It is built around the Intel Cyclone 10 LP FPGA, which is optimized for low-cost and low-power, making them ideal for high-volume and cost-sensitive applications. High density sea of programmable gates and on-board resources allow implementation of Nios II 32-bit microcontroller IP, which provides the ideal solution for I/O expansion, chip-to-chip interfacing, industrial, automotive and consumer applications.

The CYC1000 is equipped with an Arrow USB Programmer2, SDRAM, flash memory, accelerometer sensor and PMOD/ADRUINO MKR connectors making it a fully featured plug and play solution without any additional costs.

The CYC1000 board contains all the tools needed to use the board in conjunction with a computer that runs a 64-bit Linux / Microsoft Windows 7 operating system or later.

1.2 Useful Links

A set of useful links that can be used to get relevant information about the CYC1000 or the Cyclone 10 LP FPGA.

- CYC1000 at Arrow Shop
- CYC1000 at Trenz Electronic Shop
- Intel Cyclone 10 LP Webpage
- CYC1000 Wiki Page

1.3 Getting Help

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

- **Arrow Electronics**

  In Person
  Arrow EMEA
  + 49 (0) 6102 5030 0

  Online
  https://arrow.com

- **Trenz Electronic GmbH**

  https://www.trenz-electronic.de/en/

Chapter 2 - Introduction to the CYC1000 Board

2.1 Layout and Components

Figure 1 shows a top view of the board. It depicts the layout of the board and indicates the location of the various connectors and key components.

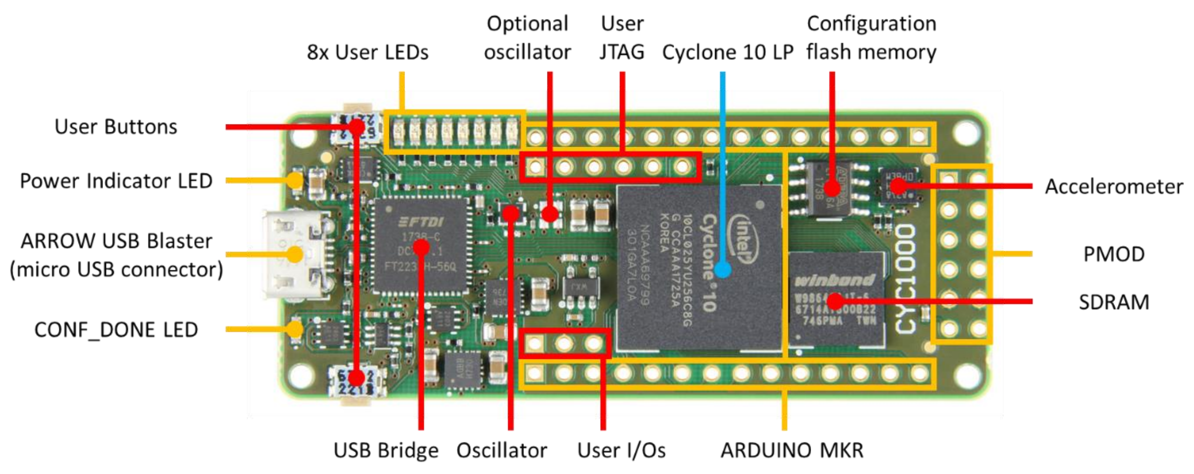


Figure 1 - CYC1000 Board (top view)

The following are available on the CYC1000 board:

- Intel Cyclone 10 LP 10CL025YU256C8G device
- Arrow USB Programmer2 on-board for programming; JTAG Mode
- 64MBit SDRAM 166MHz
- 16Mbit serial configuration flash memory
- 12MHz MEMS Oscillator
- One optional MEMS Oscillator of preferred frequency
- 8x red user LEDs
- 2x board indication LEDs
- 2x user push buttons
- 3-axis accelerometer
- 12-pin PMOD header
- Arduino MKR header
- User JTAG header
- User I/O header

2.2 Block Diagram

Figure 2 represents the block diagram of the board. All the connections are established through the Cyclone 10 LP FPGA device to provide maximum flexibility for users. Users can configure the FPGA to implement any system design.

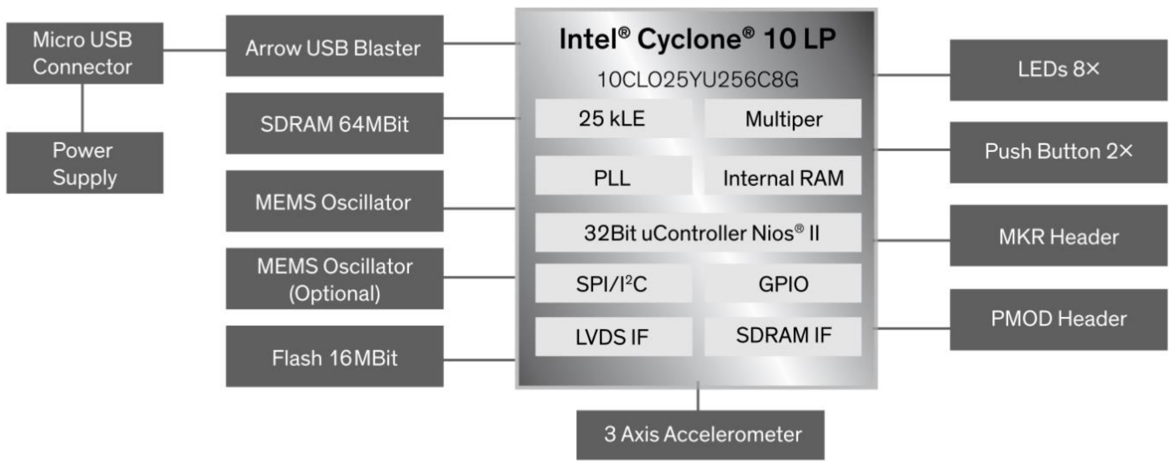


Figure 2 - CYC1000 Block Diagram

FPGA Device

- Available Cyclone 10 LP Devices for the CYC1000

Resources	Device			
	10CL006	10CL010	10CL016	10CL025
Logic Elements (LE)	6,272	10,320	15,408	24,624
M9K Memory (Kb)	270	414	504	594
18 x 18 Multiplier	15	23	56	66
PLLs	2	2	4	4

Configuration and Debug

- On-board Arrow USB Programmer2 (mini-USB type B connector)

Memory Devices

- 4MBit to 32MBit external flash memory
- 64MBit to 256MBit external SDRAM memory

Headers

- PMOD Header
- Arduino MKR Header
- User JTAG Header
- User I/O Header

Buttons and Indicators

- 2x side-buttons
- 8x red user LEDs
- 2x board indication LEDs

Sensors

- One 3-axis accelerometer

Power

Recommended external supply voltage range:	+5.0 V (nominal)
Recommended I/O signal voltage range:	0 to +3.3 V

Chapter 3 - Connections and Peripherals of the CYC1000 Board

3.1 Board Status Elements

In addition to the 8 LEDs that the FPGA can control, there are 2 additional LEDs which can indicate the status of the board.

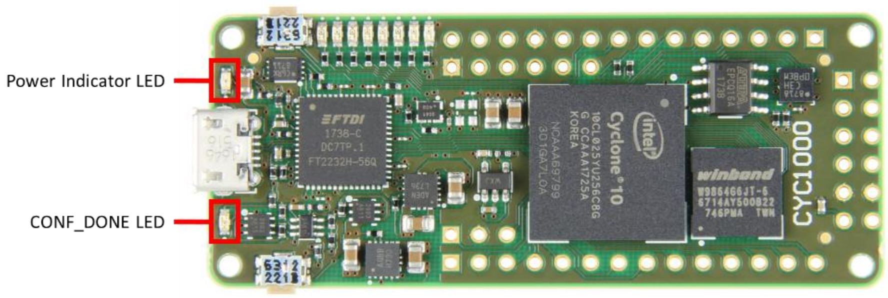


Figure 3 – Position of Indication LEDs

Board Reference	LED Name	Description
D1	3.3V	On when 3.3V power is active
D10	CONF_DONE	On when configuration data was loaded to Cyclone 10 LP device without error

3.2 Clock Circuitry

All the external clocks of the system can be seen in Figure 4. The default clock (CLK12M) is at 12MHz and is connected and driving the FPGA's user logic and the Arrow USB Programmer2. There is an optional slot of another clock (CLK_X) to add another preferred clock source to the FPGA. Both clocks are driving the internal PLLs.

For more information on clocks and PLLs of the Cyclone 10 LP, please refer to this document.

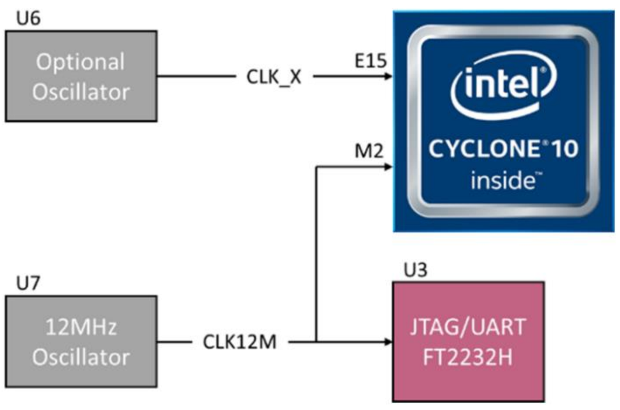


Figure 4 – CYC1000 Clock Tree

Board Reference	FPGA Pin No.	Description	I/O Standard
CLK12M	PIN_M2	12MHz clock input	3.3 V
CLK_X	PIN_E15	Optional clock input	3.3 V

3.3 Peripherals Connected to the FPGA

3.3.1 LEDs

There are eight red user-controllable LEDs connected to the FPGA. Each LED is driven directly and individually by the Cyclone 10 LP FPGA; driving its associated pin to a high logic level for on or low logic level for off.

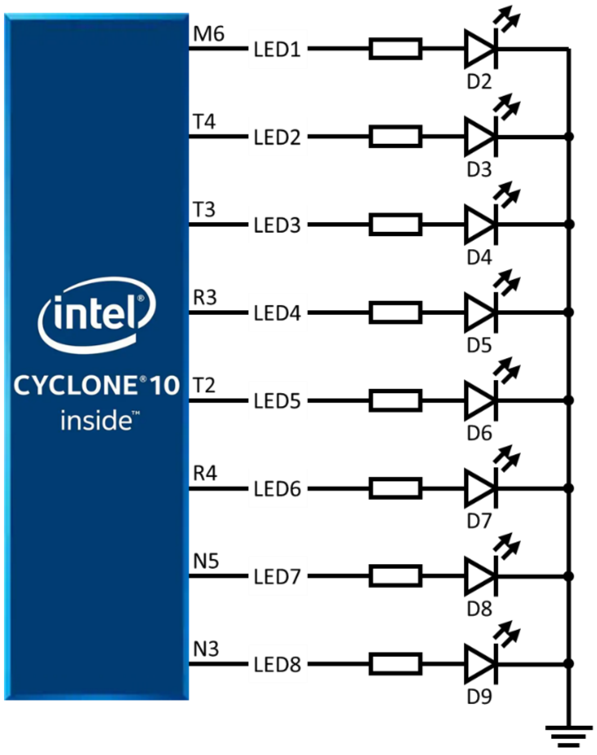


Figure 5 – LED Connections

Board Reference	FPGA Pin No.	I/O Standard
LED1	PIN_M6	3.3 V
LED2	PIN_T4	3.3 V
LED3	PIN_T3	3.3 V
LED4	PIN_R3	3.3 V
LED5	PIN_T2	3.3 V
LED6	PIN_R4	3.3 V
LED7	PIN_N5	3.3 V
LED8	PIN_N3	3.3 V

3.3.2 Push Buttons

The board has two push buttons connected to the FPGA. Push buttons drive their associated pins low logic level when pressed and high logic level when released.

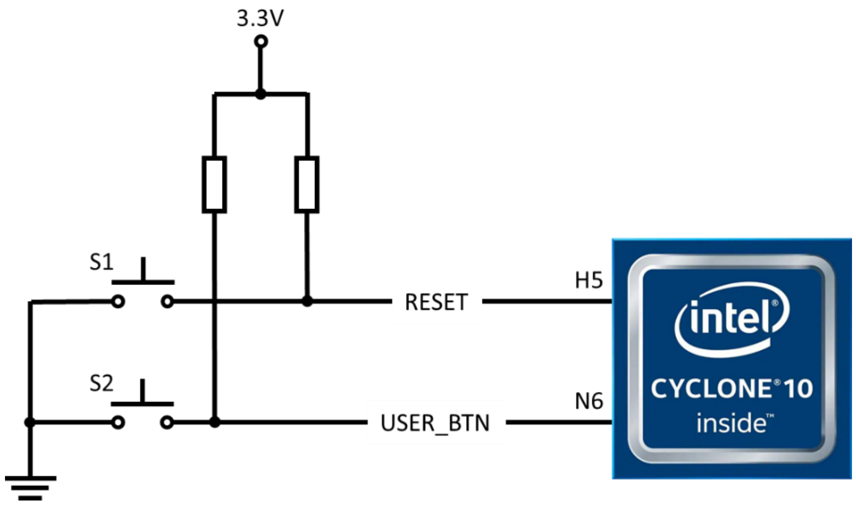


Figure 6 – Button Connections

Board Reference	FPGA Pin No.	Description	I/O Standard
RESET	PIN_H5	nCONFIG	3.3 V
USER_BTN	PIN_N6	User button	3.3 V

3.3.3 Accelerometer

The board comes with a digital accelerometer (LIS3DH), commonly known as the G-Sensor. This G-Sensor is a small, thin, ultra-low power consumption, 3-axis accelerometer with digital I2C/SPI serial interface, standard output. The LIS3DH has user-selectable full scales of +/-2g, +/-4g, +/-8g, +/-16g and it is capable of measuring accelerations with output data rates from 1 Hz to 5 kHz. The supplied power to the board (coming either from micro-USB connection or user V_{in}) can be monitored through the ADC channel 3 of the accelerometer.

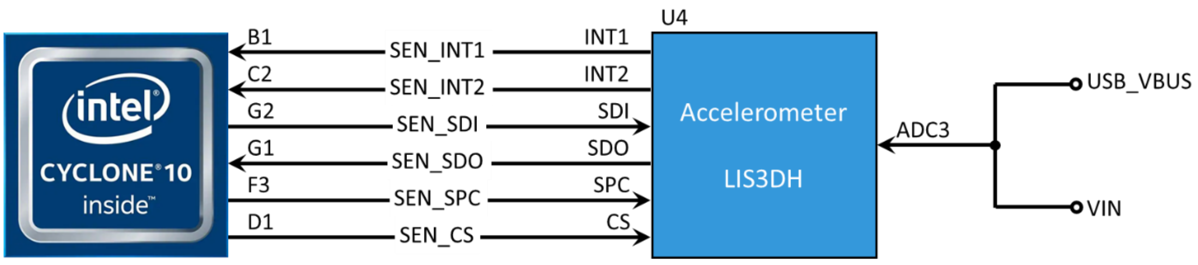


Figure 7 – Accelerometer Connections

Board Reference	FPGA Pin No.	Description	I/O Standard
SEN_INT1	PIN_B1	Interrupt 1	3.3 V
SEN_INT2	PIN_C2	Interrupt 2	3.3 V
SEN_SDI	PIN_G2	Data In (MOSI)*	3.3 V
SEN_SDO	PIN_G1	Data Out (MISO)*	3.3 V
SEN_SPC	PIN_F3	Clock*	3.3 V
SEN_CS	PIN_D1	Chip Select*	3.3 V

**For SPI connection*

3.3.4 SDRAM Memory

The CYC1000 board supports 64MBit (default version) or up to 256MBit (customized version) SDRAM which can operate up to 166 MHz clock frequency. Below are the connections and pinning of the SDRAM used in the CYCX1000.

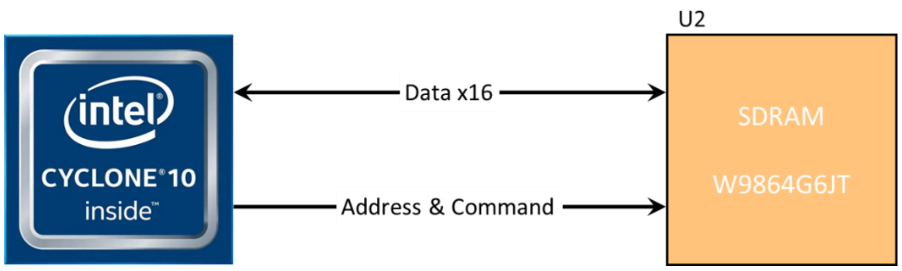


Figure 8 – SDRAM Connections

Board Reference	FPGA Pin No.	Description	I/O Standard
A0	PIN_A3	SDRAM Address [0]	3.3 V
A1	PIN_B5	SDRAM Address [1]	3.3 V
A2	PIN_B4	SDRAM Address [2]	3.3 V
A3	PIN_B3	SDRAM Address [3]	3.3 V
A4	PIN_C3	SDRAM Address [4]	3.3 V
A5	PIN_D3	SDRAM Address [5]	3.3 V
A6	PIN_E6	SDRAM Address [6]	3.3 V
A7	PIN_E7	SDRAM Address [7]	3.3 V
A8	PIN_D6	SDRAM Address [8]	3.3 V
A9	PIN_D8	SDRAM Address [9]	3.3 V
A10	PIN_A5	SDRAM Address [10]	3.3 V
A11	PIN_E8	SDRAM Address [11]	3.3 V
A12	PIN_A2	SDRAM Address [12]	3.3 V
A13	PIN_C6	SDRAM Address [13]	3.3 V
BA0	PIN_A4	SDRAM Bank Address [0]	3.3 V
BA1	PIN_B6	SDRAM Bank Address [1]	3.3 V
RAS	PIN_B7	SDRAM Row Address Strobe	3.3 V
CAS	PIN_C8	SDRAM Column Address Strobe	3.3 V
WE	PIN_A7	SDRAM Write Enable	3.3 V
CS	PIN_A6	SDRAM Chip Select	3.3 V

Board Reference	FPGA Pin No.	Description	I/O Standard
CLK	PIN_B14	SDRAM Input Clock	3.3 V
CKE	PIN_F8	SDRAM Clock Enable	3.3 V
DQ0	PIN_B10	SDRAM Data [0]	3.3 V
DQ1	PIN_A10	SDRAM Data [1]	3.3 V
DQ2	PIN_B11	SDRAM Data [2]	3.3 V
DQ3	PIN_A11	SDRAM Data [3]	3.3 V
DQ4	PIN_A12	SDRAM Data [4]	3.3 V
DQ5	PIN_D9	SDRAM Data [5]	3.3 V
DQ6	PIN_B12	SDRAM Data [6]	3.3 V
DQ7	PIN_C9	SDRAM Data [7]	3.3 V
DQ8	PIN_D11	SDRAM Data [8]	3.3 V
DQ9	PIN_E11	SDRAM Data [9]	3.3 V
DQ10	PIN_A15	SDRAM Data [10]	3.3 V
DQ11	PIN_E9	SDRAM Data [11]	3.3 V
DQ12	PIN_D14	SDRAM Data [12]	3.3 V
DQ13	PIN_F9	SDRAM Data [13]	3.3 V
DQ14	PIN_C14	SDRAM Data [14]	3.3 V
DQ15	PIN_A14	SDRAM Data [15]	3.3 V
DQM0	PIN_B13	SDRAM Lower Data Mask	3.3 V
DQM1	PIN_D12	SDRAM Upper Data Mask	3.3 V

3.3.5 Serial Configuration Flash Memory

The CYC1000 board supports up to 32MBit of serial flash memory that can be used for user data and programming non-volatile storage. The configuration bit stream is downloaded into the serial configuration device which automatically loads the configuration data into the Cyclone 10 LP when the board is powered on. Device memory capacity not consumed storing configuration data can be used as general-purpose non-volatile memory, which is perfect for program and data storage. Several interface peripherals available with Nios II embedded processors allow you to access the serial configuration device as a memory module connected to your embedded system.

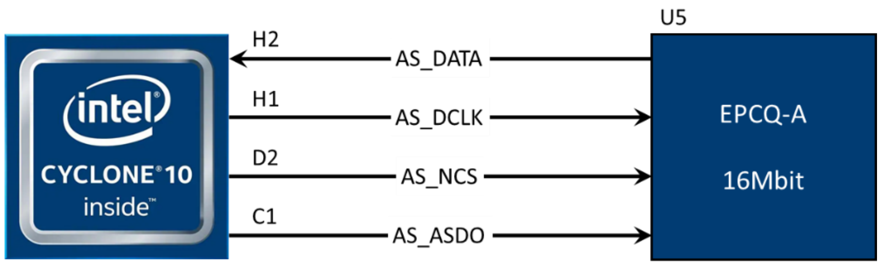


Figure 9 – Flash Connections

Board Reference	FPGA Pin No.	Description	I/O Standard
AS_DATA	PIN_H2	Data In	3.3 V
AS_DCLK	PIN_H1	Clock	3.3 V
AS_NCS	PIN_D2	Chip Select	3.3 V
AS_ASDO	PIN_C1	Data Out	3.3 V

3.3.6 Arduino MKR Connectors

The CYC1000 board offers connectivity to Arduino MKR compatible shields that could also alternatively be used as GPIOs. The MKR connectors offer up to 23 digital I/Os.

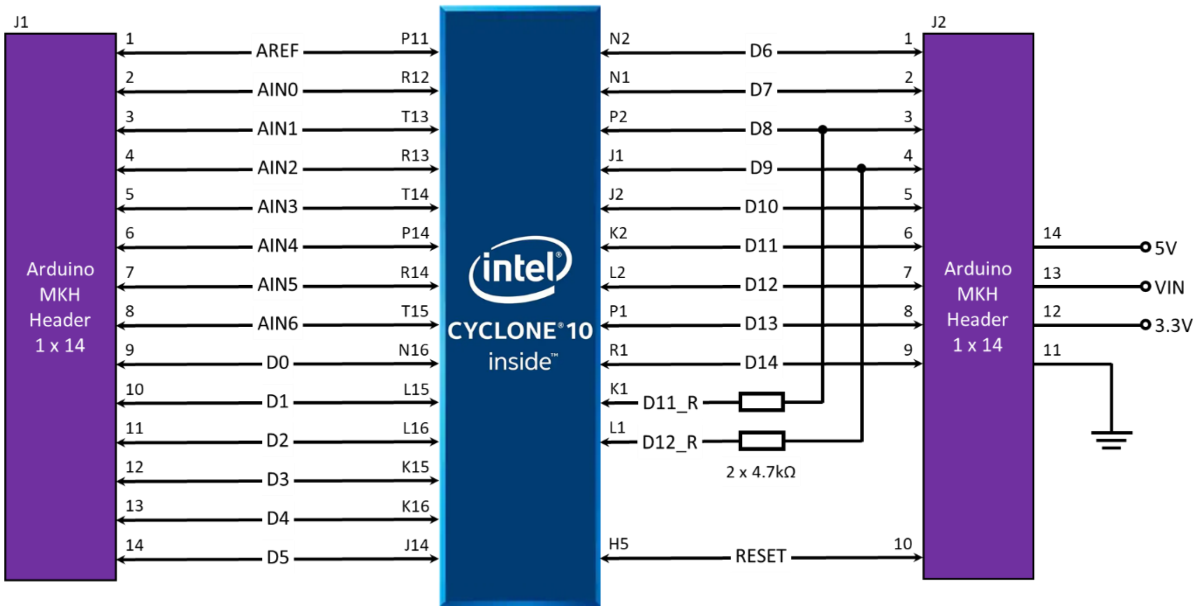


Figure 10 - Arduino MKR Header Connections

Board Reference	FPGA Pin No.	MKR Header	Description	I/O Standard
AREF	PIN_P11	J1/1	Input reference voltage or GPIO	3.3 V
AIN0	PIN_R12	J1/2	GPIO [0]	3.3 V
AIN1	PIN_T13	J1/3	GPIO [1]	3.3 V
AIN2	PIN_R13	J1/4	GPIO [2]	3.3 V
AIN3	PIN_T14	J1/5	GPIO [3]	3.3 V
AIN4	PIN_P14	J1/6	GPIO [4]	3.3 V
AIN5	PIN_R14	J1/7	GPIO [5]	3.3 V
AIN6	PIN_T15	J1/8	GPIO [6]	3.3 V
D0	PIN_N16	J1/9	Digital In [0]	3.3 V
D1	PIN_L15	J1/10	Digital In [1]	3.3 V
D2	PIN_L16	J1/11	Digital In [2]	3.3 V
D3	PIN_K15	J1/12	Digital In [3]	3.3 V
D4	PIN_K16	J1/13	Digital In [4]	3.3 V
D5	PIN_J14	J1/14	Digital In [5]	3.3 V
D6	PIN_N2	J2/1	Digital In [6]	3.3 V
D7	PIN_N1	J2/2	Digital In [7]	3.3 V

Board Reference	FPGA Pin No.	MKR Header	Description	I/O Standard
D8	PIN_P2	J2/3	Digital In [8]	3.3 V
D9	PIN_J1	J2/4	Digital In [9]	3.3 V
D10	PIN_J2	J2/5	Digital In [10]	3.3 V
D11	PIN_K2	J2/6	Digital In [11]*	3.3V
D12	PIN_L2	J2/7	Digital In [12]*	3.3 V
D13	PIN_P1	J2/8	Digital In [13]	3.3 V
D14	PIN_R1	J2/9	Digital In [14]	3.3 V
D11_R	PIN_K1	J2/6	Digital In [11] with resistor*	3.3 V
D12_R	PIN_L1	J2/7	Digital In [12] with resistor*	3.3 V
RESET	PIN_H5	J2/10	System reset of the board	3.3 V
GND	N/A	J2/11	Ground output to the connector	N/A
3.3V	N/A	J2/12	3.3V power to the connector	N/A
VIN	N/A	J2/13	User power into to the CYC1000	N/A
5V	N/A	J2/14	5V power to the connector	N/A

**Can only choose one, hence same name pinning*

3.3.7 PMOD Connector

The CYC1000 board offers connectivity to PMOD compatible connectors (2x6-pin or 1x12-pin), making it possible to add a big variety of sensors or ICs to the system. Below is the connection schematic and pinning information.

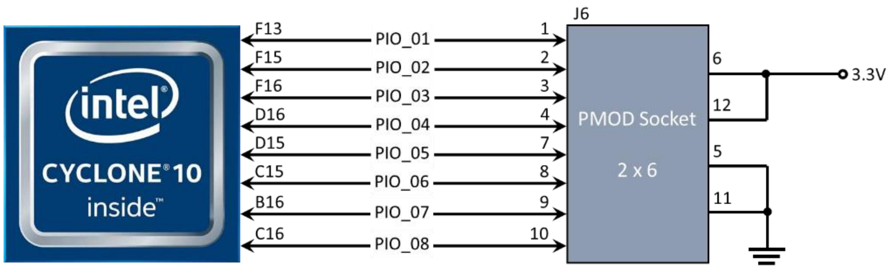


Figure 11 – PMOD Header Connections

Board Reference	FPGA Pin No.	Description	I/O Standard
PIO_01	PIN_F13	PMOD Pin 1	3.3 V
PIO_02	PIN_F15	PMOD Pin 2	3.3 V
PIO_03	PIN_F16	PMOD Pin 3	3.3 V
PIO_04	PIN_D16	PMOD Pin 4	3.3 V
PIO_05	PIN_D15	PMOD Pin 5	3.3 V
PIO_06	PIN_C15	PMOD Pin 6	3.3 V
PIO_07	PIN_B16	PMOD Pin 7	3.3 V
PIO_08	PIN_C16	PMOD Pin 8	3.3 V
GND	N/A	Ground	N/A
3.3V	N/A	3.3 V Power to PMOD	N/A

3.3.8 User I/O

The CYC1000 board has two additional pins that can be connected to the board and take GPIOs.

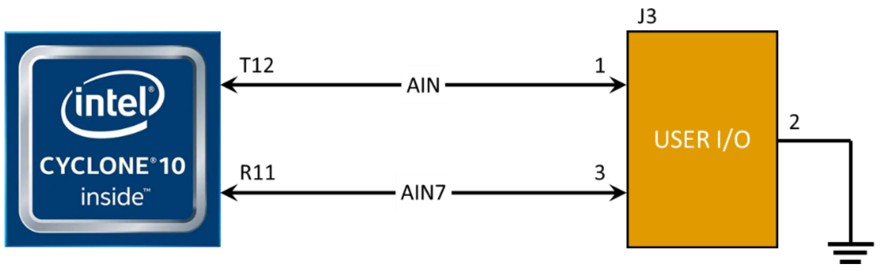


Figure 12 - User I/O Connections

Board Reference	FPGA Pin No.	Description	I/O Standard
AIN	PIN_T12	GPIO	3.3 V
AIN7	PIN_R11	GPIO	3.3 V
GND	N/A	Ground	N/A

3.3.9 Communication and Configuration

The CYC1000 board uses a single chip to perform configuration of the device and USB to UART communications, having each described below.

3.3.9.1 UART Communication

UART to USB communication supports USB 2.0 High Speed (up to 480 Mb/s) independently of other protocols used in the chip like JTAG. Below is the connection schematic and pinning information.

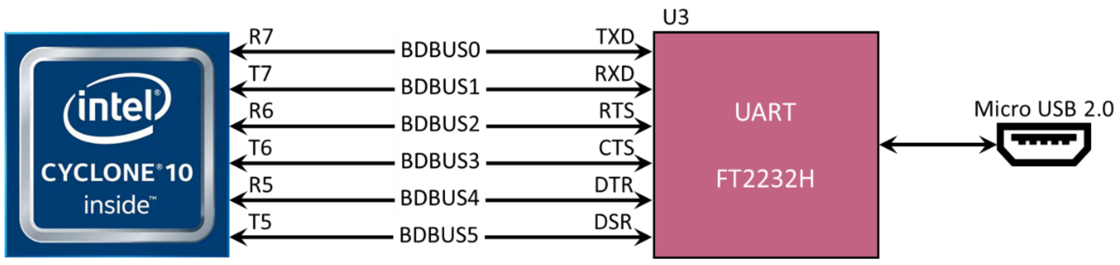


Figure 13 – UART Connections

Board Reference	FPGA Pin No.	Description	I/O Standard
BDBUS0	PIN_R7	Transmitter output of FT2232H (Tx)	3.3 V
BDBUS1	PIN_T7	Receiver input of FT2232H (Rx)	3.3 V
BDBUS2	PIN_R6	Ready To Send handshake output (RTS)	3.3 V
BDBUS3	PIN_T6	Clear To Send handshake input (CTS)	3.3 V
BDBUS4	PIN_R5	Data Transmit Ready (DTR)	3.3 V
BDBUS5	PIN_T5	Data Set Ready (DSR)	3.3 V

3.3.9.2 JTAG Chain Configuration

There are two types of configuration methods supported by CYC1000:

1. **JTAG Configuration:** configuration using JTAG ports. JTAG configuration scheme allows you to directly configure the device core through JTAG pins (TDI, TDO, TMS and TCK pins). The Quartus Prime software automatically generates a .sof that can be downloaded to the Cyclone 10 LP with a download cable through the Quartus Prime Programmer.

2. **Configuration from EPCQ-A flash:** configuration using external flash. Before configuration, you need to program the configuration data .jic into the configuration flash memory (EPCQ-A) which provides non-volatile storage for the bit stream. The information is retained within EPCQ-A even if the CYC1000 is turned off. When the board is powered on, the configuration data in the EPCQ-A is automatically loaded into the Cyclone 10 LP FPGA.

The FPGA device can be configured through JTAG interface on CYC1000, but the JTAG chain must form a closed loop, which allows Quartus Prime programmer to detect the FPGA device.

CYC1000 offers two ways of configuring your board.

1) Through the on-board Arrow USB Programmer2
2) Pins for connecting user's preferred JTAG interface

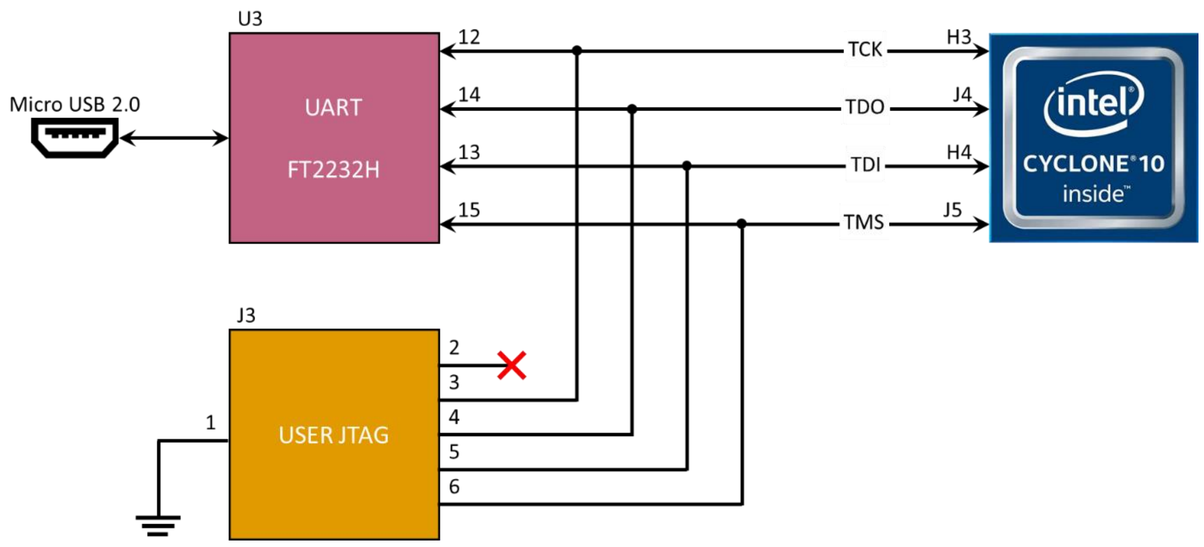


Figure 14 – JTAG Connections

Board Reference	FPGA Pin No.	Description	I/O Standard
TCK	PIN_H3	Test Interface Clock	3.3 V
TDO	PIN_J4	Test Data Out	3.3 V
TDI	PIN_H4	Test Data In	3.3 V
TMS	PIN_J5	Test Mode Select	3.3 V

For detailed information about how to configure the Cyclone 10 LP, please refer to Chapter 6.

3.3.10 Power Tree

The CYC1000 is powered by Enpirion's buck regulator, which provides high efficiency up to 1A with integrated magnetics, switches, control, and compensation. As seen from the diagram below, the board can be powered either by a micro-USB connection, or by user input voltage from the Arduino MKR header (takes precedence over the USB bus). All devices are powered by 3.3V voltage line and the 5V and 3.3V lines are fed back to the Arduino header to power that connection if needed. The Cyclone 10 LP FPGA is powered by 2 Enpirion devices, while a Microchip LDO provides auxiliary voltage.

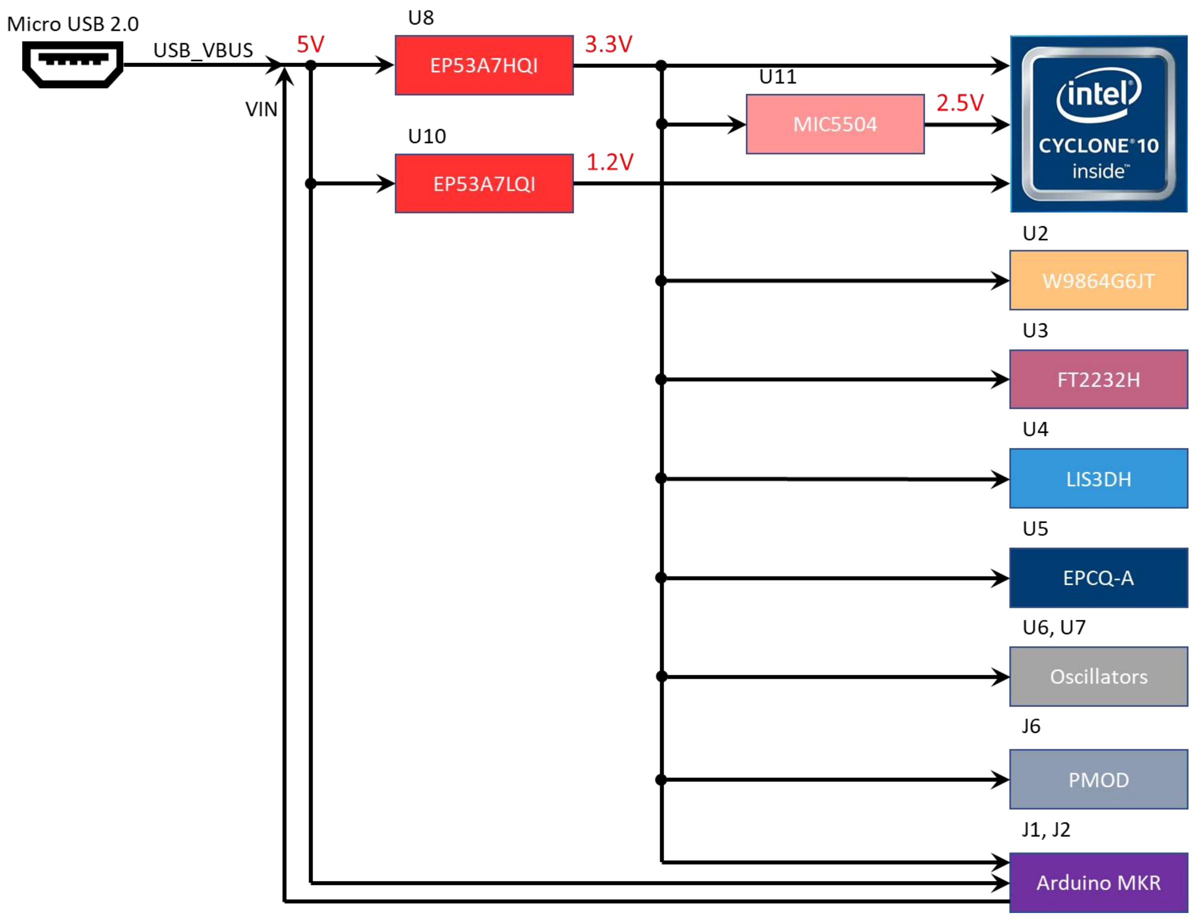


Figure 15 – Power Tree Connections

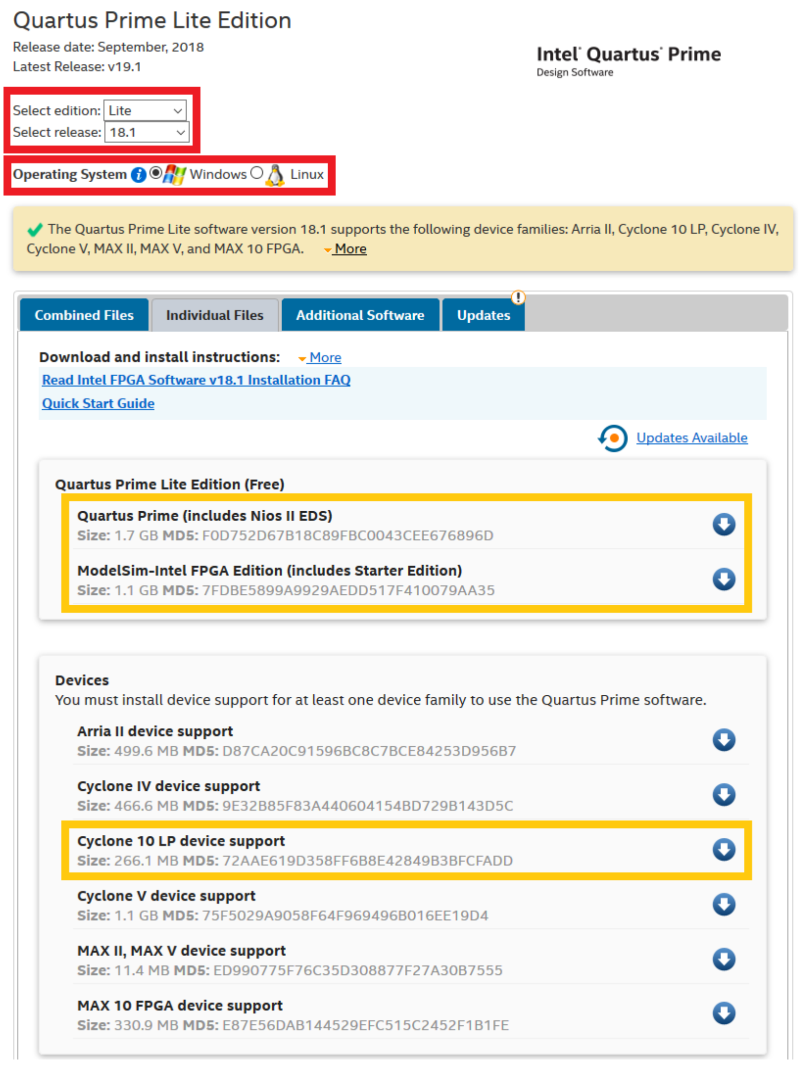
Chapter 4 - Software and Driver Installation

Firstly it is required to create your Basic Intel Account if you don't own one already. It is required to download the software. Below are guides for installing the software and drivers for Windows operating systems.

4.1 Installing Quartus Prime Software

4.1.1 Go to the Intel Download Center: Link.
4.1.2 Select **Windows** as the operating system (highlighted in red).
4.1.3 Select Release **18.1**, or your preferred version (highlighted in red).
4.1.4 Download the following files from the "Individual Files" tab (highlighted in yellow):

- Quartus Prime Lite Edition (Free)
- ModelSim-Intel FPGA Edition (includes Starter Edition)
- Cyclone 10 LP device support



Quartus Prime Lite Edition
Release date: September, 2018
Latest Release: v19.1

Intel Quartus Prime
Design Software

Select edition: Lite
Select release: 18.1

Operating System [✓ Windows] Linux

The Quartus Prime Lite software version 18.1 supports the following device families: Arria II, Cyclone 10 LP, Cyclone IV, Cyclone V, MAX II, MAX V, and MAX 10 FPGA. More

Combined Files | Individual Files | Additional Software | Updates

Download and install instructions: More
Read Intel FPGA Software v18.1 Installation FAQ
Quick Start Guide

Updates Available

Quartus Prime Lite Edition (Free)

Quartus Prime (includes Nios II EDS)
Size: 1.7 GB **MD5:** F0D752D67B18C89FBC0043CEE676896D

ModelSim-Intel FPGA Edition (includes Starter Edition)
Size: 1.1 GB **MD5:** 7FDBE5899A9929AEDD517F410079AA35

Devices
You must install device support for at least one device family to use the Quartus Prime software.

Arria II device support
Size: 499.6 MB **MD5:** D87CA20C91596BC8C7BCE84253D956B7

Cyclone IV device support
Size: 466.6 MB **MD5:** 9E32B85F83A440604154BD729B143D5C

Cyclone 10 LP device support
Size: 266.1 MB **MD5:** 72AAE619D358FF6B8E42849B3BFCFADD

Cyclone V device support
Size: 1.1 GB **MD5:** 75F5029A9058F64F969496B016EE19D4

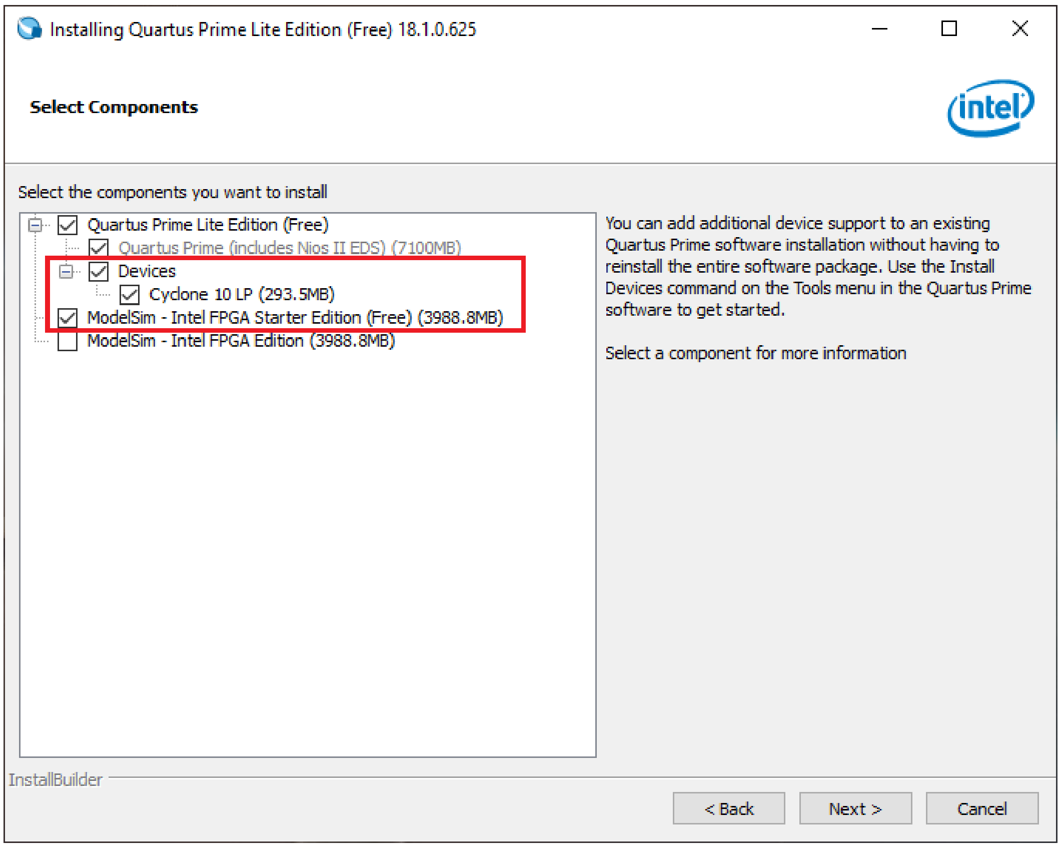
MAX II, MAX V device support
Size: 11.4 MB **MD5:** ED990775F76C35D308877F27A30B7555

MAX 10 FPGA device support
Size: 330.9 MB **MD5:** E87E56DAB144529EFC515C2452F1B1FE

4.1.5 Click on button to begin the download and save them in the same folder.

4.1.6 After the download is finished, run the Quartus Prime installer.

4.1.7 When prompted to select the components, the installer will detect automatically the Cyclone 10 LP device support and ModelSim packages when they are in the same folder. Make sure these components are selected:



Installing Quartus Prime Lite Edition (Free) 18.1.0.625

Select Components

Select the components you want to install

- [✓] Quartus Prime Lite Edition (Free)
  - [✓] Quartus Prime (includes Nios II EDS) (7100MB)
  - [✓] Devices
    - [✓] Cyclone 10 LP (293.5MB)
- [✓] ModelSim - Intel FPGA Starter Edition (Free) (3988.8MB)
- [ ] ModelSim - Intel FPGA Edition (3988.8MB)

You can add additional device support to an existing Quartus Prime software installation without having to reinstall the entire software package. Use the Install Devices command on the Tools menu in the Quartus Prime software to get started.

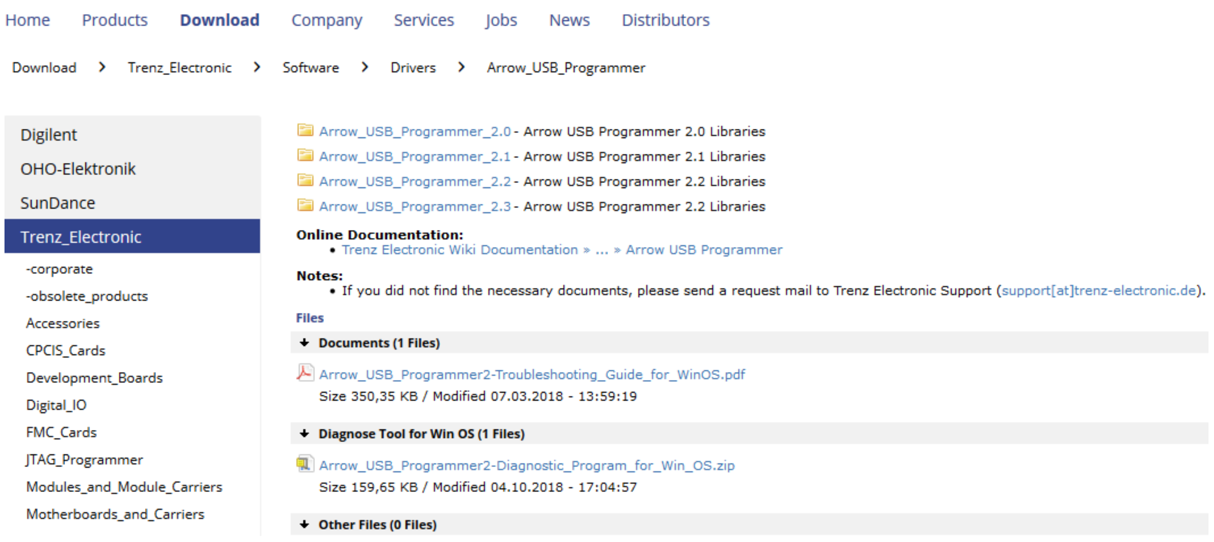
Select a component for more information

4.1.8 Finish the installation of the Quartus Lite and proceeded to the next section to install Arrow USB Programmer2 to be able to connect to the CYC1000 board.

4.2 Installing Arrow USB Programmer2

The CYC1000 board uses version 2 of the Arrow USB Programmer2 programming solution, that is an FTDI FT2232H Hi-Speed USB controller plus a programmer DLL. Since this FTDI USB controller is a very common standard device, usually no specific drivers are needed to make the CYC1000 work.

4.2.1 Download the appropriate version of Arrow USB Programmer2 for CYC1000 from Trenz Electronic Wiki page or alternatively this direct link.


Home Products Download Company Services Jobs News Distributors

Download > Trenz_Electronic > Software > Drivers > Arrow_USB_Programmer

Digilent
OHO-Elektronik
SunDance
Trenz_Electronic
-corporate
-obsolete_products
Accessories
CPCIS_Cards
Development_Boards
Digital_IO
FMC_Cards
JTAG_Programmer
Modules_and_Module_Carriers
Motherboards_and_Carriers

Arrow_USB_Programmer_2.0 - Arrow USB Programmer 2.0 Libraries
Arrow_USB_Programmer_2.1 - Arrow USB Programmer 2.1 Libraries
Arrow_USB_Programmer_2.2 - Arrow USB Programmer 2.2 Libraries
Arrow_USB_Programmer_2.3 - Arrow USB Programmer 2.2 Libraries

Online Documentation:
- Trenz Electronic Wiki Documentation » ... » Arrow USB Programmer

Notes:
- If you did not find the necessary documents, please send a request mail to Trenz Electronic Support (support[at]trenz-electronic.de).

Files

Documents (1 Files)

Arrow_USB_Programmer2-Troubleshooting_Guide_for_WinOS.pdf
Size 350,35 KB / Modified 07.03.2018 - 13:59:19

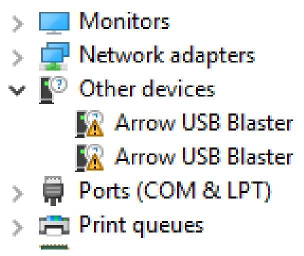
Diagnose Tool for Win OS (1 Files)

Arrow_USB_Programmer2-Diagnostic_Program_for_Win_OS.zip
Size 159,65 KB / Modified 04.10.2018 - 17:04:57

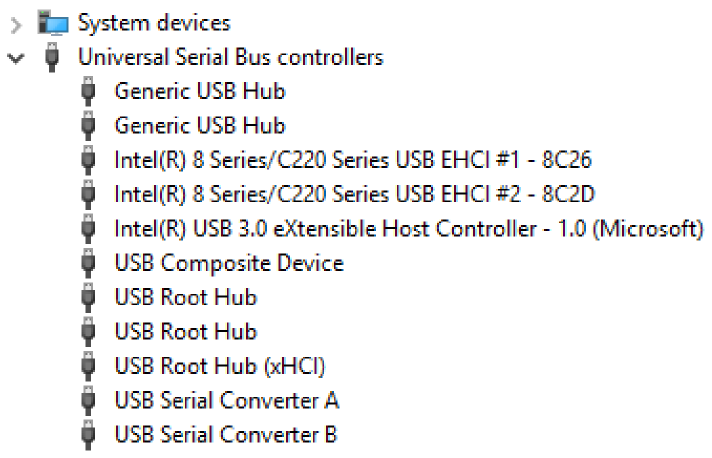
Other Files (0 Files)

4.2.2 After downloading the file, run the installer to install the Arrow USB Programmer2. The setup executable installs the programmer DLL and adds some keys to the registry of the PC.

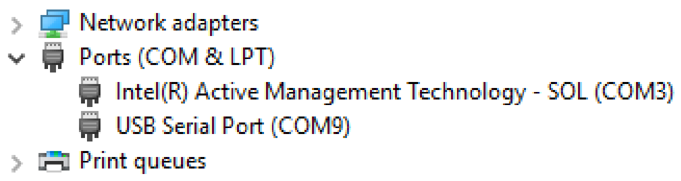
4.2.3 After connecting the CYC1000 board to the PC, two unknown devices might appear in the "Other devices" section of device manager of the PC.


Monitors
Network adapters
Other devices
Arrow USB Blaster
Arrow USB Blaster
Ports (COM & LPT)
Print queues

Windows usually automatically finds the appropriate drivers for these devices. After some time, the "Other devices" section should be empty. Instead, two USB Serial Converters should be listed in the section "USB Serial Bus controllers":


System devices
Universal Serial Bus controllers
Generic USB Hub
Generic USB Hub
Intel(R) 8 Series/C220 Series USB EHCI #1 - 8C26
Intel(R) 8 Series/C220 Series USB EHCI #2 - 8C2D
Intel(R) USB 3.0 eXtensible Host Controller - 1.0 (Microsoft)
USB Composite Device
USB Root Hub
USB Root Hub
USB Root Hub (xHCI)
USB Serial Converter A
USB Serial Converter B

Furthermore, a USB Serial Port should be listed in the “Ports (COM & LPT)” section.


Network adapters
Ports (COM & LPT)
Intel(R) Active Management Technology - SOL (COM3)
USB Serial Port (COM9)
Print queues

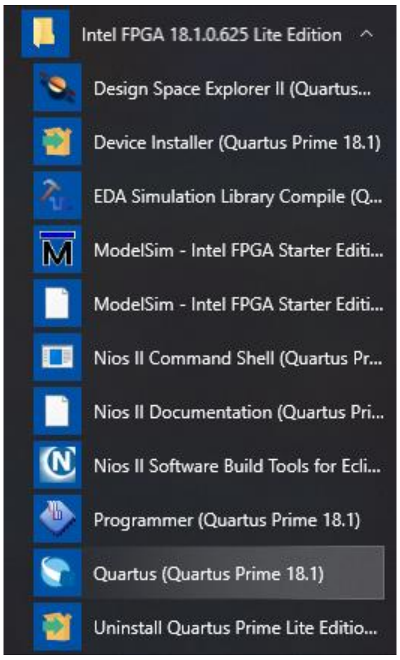
Note: The number of the port will most probably be different from the one shown here.

In case Windows does not automatically find the appropriate drivers go to http://www.ftdichip.com/Drivers/D2XX.htm to download the setup executable to install the required drivers.

Chapter 5 - New Project with CYC1000

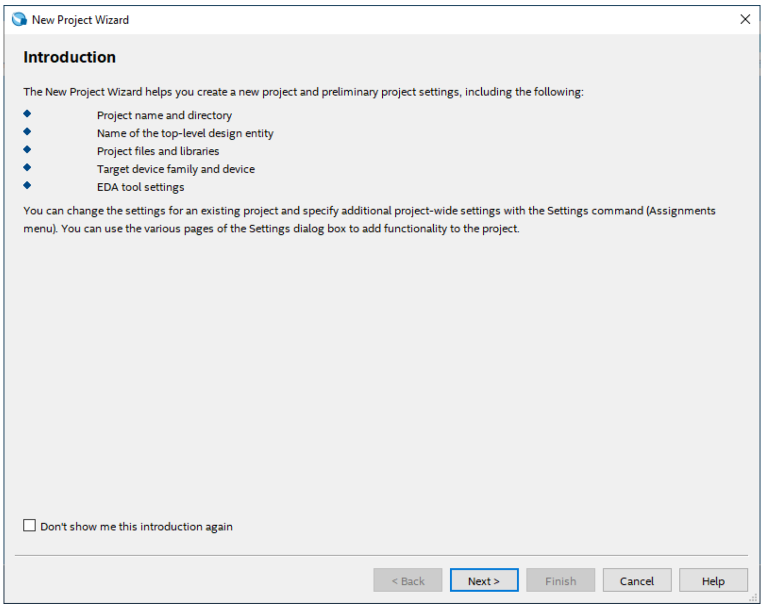
5.1 Creating a new Blinky Project with CYC1000

5.1.1 Launch Quartus Prime Lite Edition from the Start Menu.


Intel FPGA 18.1.0.625 Lite Edition
Design Space Explorer II (Quartus...
Device Installer (Quartus Prime 18.1)
EDA Simulation Library Compile (Q...
ModelSim - Intel FPGA Starter Editi...
ModelSim - Intel FPGA Starter Editi...
Nios II Command Shell (Quartus Pr...
Nios II Documentation (Quartus Pri...
Nios II Software Build Tools for Ecli...
Programmer (Quartus Prime 18.1)
Quartus (Quartus Prime 18.1)
Uninstall Quartus Prime Lite Editio...

5.1.2 In the Quartus Prime tool, create a new project: **File -> New Project Wizard.**

The New Project Wizard walks you through the project settings, such as the name, directories, files, directories, device family and other settings. These settings can be changed later if needed.


New Project Wizard

Introduction

The New Project Wizard helps you create a new project and preliminary project settings, including the following:

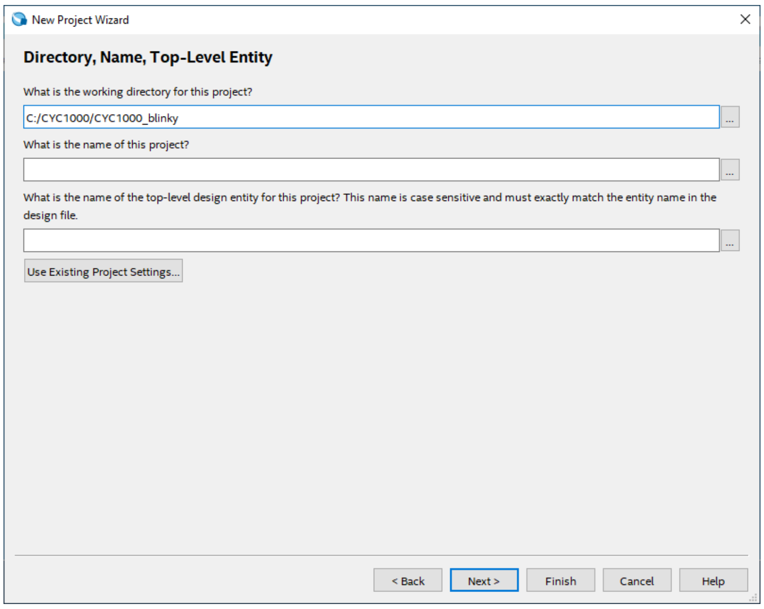
- Project name and directory
- Name of the top-level design entity
- Project files and libraries
- Target device family and device
- EDA tool settings

You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.

☐ Don't show me this introduction again

5.1.3 Click "Next".

5.1.4 Browse in the project directory and choose a preferred location for the new project. Then create new folder named CYC1000_blinky. This will be the folder containing all the project files.


New Project Wizard

Directory, Name, Top-Level Entity

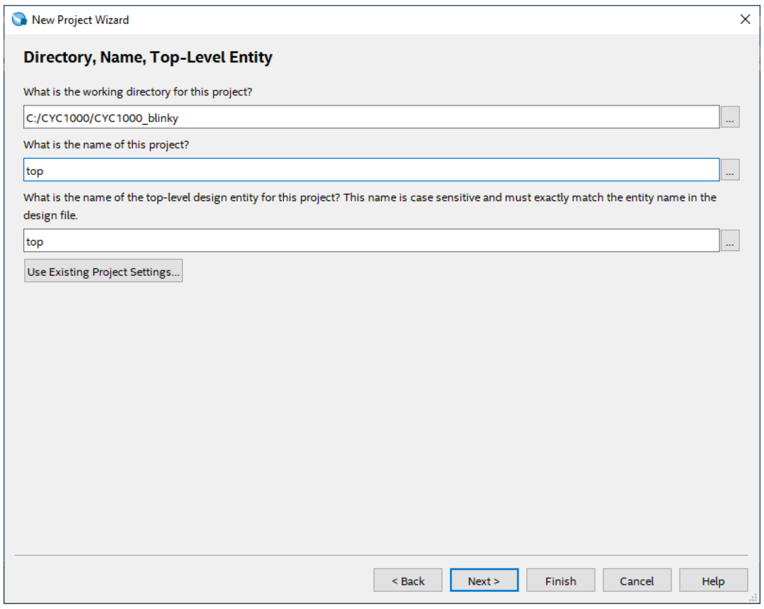
What is the working directory for this project?

C:/CYC1000/CYC1000_blinky

What is the name of this project?

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

5.1.5 Enter the project name: “top”.


New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

C:/CYC1000/CYC1000_blinky

What is the name of this project?

top

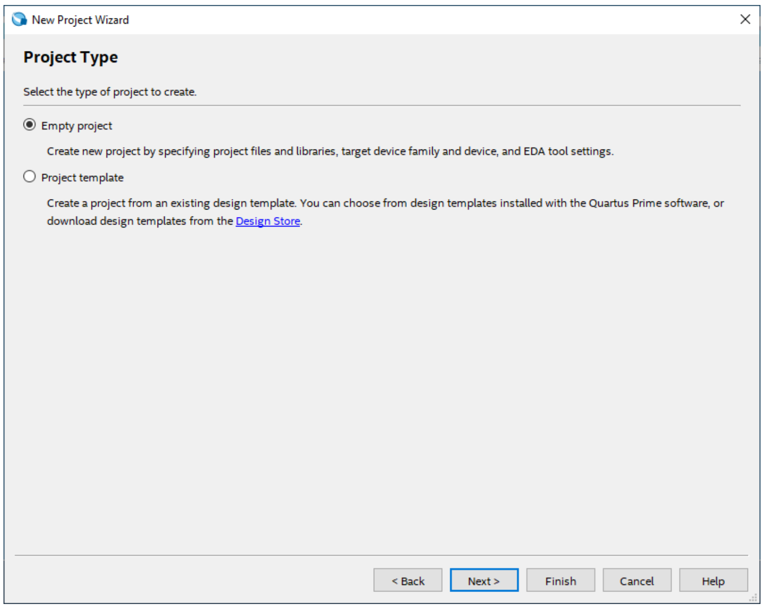
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

top

5.1.6 Click “Next”.

5.1.7 Project Type

In this page you choose the Project Type. In this tutorial, a new project will be created, and thus the default settings of empty project should be selected.



New Project Wizard

Project Type

Select the type of project to create.

[✓ Empty project]

Create new project by specifying project files and libraries, target device family and device, and EDA tool settings.

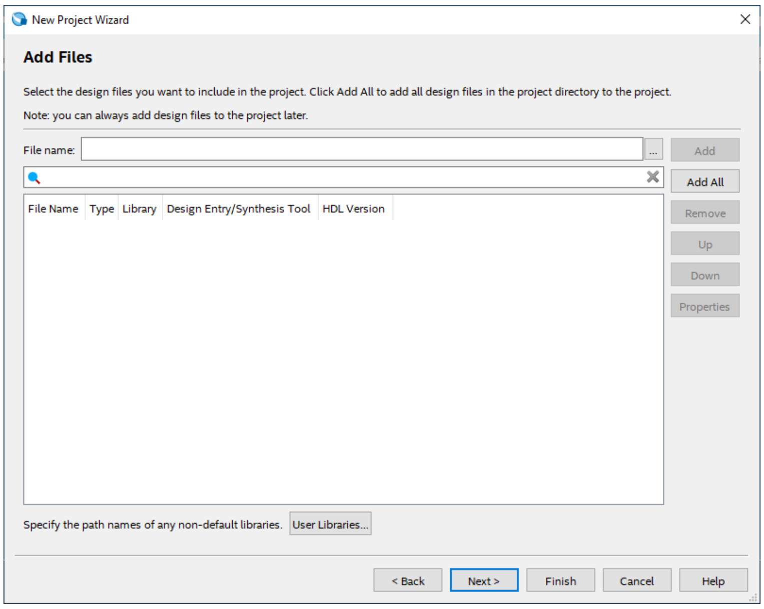
Project template

Create a project from an existing design template. You can choose from design templates installed with the Quartus Prime software, or download design templates from the Design Store.

5.1.8 Click “Next”.

5.1.9 Add Project Files

The Add File window will appear. For this tutorial, new design files will be created so no files will be added. For other designs, files could be added here.



New Project Wizard

Add Files

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

Note: you can always add design files to the project later.

File name:

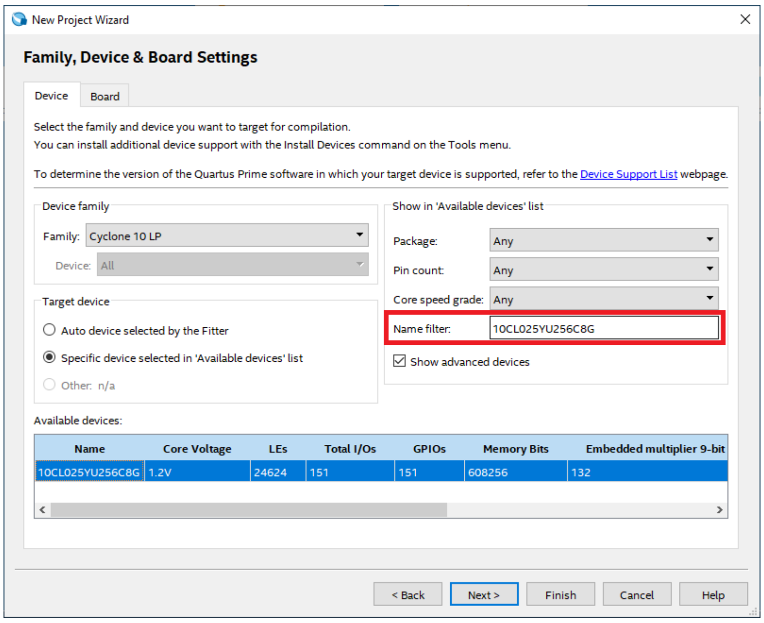
File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version

Specify the path names of any non-default libraries.

5.1.10 Click “Next”.

5.1.11 Select the Device Part Number of the CYC1000 Board

In the Family and Device Settings, use the pull-down menu to select the family as Cyclone 10 LP. Then in the Name Filter enter **10CL025YU256C8G**.



New Project Wizard

Family, Device & Board Settings

Device | Board

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the Device Support List webpage.

Device family
Family: Cyclone 10 LP
Device: All

Target device
Auto device selected by the Fitter
[✓ Specific device selected in 'Available devices' list]
Other: n/a

Show in 'Available devices' list
Package: Any
Pin count: Any
Core speed grade: Any
Name filter: 10CL025YU256C8G
[✓ Show advanced devices]

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit
10CL025YU256C8G	1.2V	24624	151	151	608256	132

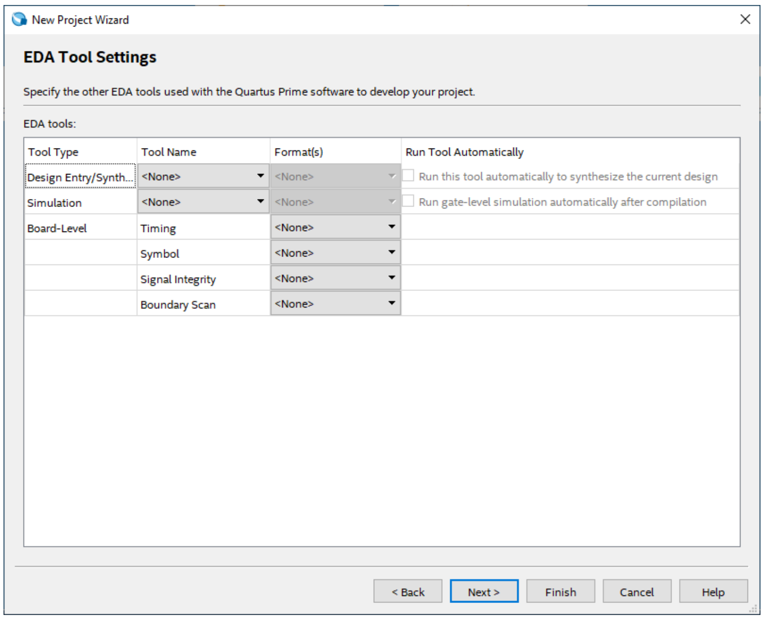
< Back | Next > | Finish | Cancel | Help

Rather than entering the exact part number, the pull-down menus can be used to select the correct family, package, pin count, and speed grade. Quartus Prime will use these settings to compile the design, and also provide the programming file that you will use later to program the device.

5.1.12 Click “Next”.

5.1.13 EDA Tool Settings

In the EDA tool Settings window, disable any EDA tools, if there are any present. EDA tools are third party tools that work with Quartus Prime for design entry, simulation, verification and board-level timing. For this tutorial, no EDA software will be used, as only Quartus Prime will be used.



New Project Wizard

EDA Tool Settings

Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

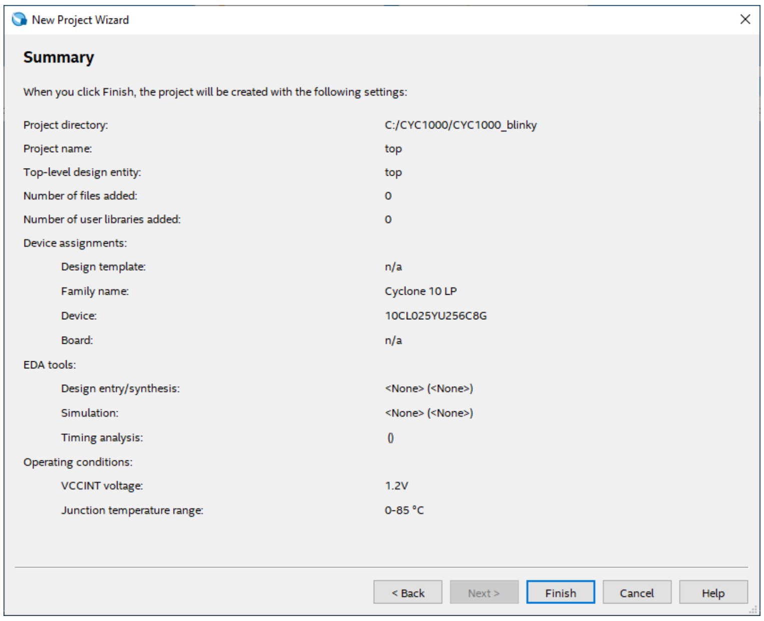
Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synth...	<None>	<None>	Run this tool automatically to synthesize the current design
Simulation	<None>	<None>	Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

< Back | Next > | Finish | Cancel | Help

5.1.14 Click “Next”.

5.1.15 Project Summary Page

This is the Summary Page that showing the settings Quartus Prime will use for this Project. Those settings can be changed if required at a later time.



Summary

When you click Finish, the project will be created with the following settings:

Project directory:	C:/CYC1000/CYC1000_blinky
Project name:	top
Top-level design entity:	top
Number of files added:	0
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	Cyclone 10 LP
Device:	10CL025YU256C8G
Board:	n/a
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	<None> (<None>)
Timing analysis:	0
Operating conditions:	
VCCINT voltage:	1.2V
Junction temperature range:	0-85 °C

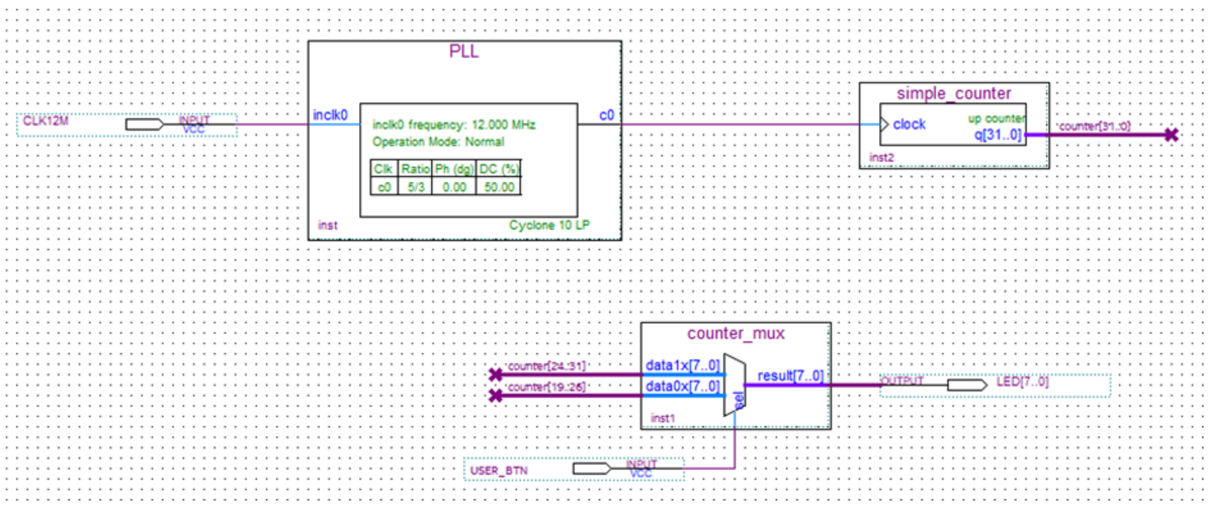
5.1.16 Click “Finish”.

5.2 Building a Blinky Project with CYC1000

Overview: In this section you will create the components to a design, make connections, set the pins and compile a project. The goal is to go through the design process of a simple blinky project, where the toggle speed of the LEDs could be controlled by one of the pushbuttons of the CYC1000.

5.2.1 Block Diagram

The final system that will be built with the following steps will look as follows when complete:

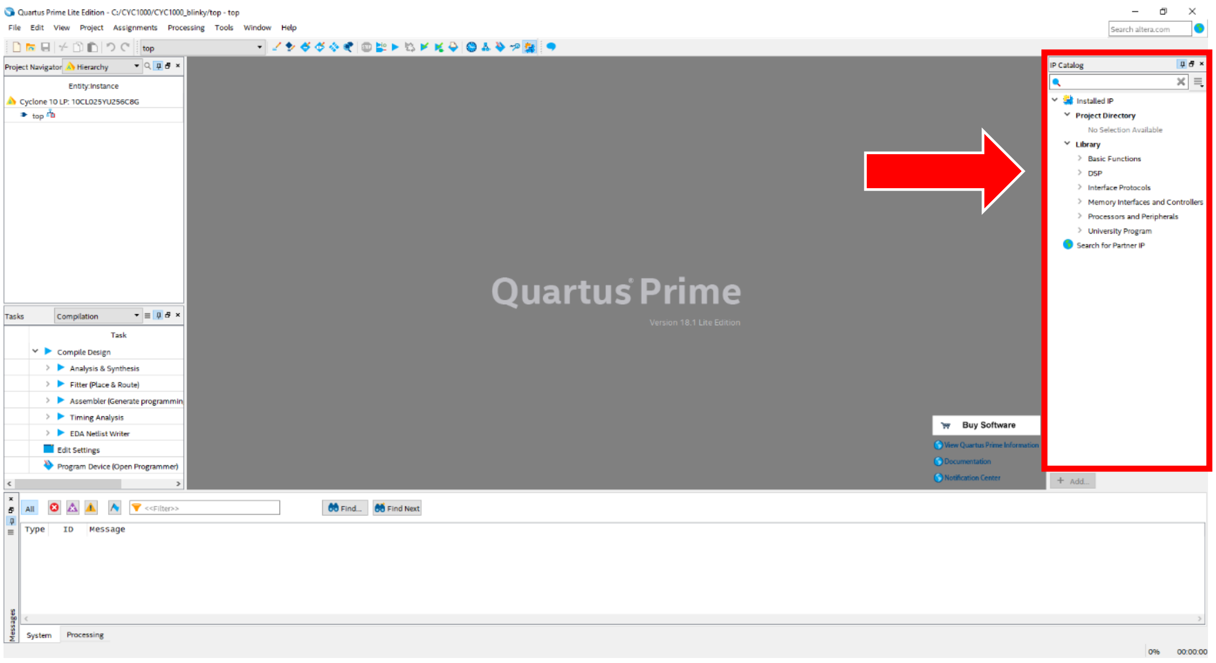


5.2.2 Components of the Design

There are three components in the system: a PLL, a counter and a mux. The components, in the following steps, will be built separately and then connected together. A user push button on the board controls the mux. The mux in turn control which of the counter outputs (slow counting or fast counting) will be shown on the LEDs. There are different ways to create components, such as RTL or schematic. In this lab, schematics will be used. There are also different ways for entering schematics such as Qsys and IP Catalog. This lab will focus on the IP Catalog.

5.2.3 Catalog IP

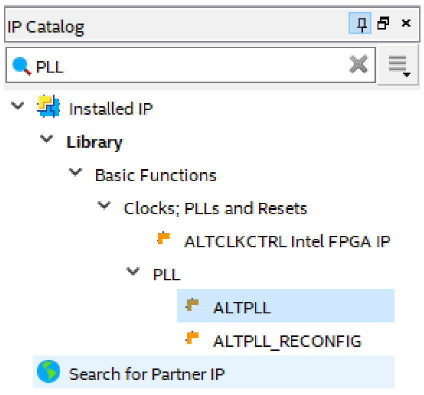
The IP Catalog allows you to create and modify design files with custom variations. The IP Catalog window is open by default when you open Quartus Prime. If it's not present, you can open it by going to the tab **Tool → IP Catalog**.



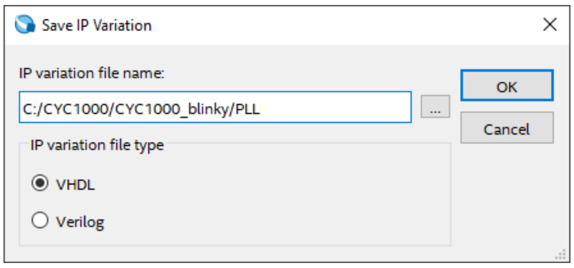
5.2.4 Create a PLL

In the IP Catalog, browse for ALTPLL, via: Basic Functions → Clocks; PLLs and Resets → PLL or type in the search field for "PLL".

5.2.4.1 In the Search bar of the IP Catalog, type "pll" and select **ALTPLL** which stands for Altera Phase Locked Loop.



5.2.4.2 Click “Add”. When the Save IP Variation window appears, enter the file name variation as PLL and select VHDL (Verilog can be used as well). Both Verilog and VHDL schematics will be created.


Save IP Variation

IP variation file name: C:/CYC1000/CYC1000_blinky/PLL

IP variation file type: [✓ VHDL] Verilog

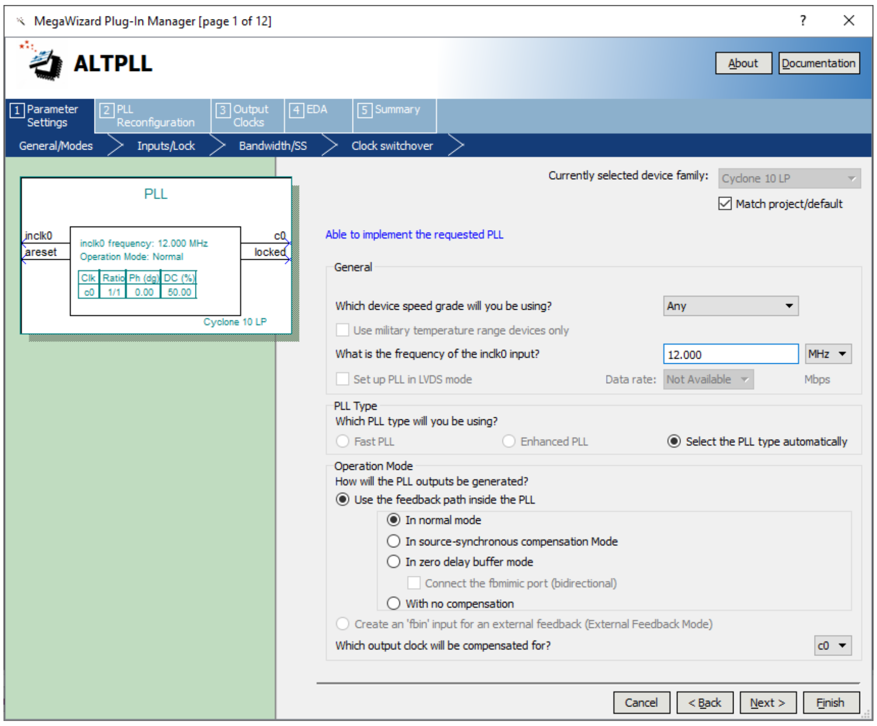
5.2.4.3 Click “OK”.

5.2.5 Create and Configure the PLL

The next step is to configure the PLL component that we just named.

5.2.5.1 Enter the PLL reference clock frequency to match the clock input on the CYC1000 Board. Since we have a 12 MHz coming into the FPGA, the inclk0 input will be 12MHz.

The setting should look like this:


MegaWizard Plug-In Manager [page 1 of 12]

ALTPLL

inclk0 frequency: 12.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

Currently selected device family: Cyclone 10 LP
☑ Match project/default

Able to implement the requested PLL

General

Which device speed grade will you be using? Any
☐ Use military temperature range devices only
What is the frequency of the inclk0 input? 12.000 MHz
☐ Set up PLL in LVDS mode   Data rate: Not Available Mbps

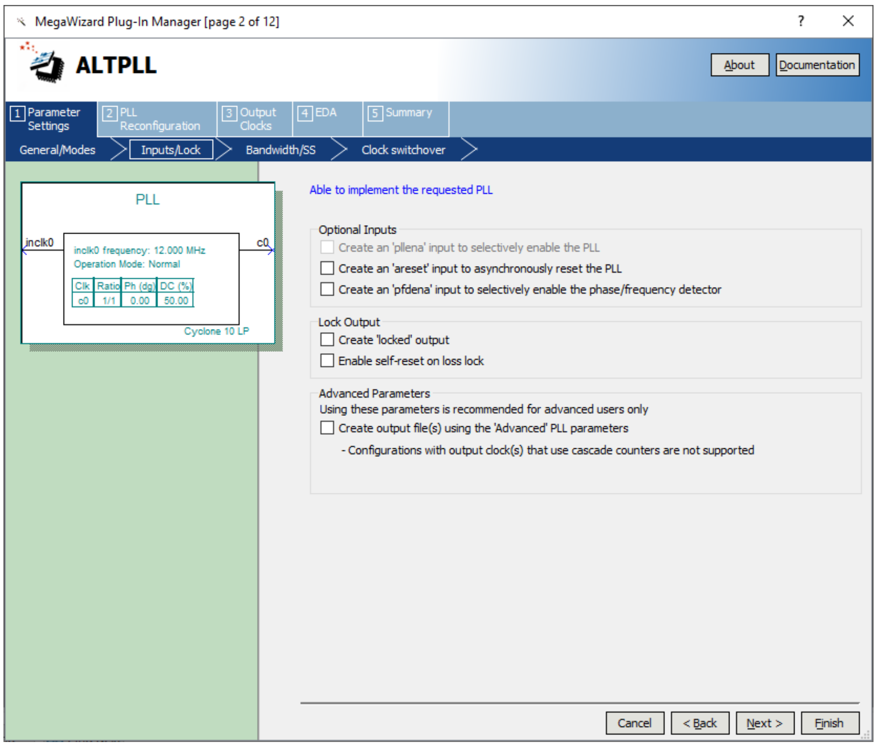
PLL Type
Which PLL type will you be using?
Fast PLL   Enhanced PLL   [✓ Select the PLL type automatically]

Operation Mode
How will the PLL outputs be generated?
[✓ Use the feedback path inside the PLL]
[✓ In normal mode]
In source-synchronous compensation Mode
In zero delay buffer mode
☐ Connect the fbmimic port (bidirectional)
With no compensation
Create an 'fbin' input for an external feedback (External Feedback Mode)
Which output clock will be compensated for? c0

5.2.5.2 Click “Next”.

5.2.5.3 Simplify the PLL, by disabling 'areset' and 'locked outputs'.

The setting should look like this:


MegaWizard Plug-In Manager [page 2 of 12]

ALTPLL

Able to implement the requested PLL

Optional Inputs
- [ ] Create an 'pllena' input to selectively enable the PLL
- [ ] Create an 'areset' input to asynchronously reset the PLL
- [ ] Create an 'pfdena' input to selectively enable the phase/frequency detector

Lock Output
- [ ] Create 'locked' output
- [ ] Enable self-reset on loss lock

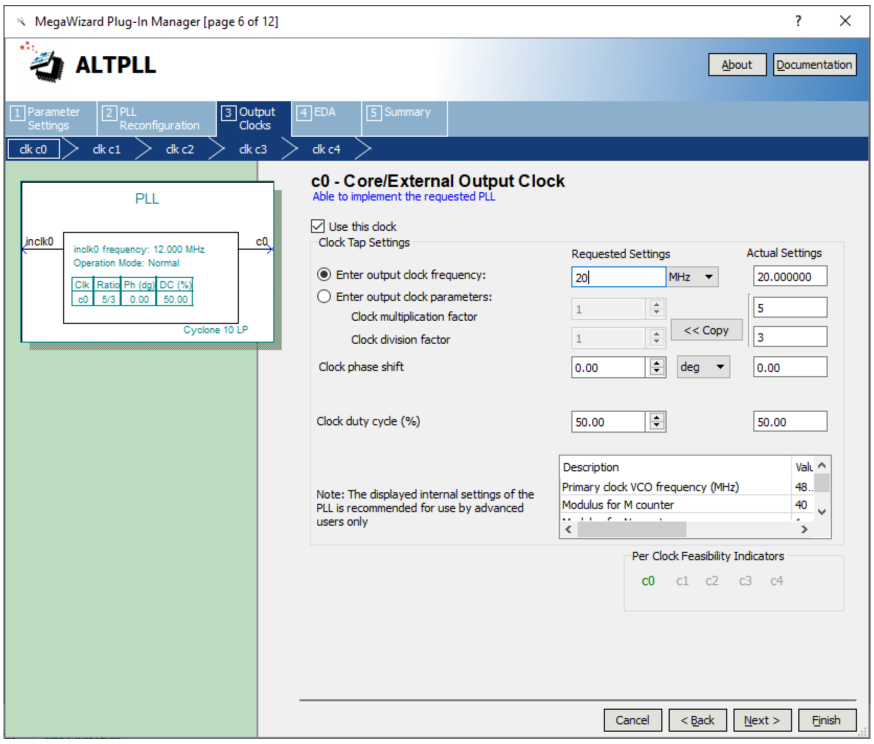
Advanced Parameters
Using these parameters is recommended for advanced users only
- [ ] Create output file(s) using the 'Advanced' PLL parameters

  - Configurations with output clock(s) that use cascade counters are not supported

5.2.5.4 Click "Next".

5.2.5.5 Continue to select Next to go through the various options (e.g. Pages 3 to Pages 5) but leaving the default options as they are. The page numbers can be seen on the top of the window.

5.2.5.6 On page 6, (c0-Core/External Output Clock) select "Enter output clock frequency" and set the requested setting to 20 MHz, leave the rest as default. For simplification, there is one input to the PLL (12 MHz), and one output of the PLL (20 MHz)



inclk0 frequency: 12.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	5/3	0.00	50.00

c0 - Core/External Output Clock
Able to implement the requested PLL

☑ Use this clock

Clock Tap Settings

	Requested Settings		Actual Settings
● Enter output clock frequency:	20	MHz	20.000000
○ Enter output clock parameters:			
Clock multiplication factor	1	<< Copy	5
Clock division factor	1		3
Clock phase shift	0.00	deg	0.00
Clock duty cycle (%)	50.00		50.00

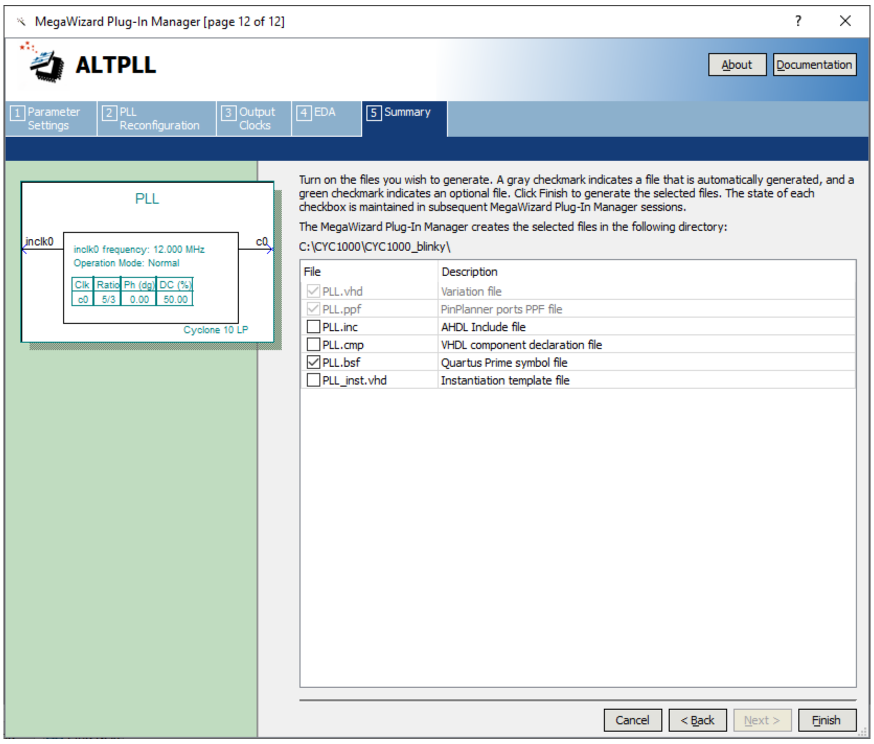
Note: The displayed internal settings of the PLL is recommended for use by advanced users only

Description	Valu
Primary clock VCO frequency (MHz)	48...
Modulus for M counter	40

Per Clock Feasibility Indicators
c0 c1 c2 c3 c4

5.2.5.7 Click "Next" until reaching page 12.

5.2.5.8 On page 12 there is a list of output files that will be generated. Since the design will be done in a schematic, you will need to select PLL.bsf checkbox. The .bsf file provides a symbol that can be used in the schematic design we will be creating later.



inclk0 frequency: 12.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	5/3	0.00	50.00

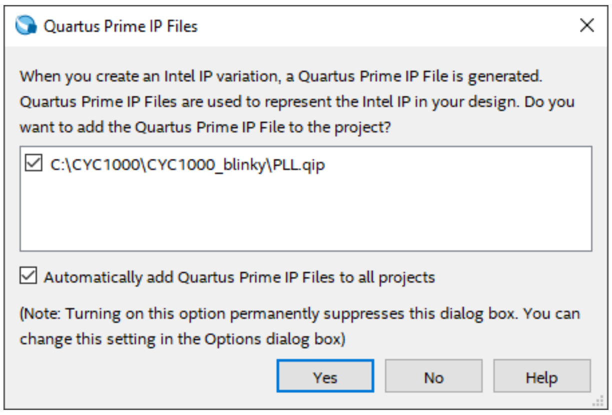
Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a green checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

The MegaWizard Plug-In Manager creates the selected files in the following directory:
C:\CYC1000\CYC1000_blinky\

File	Description
☑ PLL.vhd	Variation file
☑ PLL.ppf	PinPlanner ports PPF file
☐ PLL.inc	AHDL Include file
☐ PLL.cmp	VHDL component declaration file
☑ PLL.bsf	Quartus Prime symbol file
☐ PLL_inst.vhd	Instantiation template file

5.2.5.9 Click “Finish”. The PLL (1st component) will now be created.

5.2.5.10 If this is the first time that you are using this version of Quartus Prime, you might see a pop-up Window for Quartus Prime IP Files, that asks if the tool should add IP files automatically after generating them.


Quartus Prime IP Files

When you create an Intel IP variation, a Quartus Prime IP File is generated. Quartus Prime IP Files are used to represent the Intel IP in your design. Do you want to add the Quartus Prime IP File to the project?

☑ C:\CYC1000\CYC1000_blinky\PLL.qip

☑ Automatically add Quartus Prime IP Files to all projects

(Note: Turning on this option permanently suppresses this dialog box. You can change this setting in the Options dialog box)

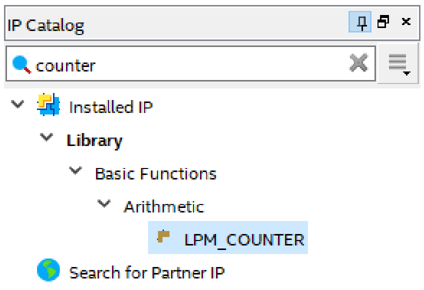
5.2.5.11 Select “Automatically add Quartus Prime IP Files to all projects”.

5.2.5.12 Click “Yes” to allow all of the IP to automatically be added to the project, and so that this message will not be seen for other designs.

5.2.6 Create and Configure the Counter

The next step is to create the counter which will drive the LEDs on the CYC1000 board.

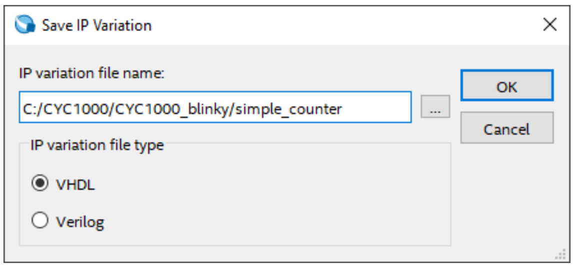
5.2.6.1 To create this counter, select the IP Catalog and expand the **Basics → Arithmetic** and select the LPM_COUNTER or type “counter” in the search field.



Note: LPM stands for Library of Parameterized Modules

5.2.6.2 Click “Add”.

5.2.6.3 When the Save IP Variation pop up appears, enter "simple_counter" and select VHDL as below:



Save IP Variation

IP variation file name: C:/CYC1000/CYC1000_blinky/simple_counter

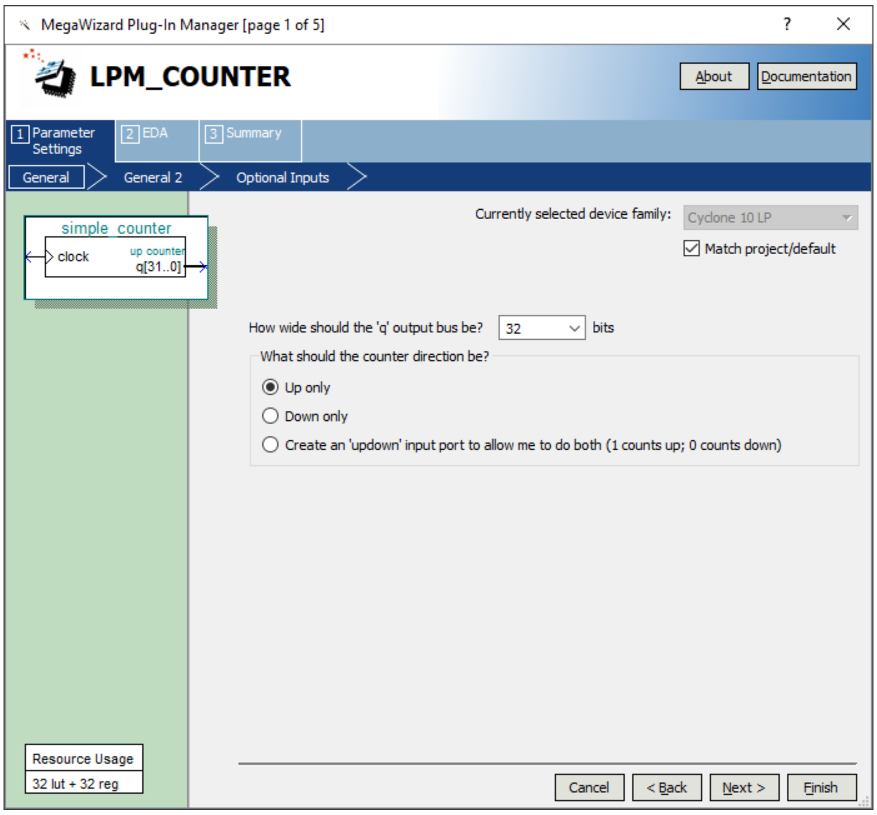
IP variation file type: [✓ VHDL] Verilog

5.2.6.4 Click "OK".

5.2.6.5 The next step is to increase the size of the counter to a number of bits large enough to divide down the clock so we can see the LEDs toggling.

5.2.6.6 Change this number to 32.

5.2.6.7 Let the counter to be Up only, so the LEDs will show the counters counting up.



MegaWizard Plug-In Manager [page 1 of 5]

LPM_COUNTER

Currently selected device family: Cyclone 10 LP

[✓] Match project/default

How wide should the 'q' output bus be? 32 bits

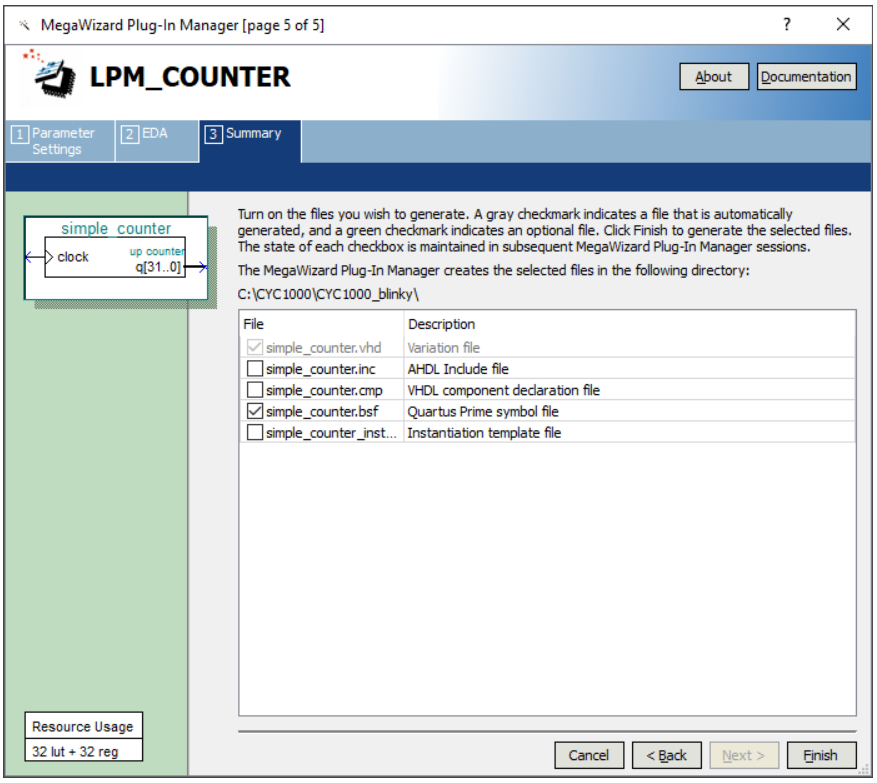
What should the counter direction be?

[✓ Up only] Down only Create an 'updown' input port to allow me to do both (1 counts up; 0 counts down)

Resource Usage: 32 lut + 32 reg

5.2.6.8 Select "Next" until reaching Page 5.

5.2.6.9 Select simple_counter.bsf checkbox to generate a symbol for our schematic design.



Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a green checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

The MegaWizard Plug-In Manager creates the selected files in the following directory:

C:\CYC1000\CYC1000_blinky\

File	Description
✓ simple_counter.vhd	Variation file
☐ simple_counter.inc	AHDL Include file
☐ simple_counter.cmp	VHDL component declaration file
✓ simple_counter.bsf	Quartus Prime symbol file
☐ simple_counter_inst...	Instantiation template file

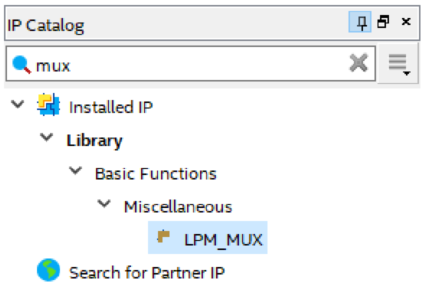
5.2.6.10 Click "Finish".

The counter is now created.

5.2.7 Create and Configure the Multiplexer

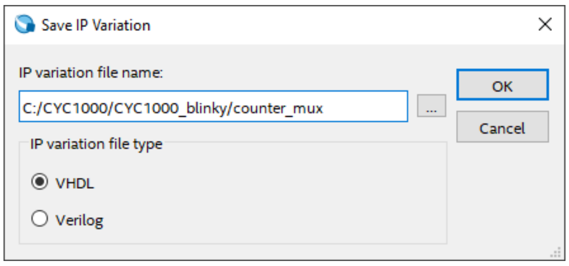
The next step is to create a mux component. This mux will be used along with a push button on the CYC1000 board to control the speed of the counter, where the counter outputs will be seen on the LEDs.

5.2.7.1 To create this mux, select IP Catalog and expand **Basic Functions → Miscellaneous** and select LPM_MUX or type mux in the search field.



5.2.7.2 Click "Add".

5.2.7.3 In the Save IP Variation, enter the name of the counter_mux and the file type to be VHDL.



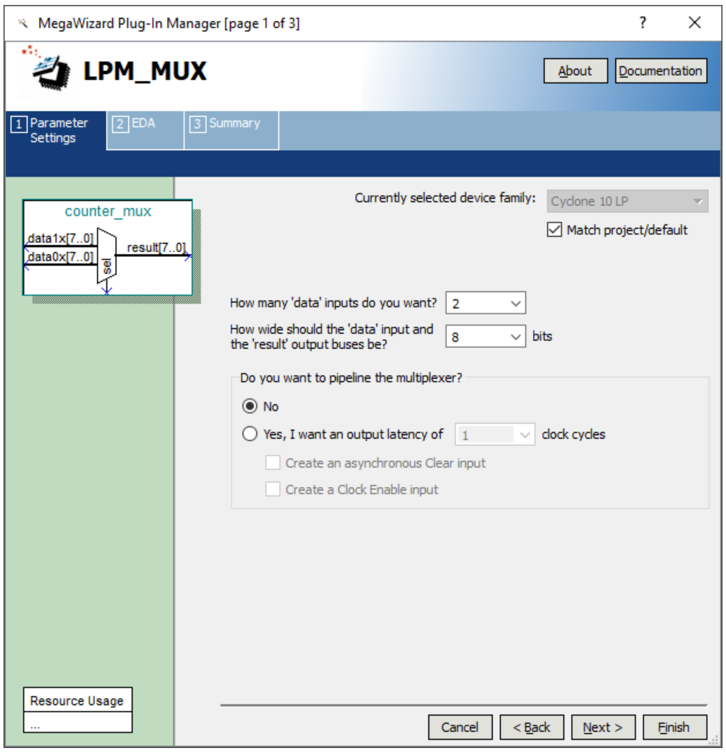
IP variation file name: C:/CYC1000/CYC1000_blinky/counter_mux

IP variation file type: [✓ VHDL] Verilog

5.2.7.4 Click "OK".

5.2.7.5 Select 2 data inputs and the width of the input and output buses to be 8 bits. The reason for 8 bits is that there are 8 LEDs to be toggled (showing count values).

The screen should look like this now:



Currently selected device family: Cyclone 10 LP

☑ Match project/default

How many 'data' inputs do you want? 2

How wide should the 'data' input and the 'result' output buses be? 8 bits

Do you want to pipeline the multiplexer?

[✓ No]

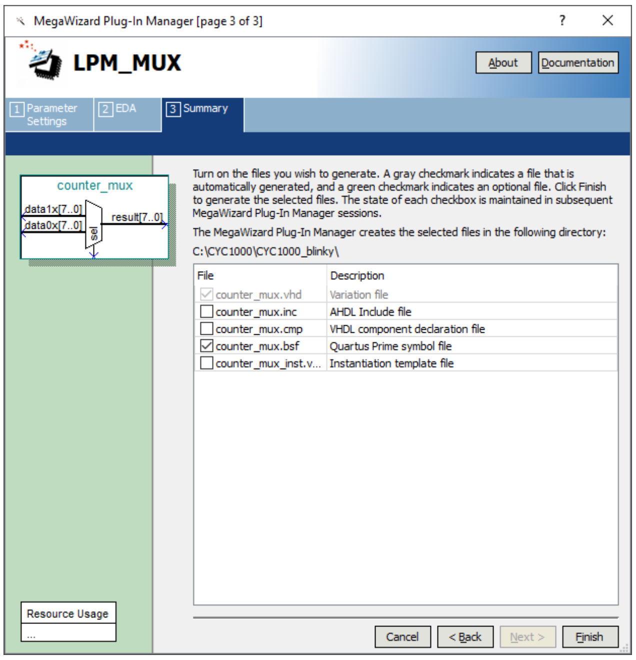
Yes, I want an output latency of 1 clock cycles

☐ Create an asynchronous Clear input

☐ Create a Clock Enable input

5.2.7.6 Click "Next" until Page 3.

5.2.7.7 Select counter_mux.bsf checkbox to generate a symbol for our schematic design.



MegaWizard Plug-In Manager [page 3 of 3]

LPM_MUX

Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a green checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

The MegaWizard Plug-In Manager creates the selected files in the following directory:

C:\CYC1000\CYC1000_blinky\

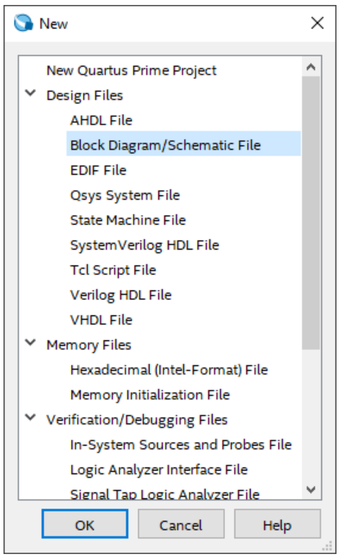
File	Description
✓ counter_mux.vhd	Variation file
☐ counter_mux.inc	AHDL Include file
☐ counter_mux.cmp	VHDL component declaration file
✓ counter_mux.bsf	Quartus Prime symbol file
☐ counter_mux_inst.v...	Instantiation template file

5.2.7.8 Click “Finish”.

5.2.8 Adding the Components to the Schematic

The next step would be to connect all three components together.

5.2.8.1 To do so, select File menu, then select New and select Block Diagram/Schematic File.

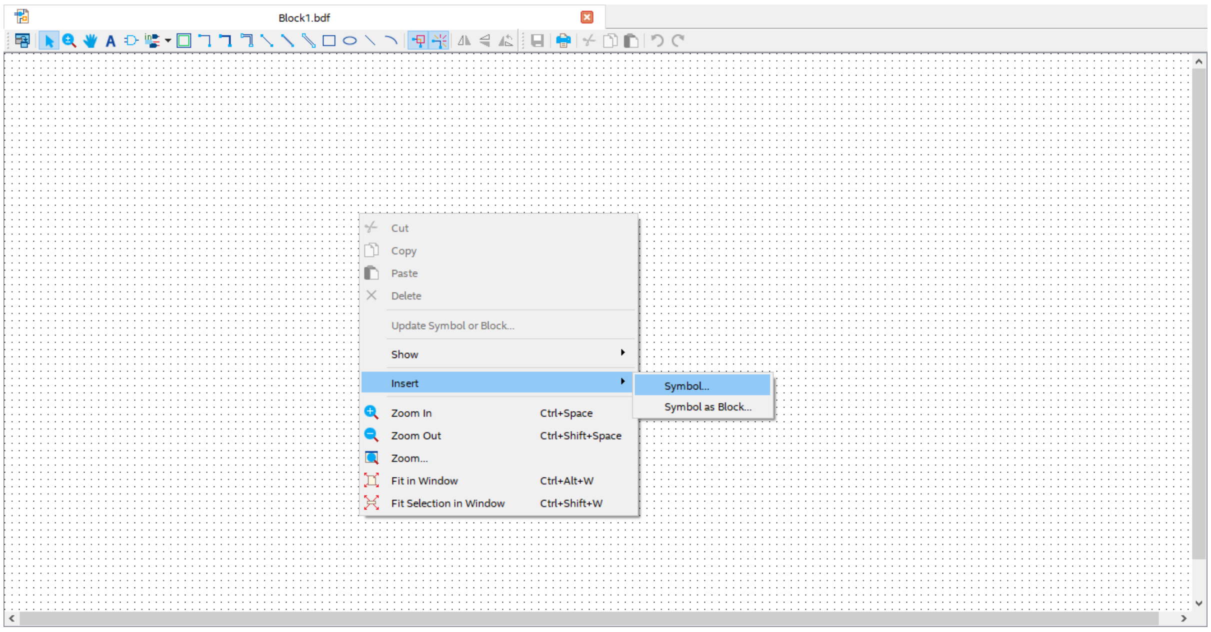


New
New Quartus Prime Project
Design Files
AHDL File
Block Diagram/Schematic File
EDIF File
Qsys System File
State Machine File
SystemVerilog HDL File
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes File
Logic Analyzer Interface File
Signal Tap Logic Analyzer File
OK
Cancel
Help

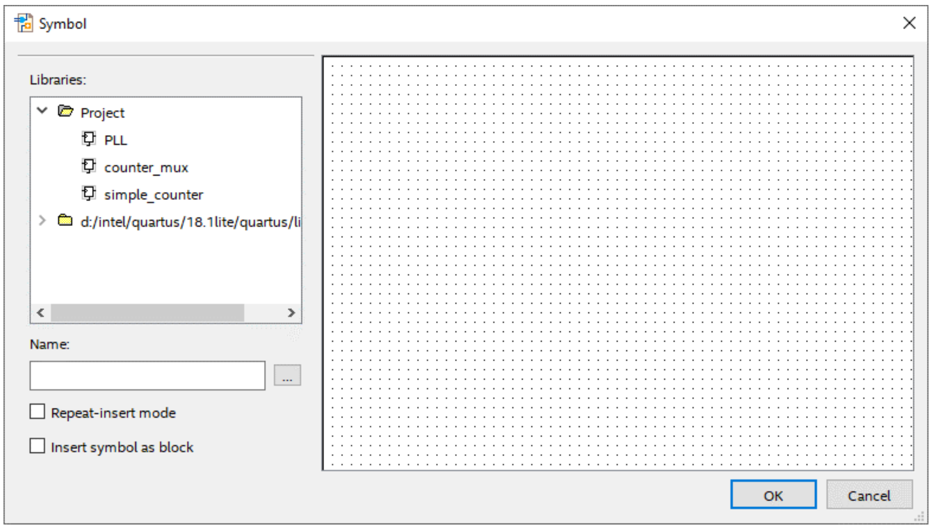
5.2.8.2 Click “OK”.

A new schematic will be created, where the components can be added.

5.2.8.3 Right click on the schematic page and select **Insert → Symbol…** as seen below.



5.2.8.4 In the new window, expand "Project" and the three components that were created can now be seen.



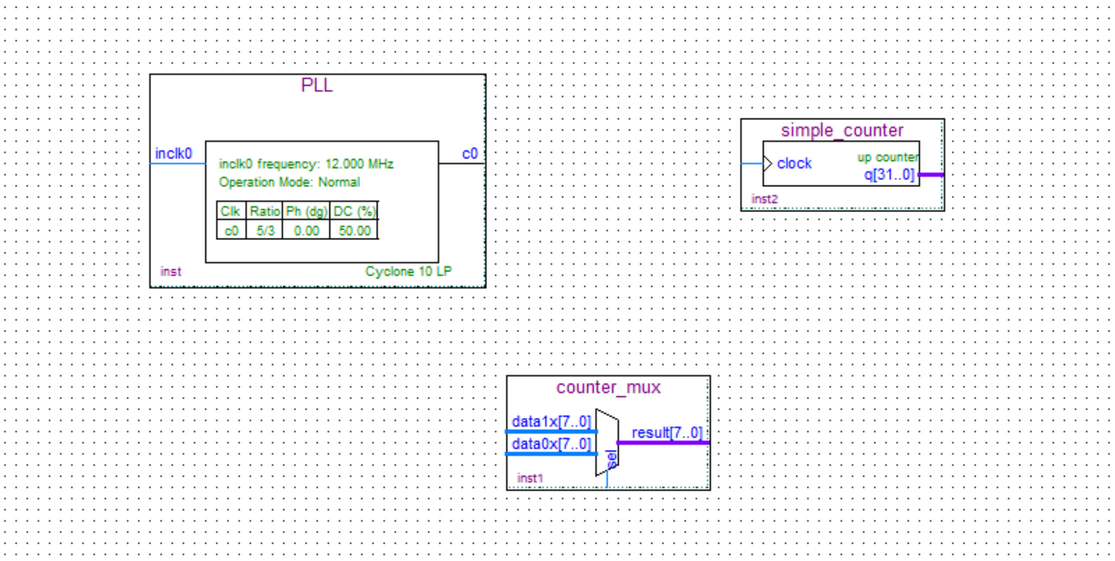
5.2.8.5 Select "PLL".

5.2.8.6 Click "OK".

5.2.8.7 The PLL component can be added now by left clicking on the schematic page.

5.2.8.8 Just like in the steps from 5.2.8.3 to 5.2.8.6, do the same for counter_mux and simple_counter to add them to the schematic page. The order of adding the components does not matter, as the connections between them will happen in the following steps.

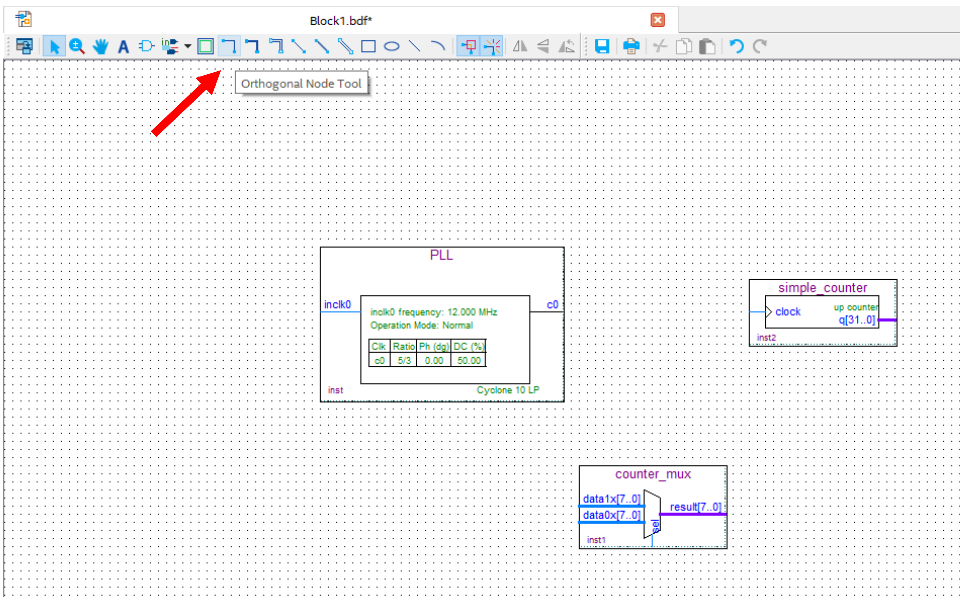
5.2.8.9 After adding three components, your schematic should look similar to the following. To place them similarly, simply drag the components to the appropriate locations.



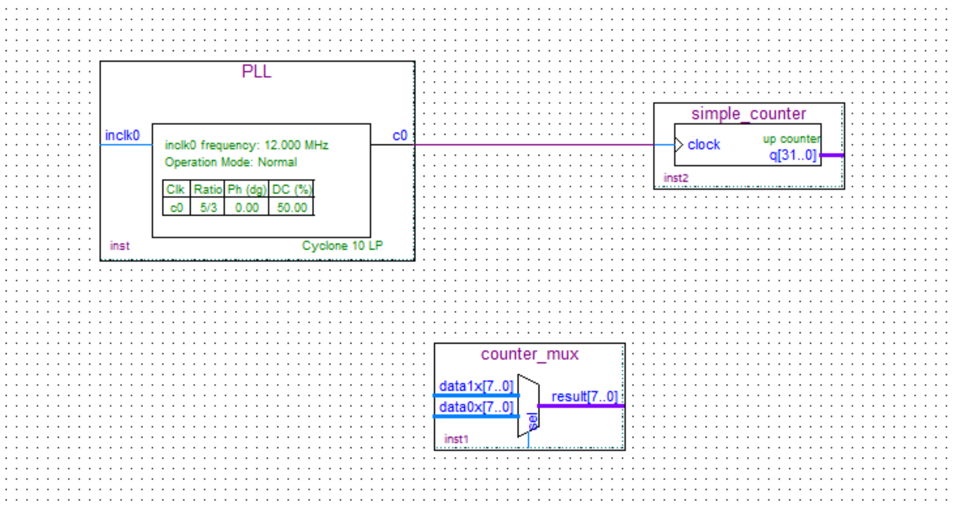
5.2.9 Connecting the Components

Next step is to make the proper connections between the three components we just added to the schematic.

5.2.9.1 Select the **"Node Tool"**.

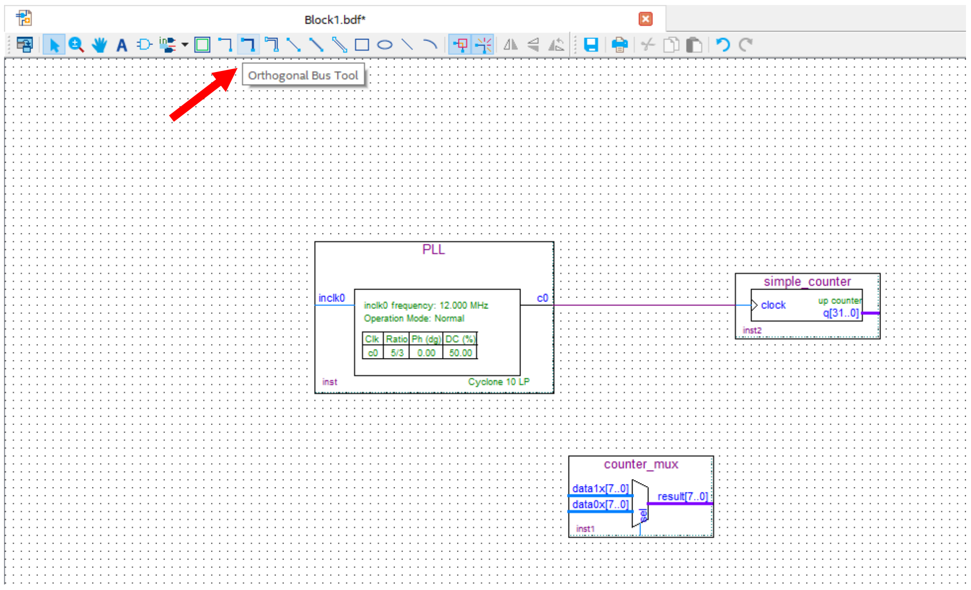


5.2.9.2 Connect the c0 of the PLL to the simple_counter as shown below:

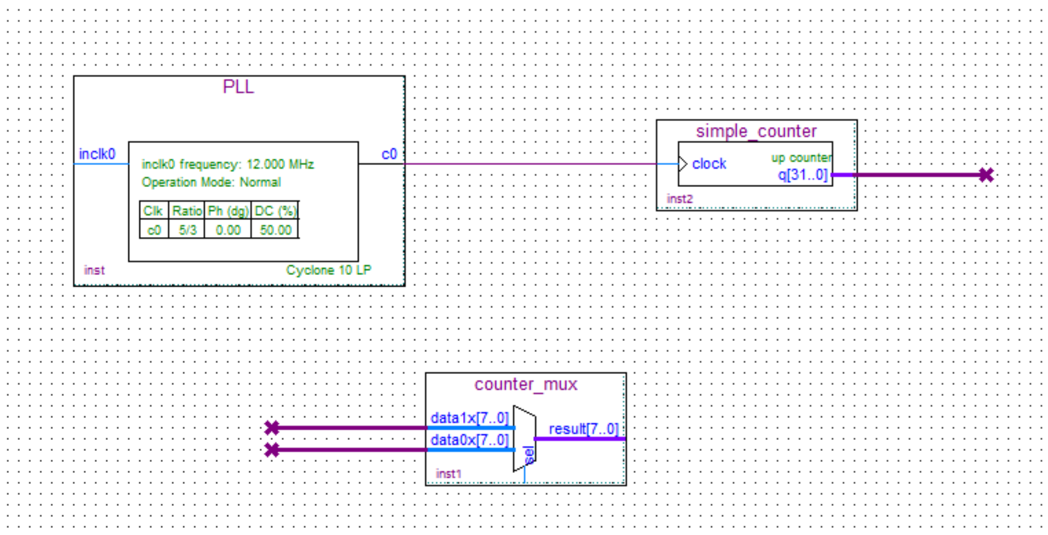


This will mean that a single signal (c0) is connected to the simple_counter (clock).

5.2.9.3 Select the **"Bus Tool"**.



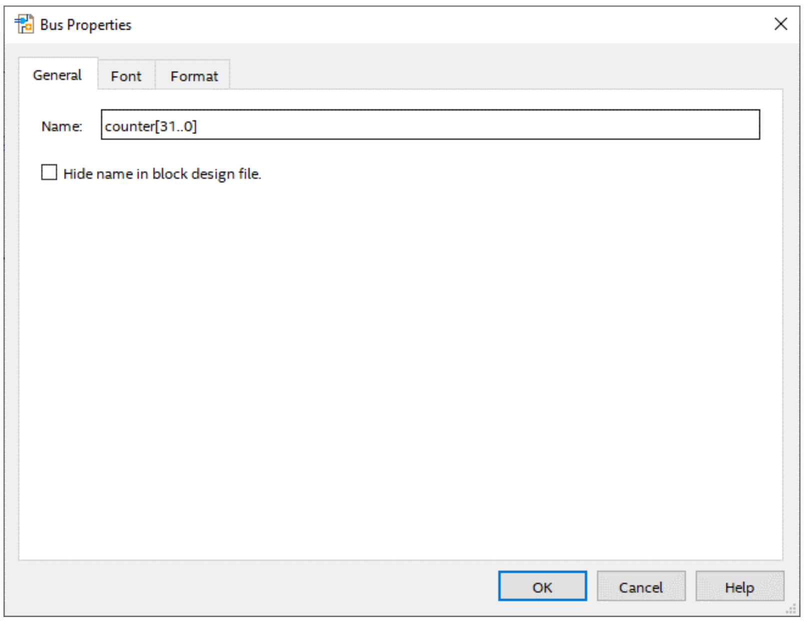
5.2.9.4 Using the bus tool create a connection coming out of the simple_counter and one connection for each of the inputs of the counter_mux as show below.



5.2.9.5 Right click on the output bus of the simple counter that you just created and select **"Properties"**.

Set the name of the bus to: **counter[31..0]**

The view of the "Bus Properties" should look like this:



Name: counter[31..0]

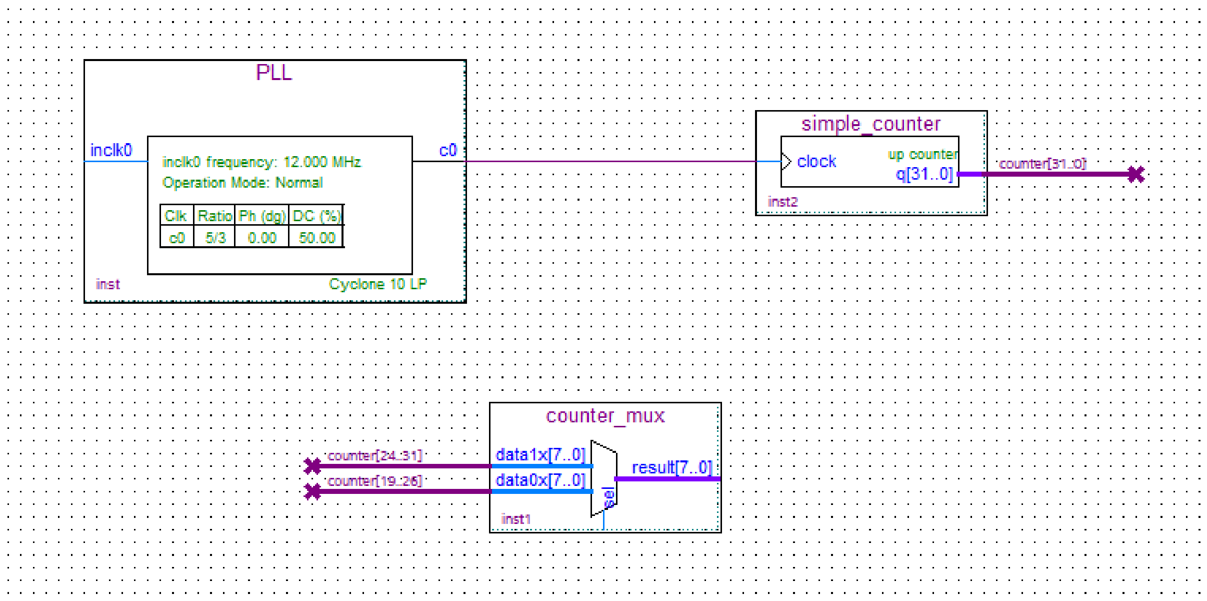
☐ Hide name in block design file.

5.2.9.6 Click "OK".

5.2.9.7 Do the same for input buses of the mux:

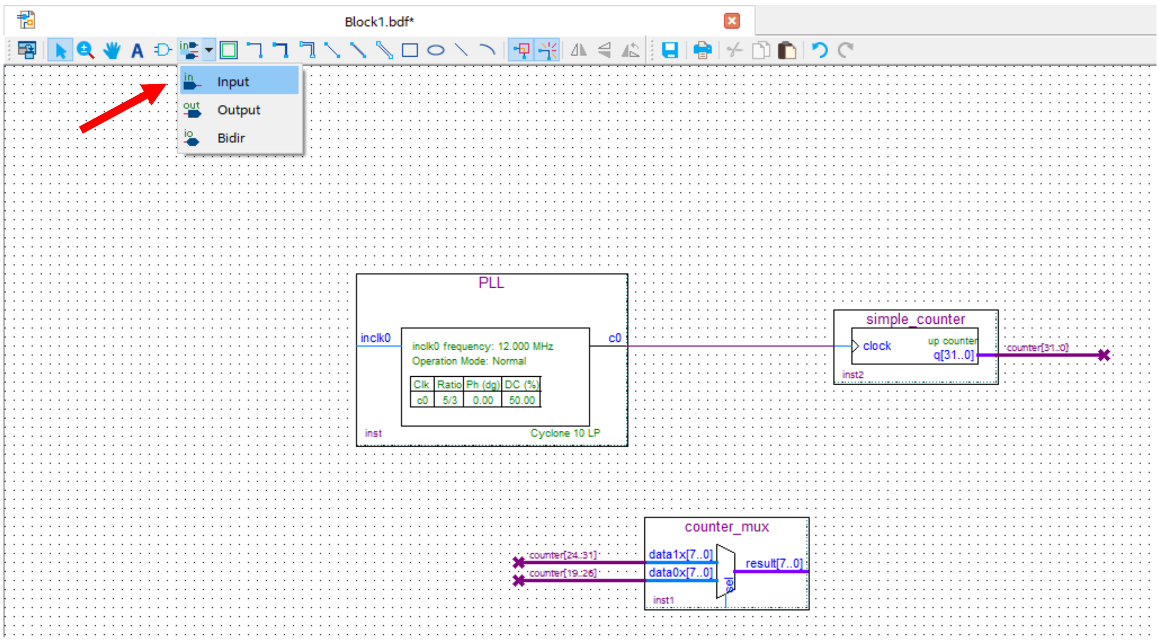
Name the top bus input:	**data1x[7..0] → counter[24..31]**
Name the bottom bus input:	**data0x[7..0] → counter[19..26]**

Schematic should look like this:



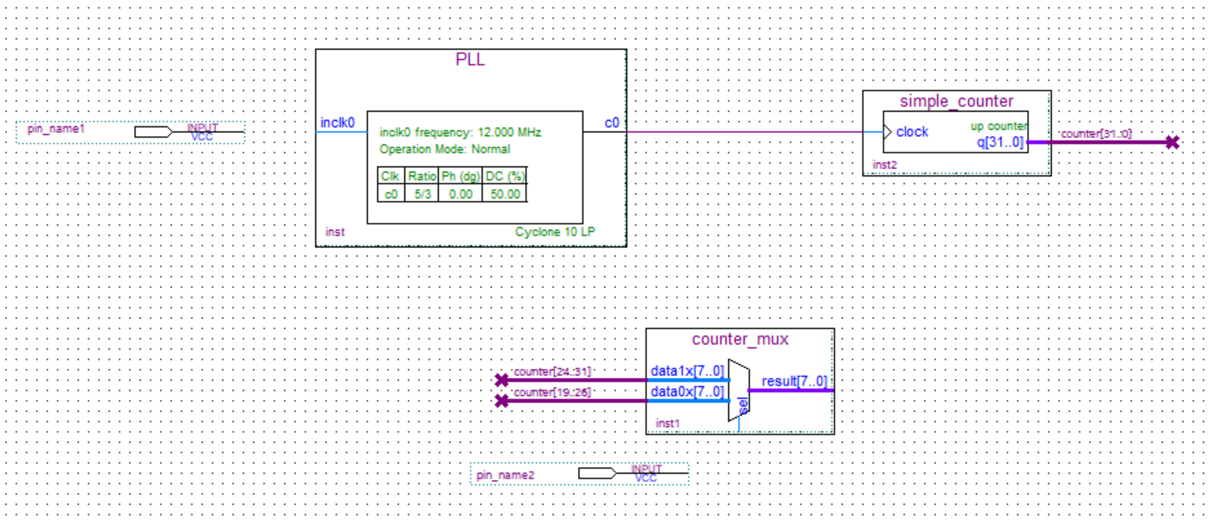
5.2.10 Add inputs, outputs to the schematic

5.2.10.1 Click on the **"Pin Tool"** as show below and select **"Input"**.



5.2.10.2 Add one input pin for inclk0 of the PLL and add other one input pin for sel of counter_mux.

Your schematic should look like this:



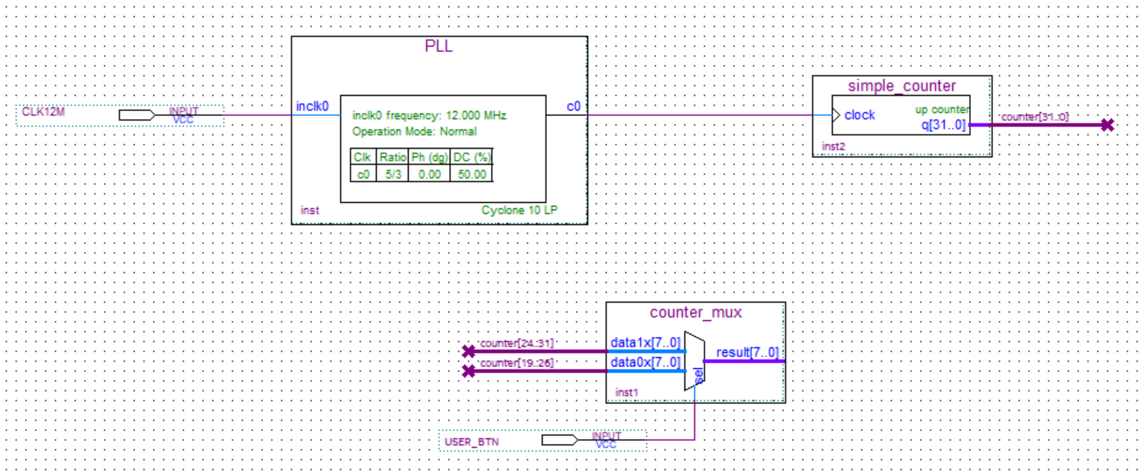
5.2.10.3 Rename the pin_name1 to **CLK12M** by double clicking its current name. This is going to be the clock signal coming into the FPGA.

5.2.10.4 Rename the pin_name2 to **USER_BTN** by double clicking its current name. This is going to be the user button of the CYC1000 board to select the mux.

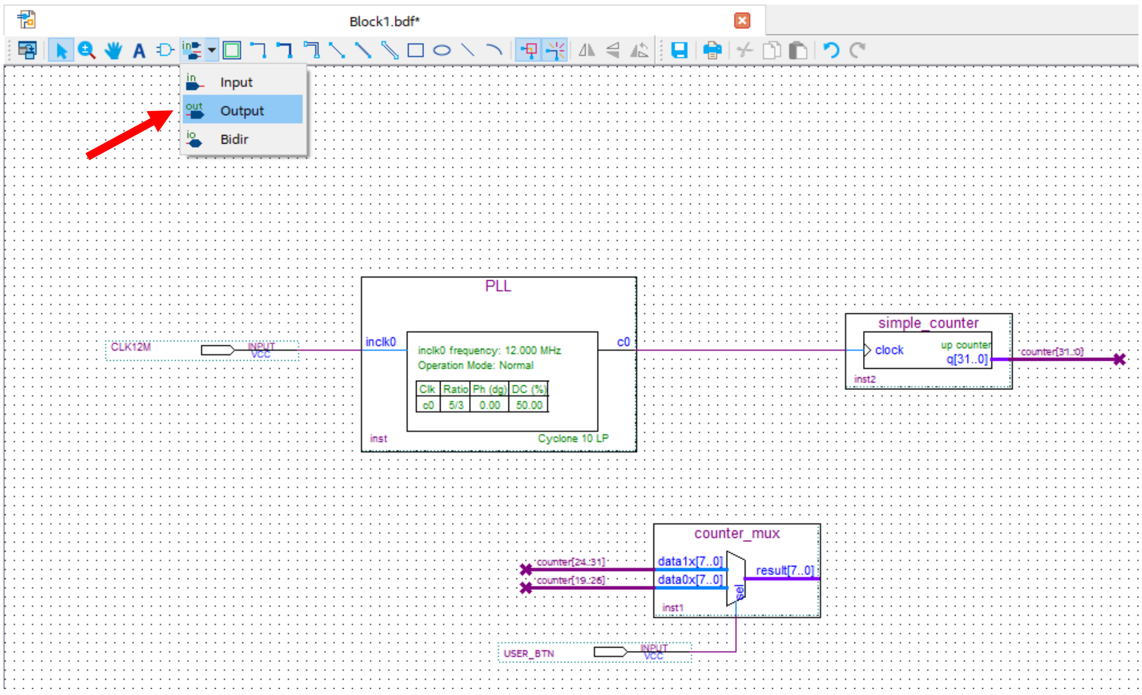
5.2.10.5 Using the **"Node Tool"** connect:

CLK12M → **inclk0** (of the PLL component)
USER_BTN → **sel** (of the counter_mux component)

Your schematic should look like this now:



5.2.10.6 Click on the **“Pin Tool”** as before, but this time select **“Output”**.



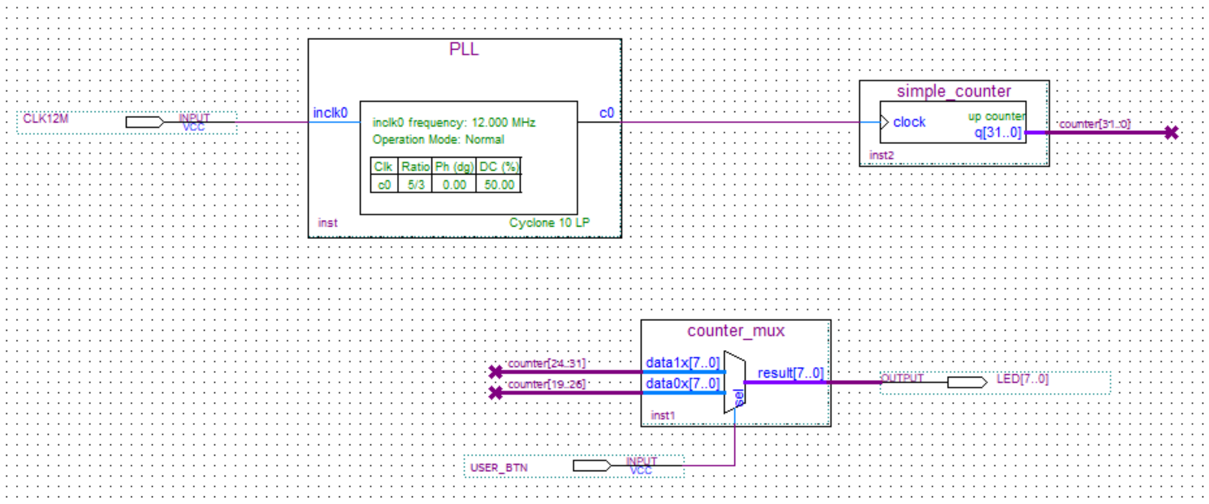
5.2.10.7 Add one output pin for the LEDs.

5.2.10.8 Rename the pin to **LED[7..0]**.

5.2.10.9 Using the **“Bus Tool”**, make the connection between counter_mux component and output pin:

result[7..0] → LED[7..0]

The final schematic should look like the following:



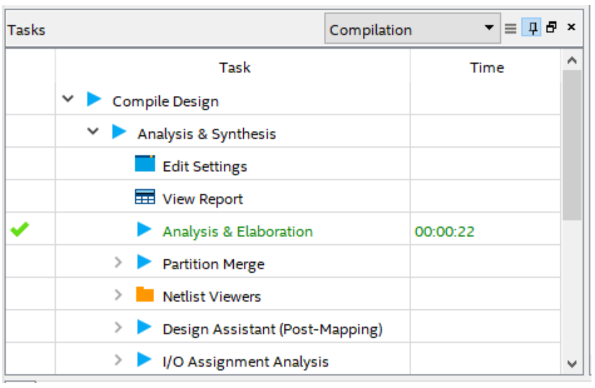
Looking at the schematic, even though the buses are not connected together by wires, the names of counter tell Quartus Prime to connect the signals together. Overall, the user button will toggle between displaying higher 8 bits of the counter and 8 lower bits of the counter. The signals of the counter that are not connected will not be used by Quartus Prime.

5.2.10.10 Save your design. Open the File Menu and select **"Save"**. Save it as **top.bdf**

5.2.11 Analysis and Synthesis

The next step is to run Analysis and Synthesis to ensure that there are no errors in the design. To run Analysis and synthesis open **Processing → Start → Analysis and Synthesis** or from clicking ✓ button on the top toolbar.

There should be no errors. If there are errors, they should be fixed before continuing and Analysis and Synthesis run again.



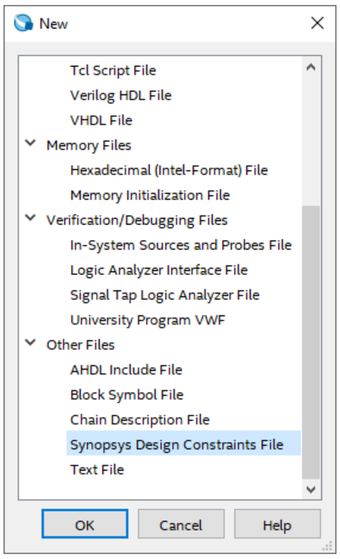
Tasks — Compilation

	Task	Time
	Compile Design	
	Analysis & Synthesis	
	Edit Settings	
	View Report	
✓	Analysis & Elaboration	00:00:22
	Partition Merge	
	Netlist Viewers	
	Design Assistant (Post-Mapping)	
	I/O Assignment Analysis	

5.2.12 Adding Timing Constraints

Timing Constraints tell the Quartus what are the timing requirements for this design. Timing Constraints are required in every CPLD/FPGA design.

5.2.12.1 To add the timing constraints, select **File → New** and under the "Other File" section, select "Synopsys Design Constraints File" and select "OK".


New
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes File
Logic Analyzer Interface File
Signal Tap Logic Analyzer File
University Program VWF
Other Files
AHDL Include File
Block Symbol File
Chain Description File
Synopsys Design Constraints File
Text File
OK
Cancel
Help

5.2.12.2 Type or copy the following lines into this new file:

```
#create input clock which is 12MHz
create_clock -name CLK12M -period 83.333 [get_ports {CLK12M}]

#derive PLL clocks
derive_pll_clocks

#derive clock uncertainty
derive_clock_uncertainty

#set false path
set_false_path -from [get_ports {USER_BTN}]
set_false_path -from * -to [get_ports {LED*}]
```

The first line “create_clock” tells Quartus Prime that the clock, CLK12M is 83.333 ns (12 MHz). It also assigns the CLK12M to a pin (port) in the .sdc format.

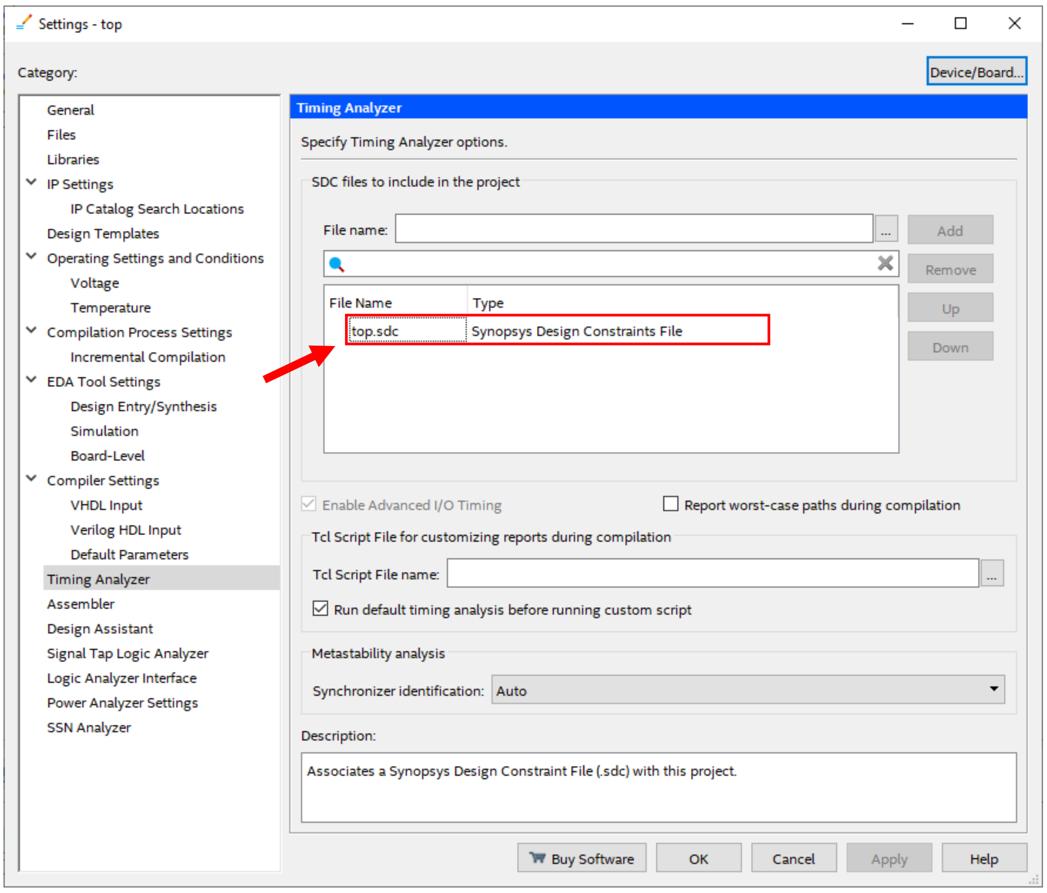
The second line “derive_pll_clocks” tells the software to look if there are any PLLs, and if so, automatically derive the clock multiplication/division of the outputs of the PLL even if they are used internally within the CPLD/FPGA.

The third line “derive_clock_uncertainty” tells the software to automatically determine the internal clock uncertainty. No clock is ideal, and thus there will be some internal jitter within the FPGA associated with it.

The fourth and fifth line “set_false_path” tells the software to not do any timing optimization to the stated paths/pins. The I/Os of this design are trivial, so they can be ignored in the Timing Analysis.

5.2.12.3 Use **File → Save** to save it as **top.sdc**.

5.2.12.4 Ensure that the file is added to the Project: **Assignments → Settings** and select "Timing Analyzer". The top.sdc should have been already added by default. If it is not, it will need to be added manually.



Timing Analyzer

Specify Timing Analyzer options.

SDC files to include in the project

File name:

File Name	Type
top.sdc	Synopsys Design Constraints File

☑ Enable Advanced I/O Timing   ☐ Report worst-case paths during compilation

Tcl Script File for customizing reports during compilation

Tcl Script File name:

☑ Run default timing analysis before running custom script

Metastability analysis

Synchronizer identification: Auto

Description:

Associates a Synopsys Design Constraint File (.sdc) with this project.

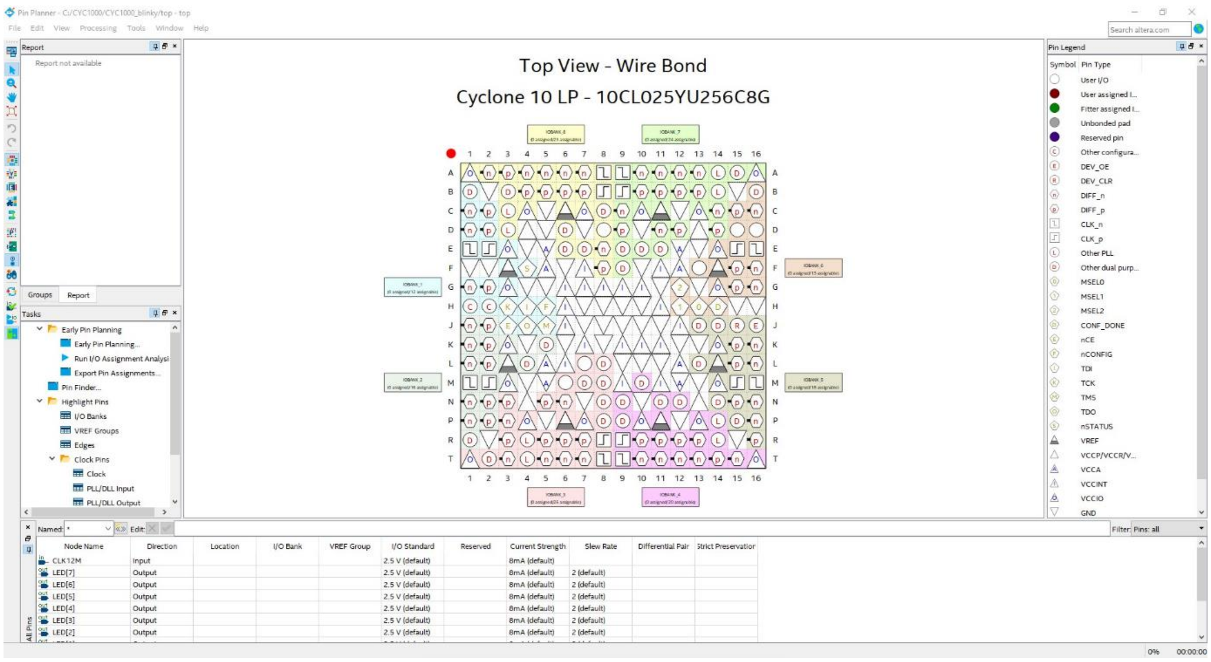
5.2.13 Pinning Assignments

Before the design can be downloaded to the FPGA, pin assignments that match the hardware on the board are needed. There are different ways to do this such as the Pin Planner, Assignment Editor, and text files.

The following steps will show one of these ways, the Pin Planner. Since there are only 10 pins that need to be assigned, the Pin Planner can be used. If many pins are needed, other ways can be used such as the Quartus Assignment Editor, or by importing constraints from a text file or spreadsheet.

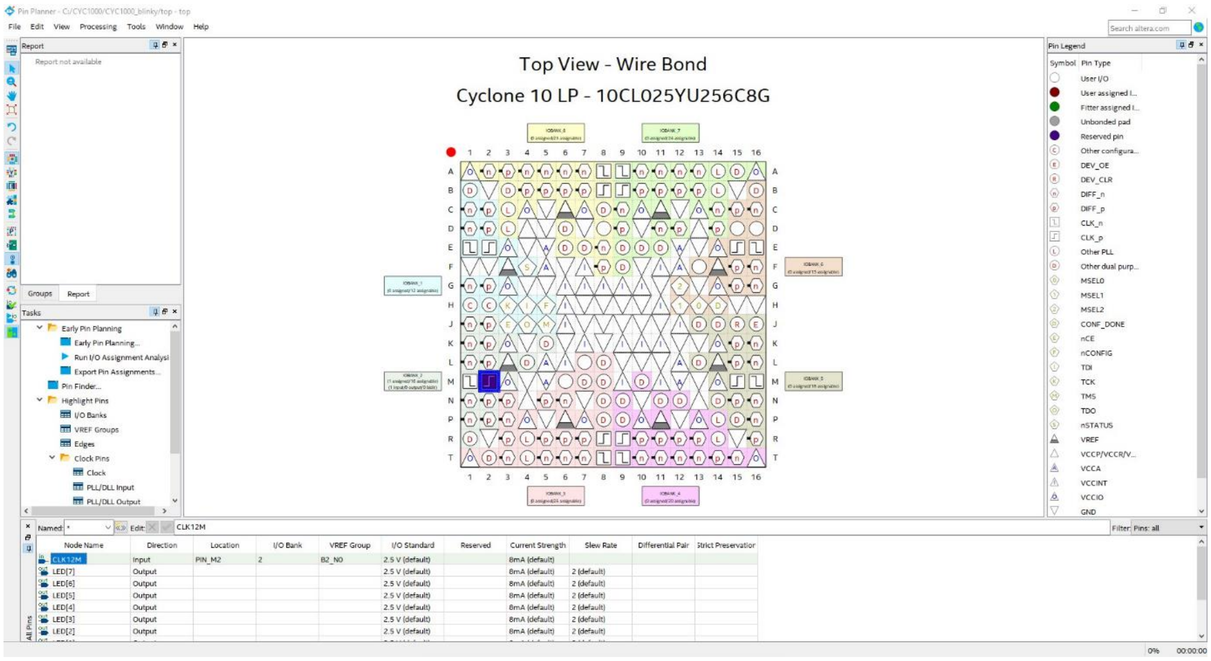
5.2.13.1 Open the Pin Planner: **Assignments → Pin Planner**.

A new window will open as seen below:



Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
CLK12M	Input				2.5 V (default)		8mA (default)			
LED[7]	Output				2.5 V (default)		8mA (default)	2 (default)		
LED[6]	Output				2.5 V (default)		8mA (default)	2 (default)		
LED[5]	Output				2.5 V (default)		8mA (default)	2 (default)		
LED[4]	Output				2.5 V (default)		8mA (default)	2 (default)		
LED[3]	Output				2.5 V (default)		8mA (default)	2 (default)		
LED[2]	Output				2.5 V (default)		8mA (default)	2 (default)		

5.2.13.2 To make pin assignments, select the CLK12M (node name) on the bottom portion and drag and drop it to pin M2 of the Top View of the FPGA or alternatively set the Location field of the CLK12M to **PIN_M2**.



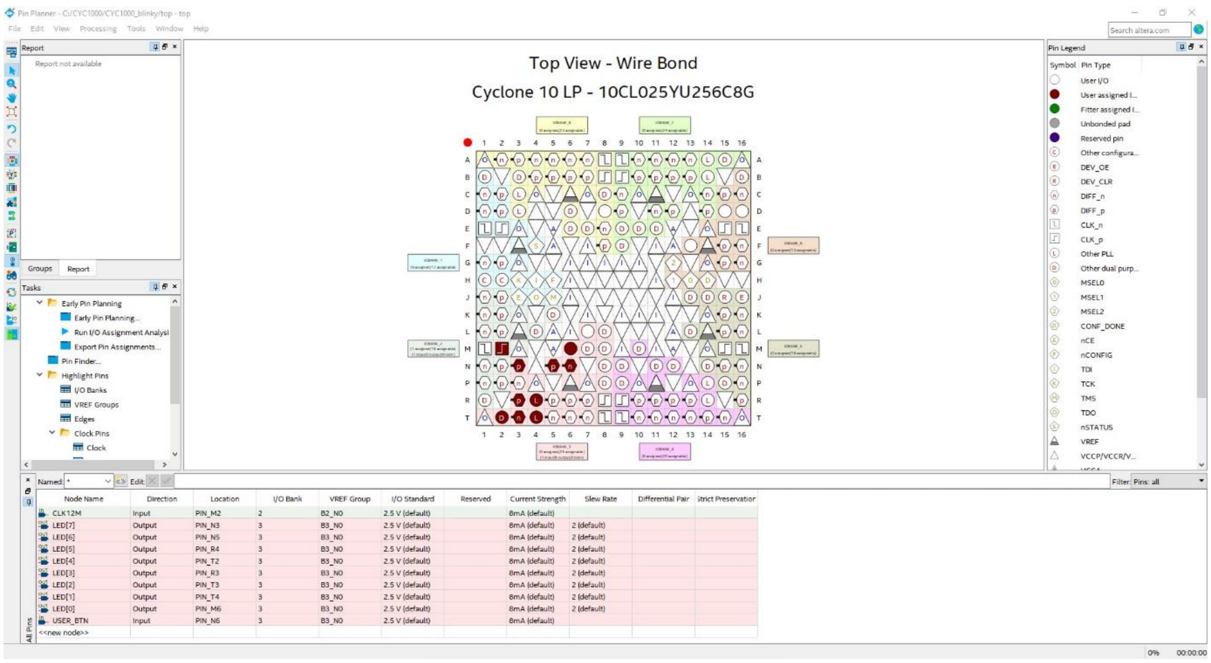
Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
CLK12M	Input	PIN_M2	2	B2_N0	2.5 V (default)		8mA (default)			
LED[7]	Output				2.5 V (default)		8mA (default)	2 (default)		
LED[6]	Output				2.5 V (default)		8mA (default)	2 (default)		
LED[5]	Output				2.5 V (default)		8mA (default)	2 (default)		
LED[4]	Output				2.5 V (default)		8mA (default)	2 (default)		
LED[3]	Output				2.5 V (default)		8mA (default)	2 (default)		
LED[2]	Output				2.5 V (default)		8mA (default)	2 (default)		

Note that the Location of the CLK12M is now set to Location PIN_M2 (as seen in blue colour in the top view of the FPGA).

5.2.13.3 The other pins need to be assigned as well. Just like previously set all the pins to their appropriate locations using the table below, by either drag and drop or writing manually the location.

Node Name	Pin Location
LED[7]	PIN_N3
LED[6]	PIN_N5
LED[5]	PIN_R4
LED[4]	PIN_T2
LED[3]	PIN_R3
LED[2]	PIN_T3
LED[1]	PIN_T4
LED[0]	PIN_M6
USER_BTN	PIN_N6

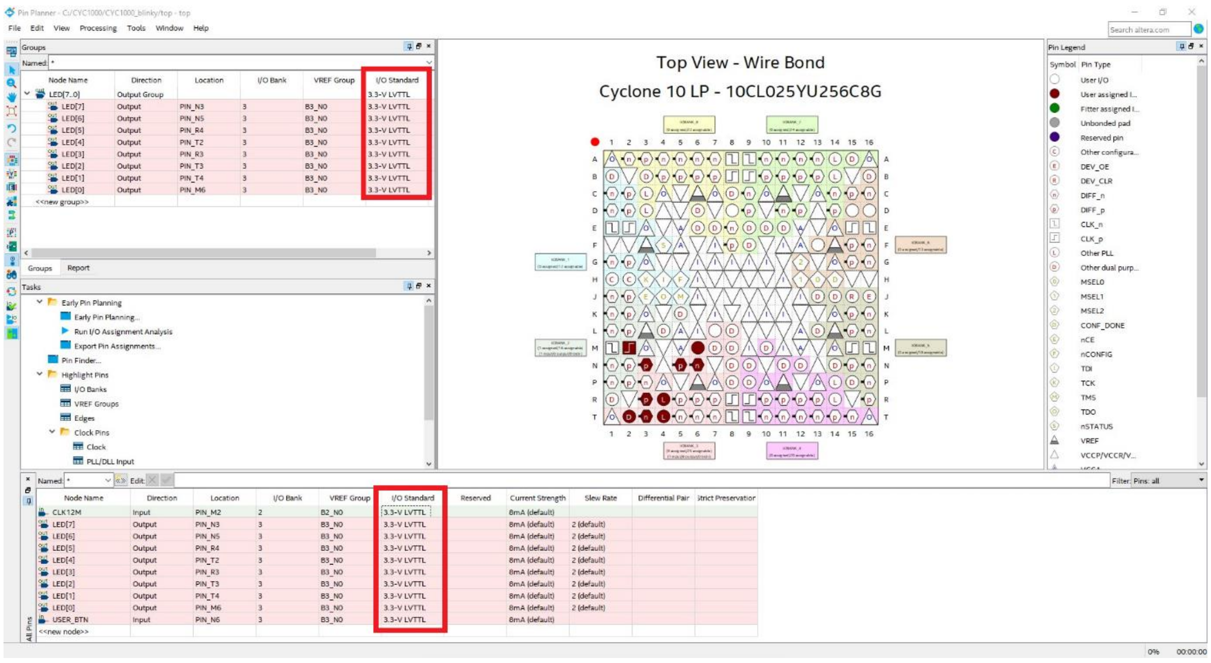
5.2.13.4 Now the Pin Planner should look like this after assigning all the pin locations.



Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
CLK12M	Input	PIN_M2	2	B2_N0	2.5 V (default)		8mA (default)			
LED[7]	Output	PIN_N3	3	B3_N0	2.5 V (default)		8mA (default)	2 (default)		
LED[6]	Output	PIN_N5	3	B3_N0	2.5 V (default)		8mA (default)	2 (default)		
LED[5]	Output	PIN_R4	3	B3_N0	2.5 V (default)		8mA (default)	2 (default)		
LED[4]	Output	PIN_T2	3	B3_N0	2.5 V (default)		8mA (default)	2 (default)		
LED[3]	Output	PIN_R3	3	B3_N0	2.5 V (default)		8mA (default)	2 (default)		
LED[2]	Output	PIN_T3	3	B3_N0	2.5 V (default)		8mA (default)	2 (default)		
LED[1]	Output	PIN_T4	3	B3_N0	2.5 V (default)		8mA (default)	2 (default)		
LED[0]	Output	PIN_M6	3	B3_N0	2.5 V (default)		8mA (default)	2 (default)		
USER_BTN	Input	PIN_N6	3	B3_N0	2.5 V (default)		8mA (default)			
<<new node>>										

5.2.13.5 The specific pins are now selected, but the I/O standards now need to be set as well. The button, LEDS, and clock pins are the same I/O standard for CYC1000 since all banks and peripherals are powered by 3.3V. The USER_BTN, the LEDs and clock pins are 3.3-V LVTTL. These I/O standards can be set in the Pin Planner, by selecting the I/O Standard. Select the I/O standard either from the “All Pins” tab or the “Groups” tab and change the 2.5V (default) to the specific I/O standard.

The Pin Planner should now look like this:



Named: *

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard
LED[7..0]	Output Group				3.3-V LVTTL
LED[7]	Output	PIN_N3	3	B3_N0	3.3-V LVTTL
LED[6]	Output	PIN_N5	3	B3_N0	3.3-V LVTTL
LED[5]	Output	PIN_R4	3	B3_N0	3.3-V LVTTL
LED[4]	Output	PIN_T2	3	B3_N0	3.3-V LVTTL
LED[3]	Output	PIN_R3	3	B3_N0	3.3-V LVTTL
LED[2]	Output	PIN_T3	3	B3_N0	3.3-V LVTTL
LED[1]	Output	PIN_T4	3	B3_N0	3.3-V LVTTL
LED[0]	Output	PIN_M6	3	B3_N0	3.3-V LVTTL
<<new group>>					

Named: *

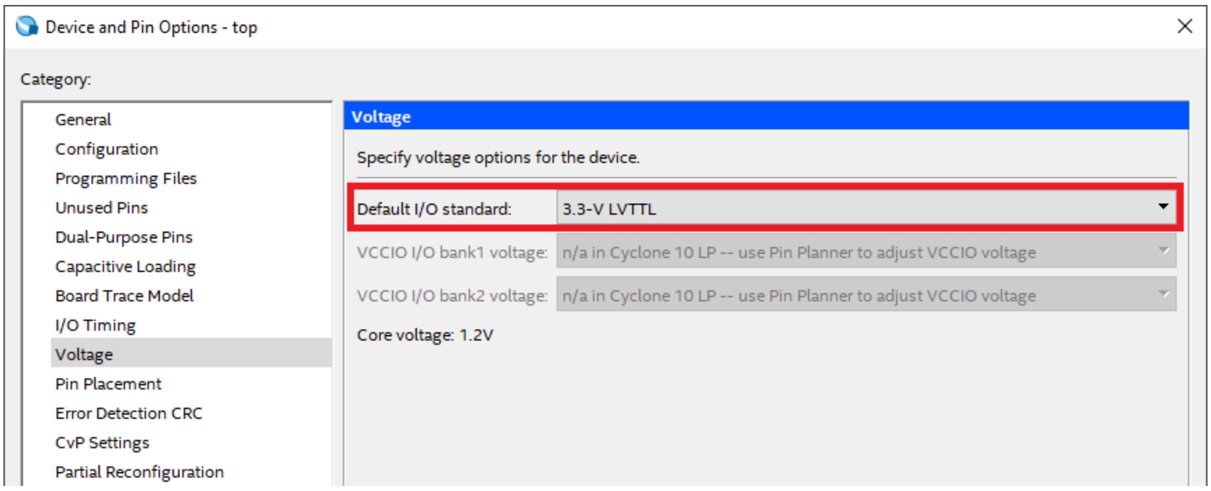
Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
CLK12M	Input	PIN_M2	2	B2_N0	3.3-V LVTTL		8mA (default)			
LED[7]	Output	PIN_N3	3	B3_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[6]	Output	PIN_N5	3	B3_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[5]	Output	PIN_R4	3	B3_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[4]	Output	PIN_T2	3	B3_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[3]	Output	PIN_R3	3	B3_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[2]	Output	PIN_T3	3	B3_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[1]	Output	PIN_T4	3	B3_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[0]	Output	PIN_M6	3	B3_N0	3.3-V LVTTL		8mA (default)	2 (default)		
USER_BTN	Input	PIN_N6	3	B3_N0	3.3-V LVTTL		8mA (default)			
<<new node>>										

5.2.13.6 Close the Pin Planner. The settings are automatically saved.

5.2.14 Compiling the Design

5.2.14.1 You can set the default I/O Standard which can eliminate some design warning and save you time from setting the standard for some pins manually.

Open **Assignments → Device → Device and Pin Options → Voltage** and set Default I/O Standard to **"3.3-V LVTTL"** and press "OK" to all the windows.



Specify voltage options for the device.

Default I/O standard: 3.3-V LVTTL

VCCIO I/O bank1 voltage: n/a in Cyclone 10 LP -- use Pin Planner to adjust VCCIO voltage

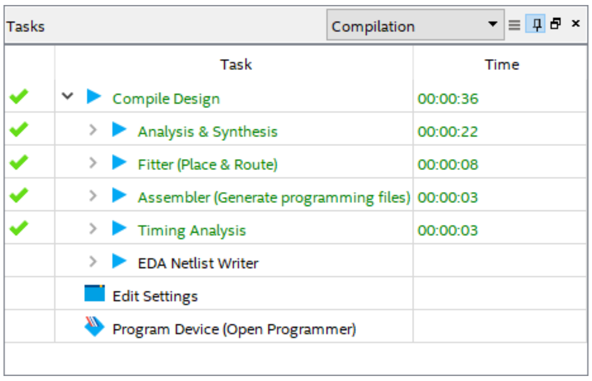
VCCIO I/O bank2 voltage: n/a in Cyclone 10 LP -- use Pin Planner to adjust VCCIO voltage

Core voltage: 1.2V

The next step is to compile and complete the design. This step will verify that there are no errors, create internal databases, and create programming files that will be used in the next steps.

5.2.14.2 To compile the design, select **Processing → Start Compilation** or push the ▶ button on the toolbar.

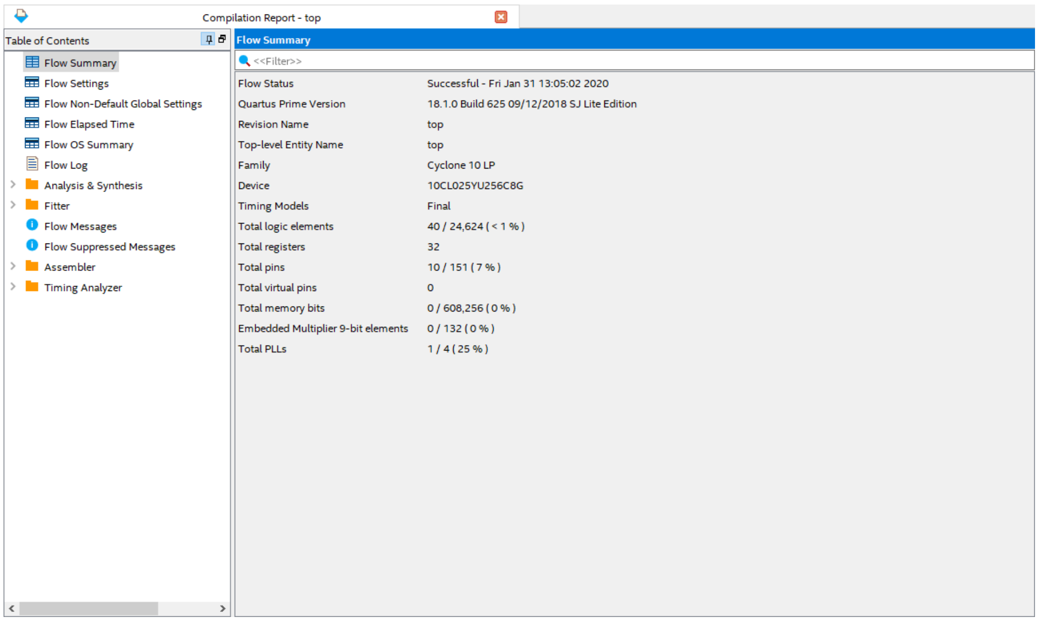
If there are errors, they will need to be resolved and re-compiled before the design can be programmed to the board. When Compiling finishes and there are no errors, there will be a message at the bottom of the window that states: Full Compilation was successful and a 100% indication along with the compile time in the right bottom corner.



	Task	Time
✓	Compile Design	00:00:36
✓	Analysis & Synthesis	00:00:22
✓	Fitter (Place & Route)	00:00:08
✓	Assembler (Generate programming files)	00:00:03
✓	Timing Analysis	00:00:03
	EDA Netlist Writer	
	Edit Settings	
	Program Device (Open Programmer)	

5.2.15 Reading the Compilation Report

After successfully compiling the design, a Compilation Report should appear as shown above:



Flow Status	Successful - Fri Jan 31 13:05:02 2020
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	top
Top-level Entity Name	top
Family	Cyclone 10 LP
Device	10CL025YU256C8G
Timing Models	Final
Total logic elements	40 / 24,624 (< 1 %)
Total registers	32
Total pins	10 / 151 (7 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	1 / 4 (25 %)

This report is very useful with a lot of information about the design. Last message state that the design was fully constrained, Timing Analysis and compilation successful, but there is more to it:

- In the Flow Summary, it can be seen how many logic elements the whole design took, along with total PLLs, registers, pins, etc.

- In Analysis and Synthesis, more detailed information about the resources used can be seen in Resource Usage Summary, as well how many LEs were used for each component in Resource Utilization by Entity.

- In the Fitter, more detailed information about the pins and their banks can be seen.

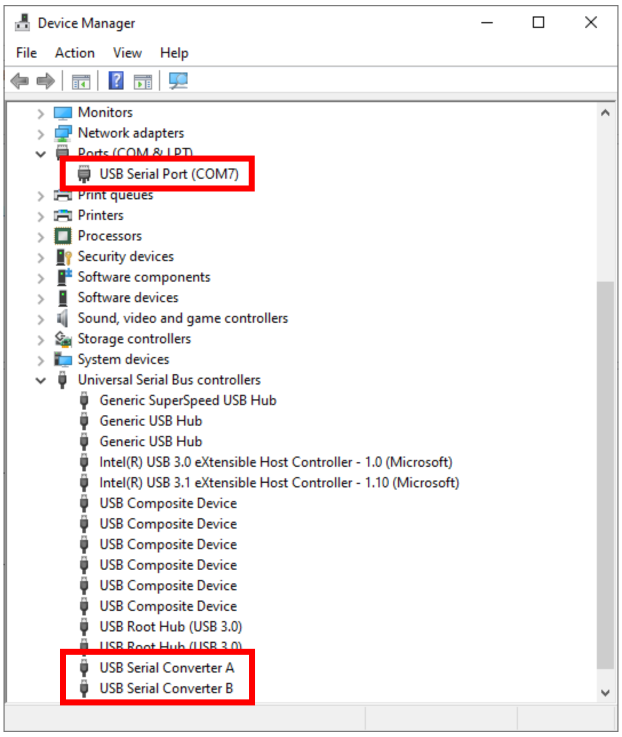
- Timing Analyzer shows various timing information concerning the design, as well as if the design has met the timing requirements. In this case timing requirements were met, but in other cases that requirements might not be met, could be solved by going over the information provided in the reports inside this folder. Most notable reports in this folder are the maximum frequency the design can achieve, setup and hold slack, unconstrained paths in case they were missed, etc.

Chapter 6 - Configuring the CYC1000

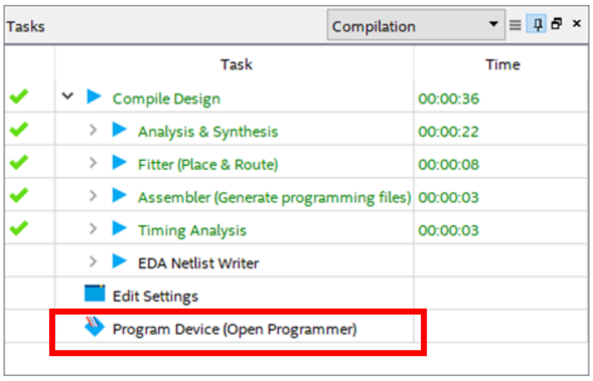
After successfully compiling your project, there should new files generated. In case of Cyclone 10 LP devices, only the .sof file is generated automatically.

6.1 Configure the FPGA in JTAG mode

6.1.1 Connect your CYC1000 board to your PC using an USB cable. Since the Arrow USB Blaster should be already installed, the Window's Device Manager should display the following entries are highlighted in red (port number may differ depending on your PC). If the Arrow USB Blaster is not installed, please refer to Chapter 4.2 for installing the drivers.

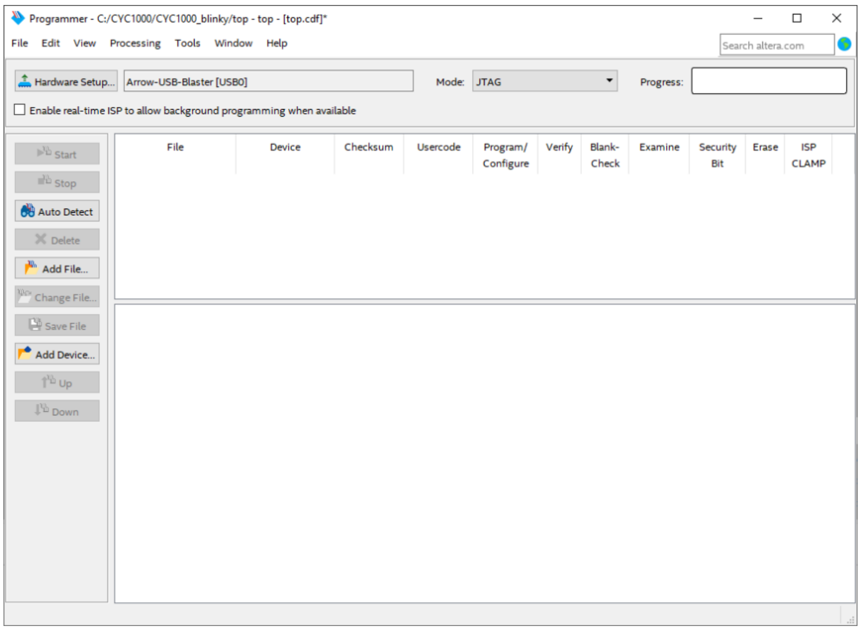


6.1.2 Open the Quartus Prime Programmer from **Tools → Programmer** or double-click on Program Device (Open Programmer) from the Tasks pane.



	Task	Time
✓	Compile Design	00:00:36
✓	Analysis & Synthesis	00:00:22
✓	Fitter (Place & Route)	00:00:08
✓	Assembler (Generate programming files)	00:00:03
✓	Timing Analysis	00:00:03
	EDA Netlist Writer	
	Edit Settings	
	Program Device (Open Programmer)	

6.1.3 The programmer should add the programming file automatically. After opening the program this should be the view of the new window:

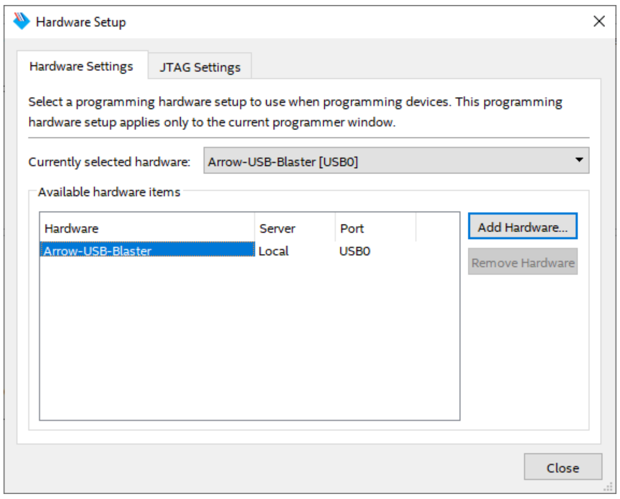


Programmer - C:/CYC1000/CYC1000_blinky/top - top - [top.cdf]*

Hardware Setup... Arrow-USB-Blaster [USB0] Mode: JTAG Progress:

☐ Enable real-time ISP to allow background programming when available

6.1.4 Click **Hardware Setup…** and double click **Arrow-USB-Blaster** entry in the Hardware Setup tab. The Currently selected hardware should now show Arrow-USB-Blaster [USB0] (depending on your PC, the USB port number may variant).



Hardware Setup

Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

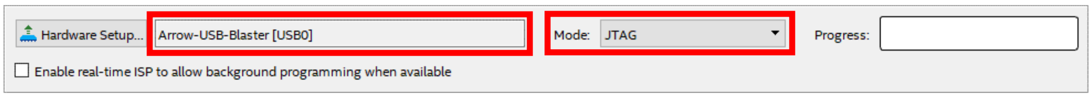
Currently selected hardware: Arrow-USB-Blaster [USB0]

Available hardware items

Hardware	Server	Port
Arrow-USB-Blaster	Local	USB0

6.1.5 Click “Close”.

6.1.6 Make sure the hardware setup is Arrow-USB-Blaster [USB0] and the mode is JTAG. If the Mode is not set to JTAG, click on it and select JTAG from the drop-down menu.

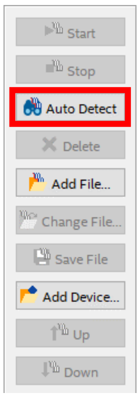


Hardware Setup... Arrow-USB-Blaster [USB0] Mode: JTAG Progress:

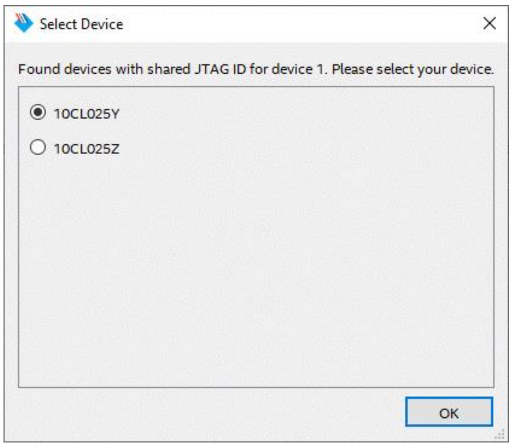
☐ Enable real-time ISP to allow background programming when available

6.1.7 If the configuration has been added by default, you can skip the following steps and continue with the 6.1.12 point.

6.1.8 Click “Auto Detect” on the left side of the Programmer.



6.1.9 Select **10CL025Y** device and click “OK” on the Select Device window.

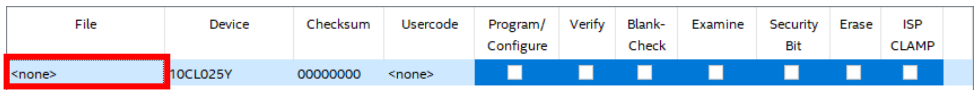


Found devices with shared JTAG ID for device 1. Please select your device.

[✓ 10CL025Y]

10CL025Z

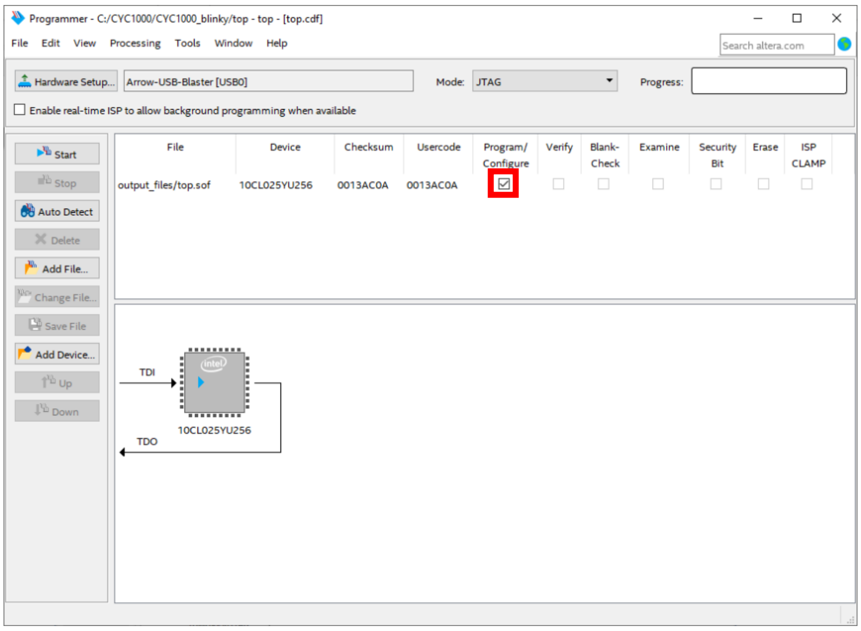
6.1.10 Double click <none> to choose programming file.



File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
<none>	10CL025Y	00000000	<none>	☐	☐	☐	☐	☐	☐	☐

6.1.11 Navigate to **<project_directory>/output_files/** in your compilation directory. Select and open the **top.sof** file.

6.1.12 Make sure the Programmer shows the correct file and correct part in the JTAG chain and check the Program/Configure checkbox.


Programmer - C:/CYC1000/CYC1000_blinky/top - top - [top.cdf]

Hardware Setup... Arrow-USB-Blaster [USB0] Mode: JTAG Progress:

☐ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
output_files/top.sof	10CL025YU256	0013AC0A	0013AC0A	☑	☐	☐	☐	☐	☐	☐

6.1.13 Click Start to program the CYC1000. When the configuration is complete, the Progress bar should reach 100% (Successful).

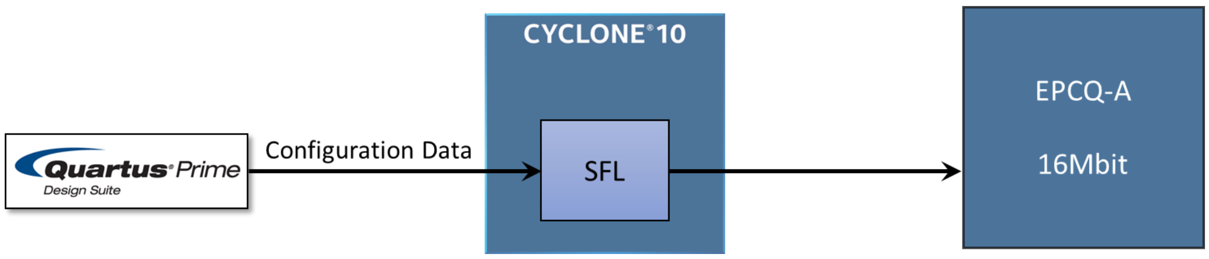
Progress: 100% (Successful)

The design is now programmed to the FPGA.

***Note:** that turning off and then on the FPGA will result into losing its configuration.*

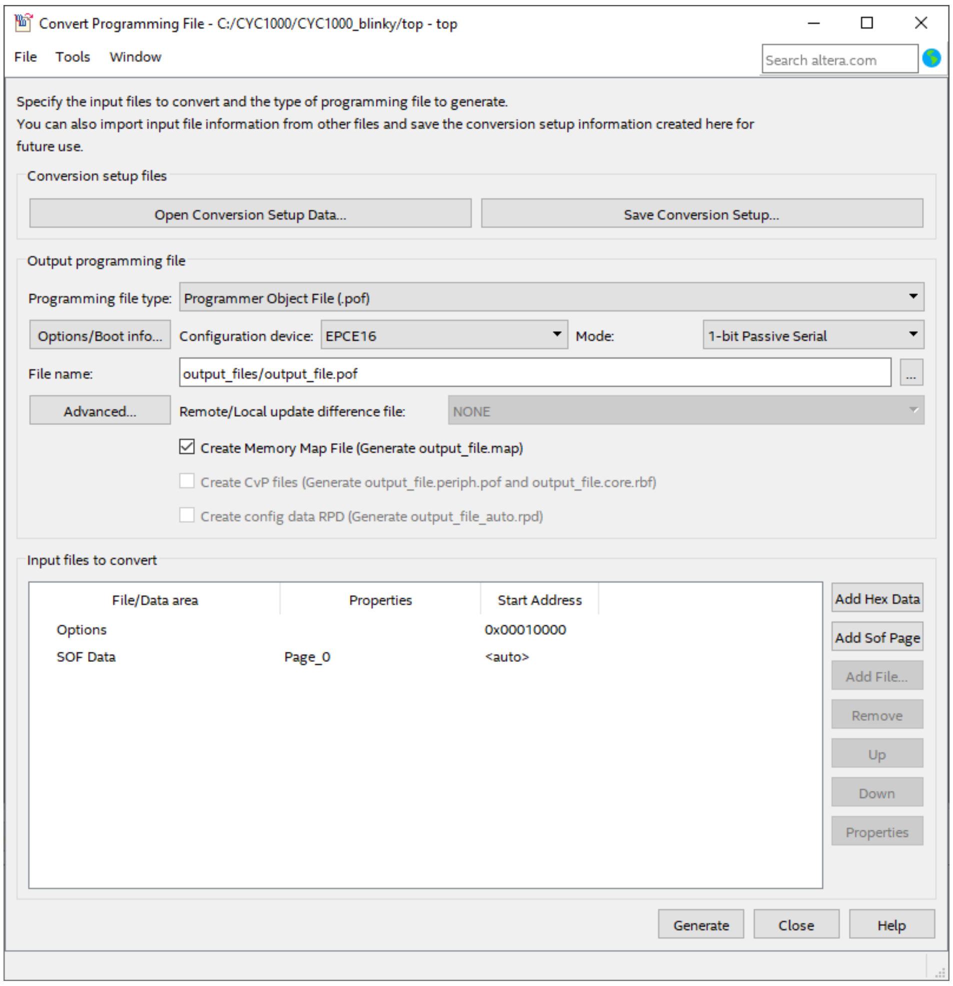
6.2 Serial configuration flash memory programming

The configuration data to be written to EPCQ-A will be part of the JTAG indirect configuration file (.jic). This configuration data is automatically loaded from the serial configuration flash into the Cyclone 10 LP device when the board is powered up.



6.2.1 Programming File generation

6.2.1.1 In Quartus Prime, go to **File → Convert Programming Files...**



Convert Programming File - C:/CYC1000/CYC1000_blinky/top - top

Specify the input files to convert and the type of programming file to generate.
You can also import input file information from other files and save the conversion setup information created here for future use.

Conversion setup files

Output programming file

Programming file type: Programmer Object File (.pof)

Configuration device: EPCE16

Mode: 1-bit Passive Serial

File name: output_files/output_file.pof

Remote/Local update difference file: NONE

☑ Create Memory Map File (Generate output_file.map)

☐ Create CvP files (Generate output_file.periph.pof and output_file.core.rbf)

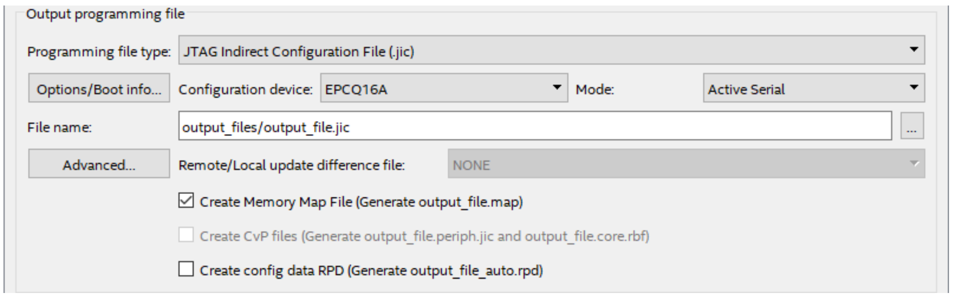
☐ Create config data RPD (Generate output_file_auto.rpd)

Input files to convert

File/Data area	Properties	Start Address
Options		0x00010000
SOF Data	Page_0	<auto>

6.2.1.2 Set the programming file type to **JTAG Indirect Configuration File (.jic)**.

6.2.1.3 Select **EPCQ16A** from the drop-down menu for configuration device and make sure that the **Active Serial** is set to mode. The output programming file settings should look like this:



Output programming file

Programming file type: JTAG Indirect Configuration File (.jic)

Configuration device: EPCQ16A

Mode: Active Serial

File name: output_files/output_file.jic

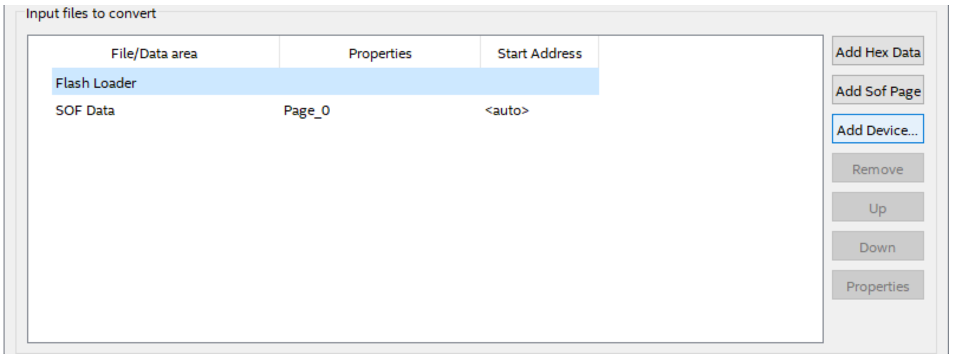
Remote/Local update difference file: NONE

☑ Create Memory Map File (Generate output_file.map)

☐ Create CvP files (Generate output_file.periph.jic and output_file.core.rbf)

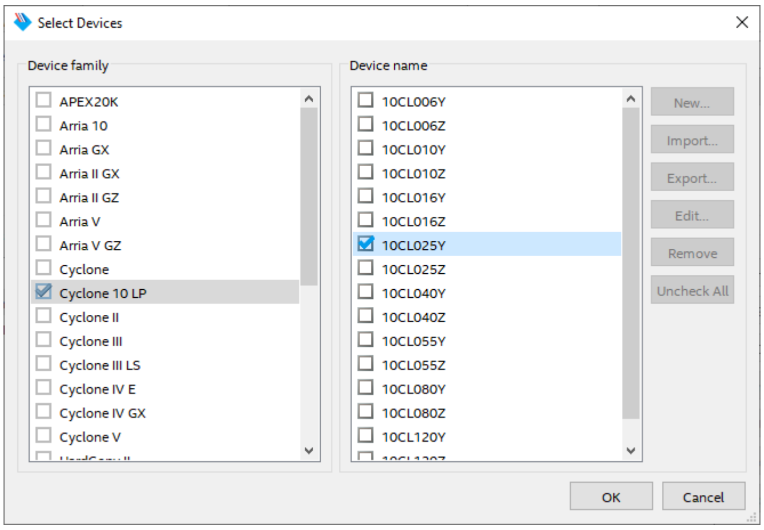
☐ Create config data RPD (Generate output_file_auto.rpd)

6.2.1.4 Select **Flash Loader** under Input files to covert settings and click on **Add Device…** button.



File/Data area	Properties	Start Address
Flash Loader		
SOF Data	Page_0	<auto>

6.2.1.5 On the new window select **Cyclone 10 LP** as Device family and **10CL025Y** as Device name.

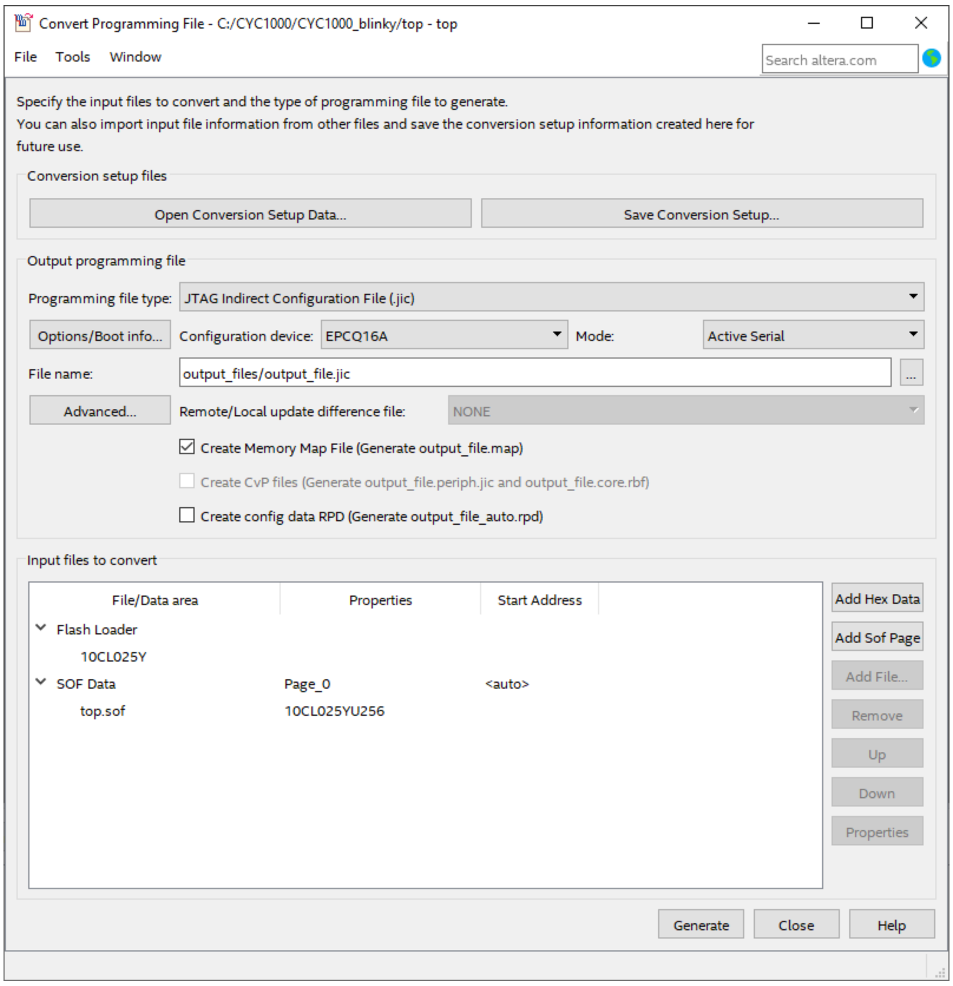


6.2.1.6 Click **OK** to add device to Flash Loader.

6.2.1.7 Select **SOF Data** under Input files to convert and click on **Add File…** button.

6.2.1.8 Go to **<project_directory>/output_files/** and open **top.sof**.

6.2.1.9 Make sure that your settings are same as the picture below and if everything is correct, click **Generate**.


Convert Programming File - C:/CYC1000/CYC1000_blinky/top - top

Specify the input files to convert and the type of programming file to generate.
You can also import input file information from other files and save the conversion setup information created here for future use.

Conversion setup files

Output programming file

Programming file type: JTAG Indirect Configuration File (.jic)

Configuration device: EPCQ16A

Mode: Active Serial

File name: output_files/output_file.jic

Remote/Local update difference file: NONE

☑ Create Memory Map File (Generate output_file.map)

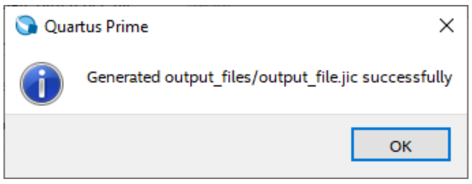
☐ Create CvP files (Generate output_file.periph.jic and output_file.core.rbf)

☐ Create config data RPD (Generate output_file_auto.rpd)

Input files to convert

File/Data area	Properties	Start Address
Flash Loader		
10CL025Y		
SOF Data	Page_0	<auto>
top.sof	10CL025YU256	

6.2.1.10 Click **OK** on the successful file generation notification and **close** Convert Programming File window.

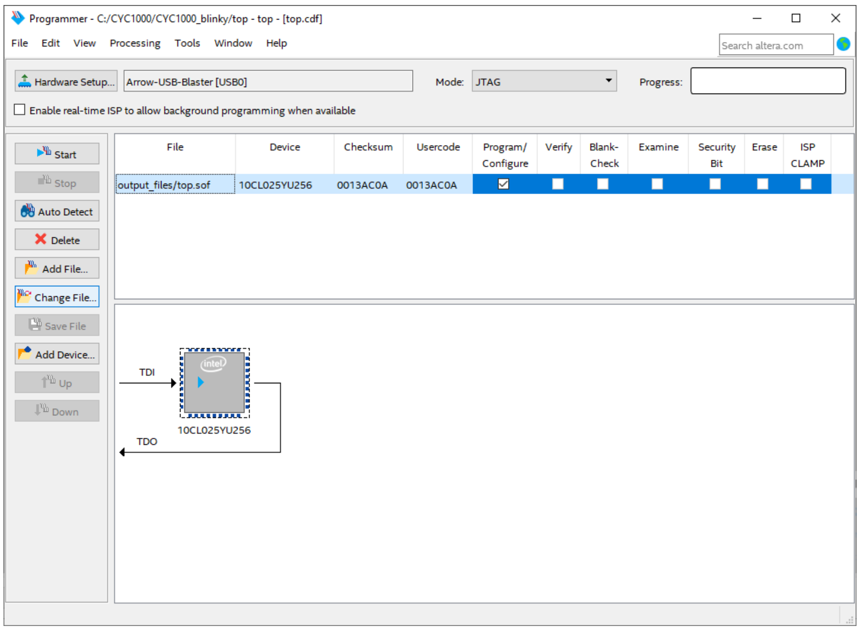

Quartus Prime

Generated output_files/output_file.jic successfully

6.2.2 Device Programming

6.2.2.1 Open Programmer.

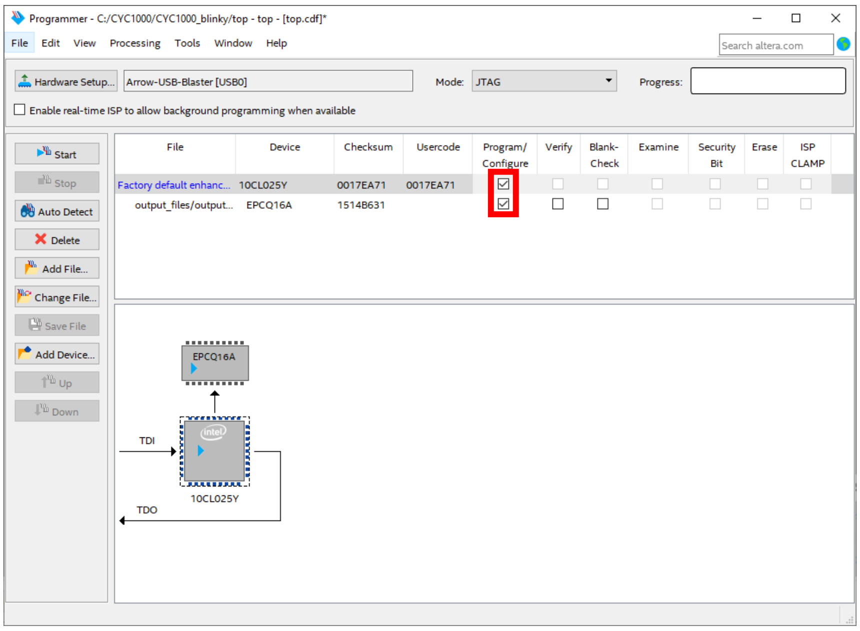
6.2.2.2 Select **output_files/top.sof** and click **Change File…** button.



File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
output_files/top.sof	10CL025YU256	0013AC0A	0013AC0A	☑	☐	☐	☐	☐	☐	☐

6.2.2.3 Go to **<project_directory>/output_files/** and open **output_file.jic**.
When you add the .jic file, the Programmer will automatically update the JTAG chain and put EPCQ-A flash memory.

6.2.2.4 Make sure the Programmer shows the correct file and correct parts in the JTAG chain and check the Program/Configure checkbox.



File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
Factory default enhanc...	10CL025Y	0017EA71	0017EA71	☑	☐	☐	☐	☐	☐	☐
output_files/output...	EPCQ16A	1514B631		☑	☐	☐	☐	☐	☐	☐

6.2.2.5 Click **Start** to configure EPCQ-A. The programming could take a while.

6.2.2.6 When the programming is finished, the CYC000 should be able to keep its configuration data even after powered off.

At this point our program is stored in the EPCQ-A flash memory, but the Cyclone 10 LP current configuration is the Serial Flash Loader which is responsible for programming configuration flash memory. We can simply reconfigure the FPGA with our program by pushing RESET button which will reset the FPGA and automatically loads the configuration from EPCQ-A.

6.3 Testing the Design

Does not matter which way the CYC1000 was configured, the results should be the same for both methods, with the only difference being if configuration is retained after power off.

On the board by default, the LEDS should now toggle in a slow counting sequence.

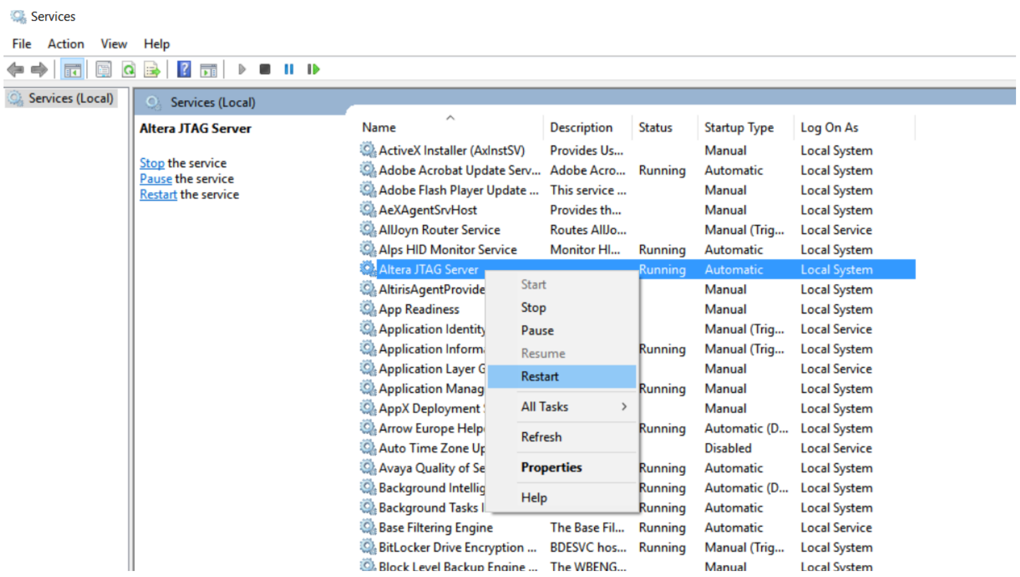
Push and hold the USER_BTN to see that the LEDs will now toggle in a very fast counting sequence. USER_BTN is on the side of the LEDs.

Releasing the USER_BTN, will make the LEDs toggle at a slower rate as before.

Chapter 7 - Common Issues and Fixes

1) **Issue:** In some rare cases when using Windows 10 operating system, the programmer DLL is not properly loaded/unloaded, causing the Quartus Programmer to not detect the Arrow USB Programmer2.

   Solution: Restart the Altera JTAG Server using the Services application of Windows.



Altera JTAG Server

Stop the service
Pause the service
Restart the service

Name	Description	Status	Startup Type	Log On As
ActiveX Installer (AxInstSV)	Provides Us...		Manual	Local System
Adobe Acrobat Update Serv...	Adobe Acro...	Running	Automatic	Local System
Adobe Flash Player Update ...	This service ...		Manual	Local System
AeXAgentSrvHost	Provides th...		Manual	Local System
AllJoyn Router Service	Routes AllJo...		Manual (Trig...	Local Service
Alps HID Monitor Service	Monitor HI...	Running	Automatic	Local System
Altera JTAG Server		Running	Automatic	Local System
AltirisAgentProvide			Manual	Local System
App Readiness			Manual	Local System
Application Identity			Manual (Trig...	Local Service
Application Inform		Running	Manual (Trig...	Local System
Application Layer G			Manual	Local Service
Application Manag		Running	Manual	Local System
AppX Deployment			Manual	Local System
Arrow Europe Help		Running	Automatic (D...	Local System
Auto Time Zone Up			Disabled	Local Service
Avaya Quality of Se		Running	Automatic	Local System
Background Intellig		Running	Automatic (D...	Local System
Background Tasks I		Running	Automatic	Local System
Base Filtering Engine	The Base Fil...	Running	Automatic	Local Service
BitLocker Drive Encryption ...	BDESVC hos...	Running	Manual (Trig...	Local System
Block Level Backup Engine ...	The WBENG...		Manual	Local System

Chapter 8 - Appendix

8.1 Revision History

Version	Change Log	Date of Change
V1.0	Initial Version	03/02/2020

8.2 Legal Disclaimer

ARROW ELECTRONICS

EVALUATION BOARD LICENSE AGREEMENT

By using this evaluation board or kit (together with all related software, firmware, components, and documentation provided by Arrow, “Evaluation Board”), You (“You”) are agreeing to be bound by the terms and conditions of this Evaluation Board License Agreement (“Agreement”). Do not use the Evaluation Board until You have read and agreed to this Agreement. Your use of the Evaluation Board constitutes Your acceptance of this Agreement.

PURPOSE
The purpose of this evaluation board is solely intended for evaluation purposes. Any use of the Board beyond these purposes is on your own risk. Furthermore, according the applicable law, the offering Arrow entity explicitly does not warrant, guarantee or provide any remedies to you with regard to the board.

LICENSE
Arrow grants You a non-exclusive, limited right to use the enclosed Evaluation Board offering limited features only for Your evaluation and testing purposes in a research and development setting. Usage in a live environment is prohibited. The Evaluation Board shall not be, in any case, directly or indirectly assembled as a part in any production of Yours as it is solely developed to serve evaluation purposes and has no direct function and is not a finished product.

EVALUATION BOARD STATUS
The Evaluation Board offers limited features allowing You only to evaluate and test purposes. The Evaluation Board is not intended for consumer or household use. You are not authorized to use the Evaluation Board in any production system, and it may not be offered for sale or lease, or sold, leased or otherwise distributed for commercial purposes.

OWNERSHIP AND COPYRIGHT
Title to the Evaluation Board remains with Arrow and/or its licensors. This Agreement does not involve any transfer of intellectual property rights (“IPR) for evaluation board. You may not remove any copyright or other proprietary rights notices without prior written authorization from Arrow or it licensors.

RESTRICTIONS AND WARNINGS
Before You handle or use the Evaluation Board, You shall comply with all such warnings and other instructions and employ reasonable safety precautions in using the Evaluation Board. Failure to do so may result in death, personal injury, or property damage.

You shall not use the Evaluation Board in any safety critical or functional safety testing, including but not limited to testing of life supporting, military or nuclear applications. Arrow expressly disclaims any responsibility for such usage which shall be made at Your sole risk.

WARRANTY
Arrow warrants that it has the right to provide the evaluation board to you. This warranty is provided by Arrow in lieu of all other warranties, written or oral, statutory, express or implied, including any warranty as to merchantability, non-infringement, fitness for any particular purpose, or uninterrupted or error-free operation, all of which are expressly disclaimed. The evaluation board is provided “as is” without any other rights or warranties, directly or indirectly.

You warrant to Arrow that the evaluation board is used only by electronics experts who understand the dangers of handling and using such items, you assume all responsibility and liability for any improper or unsafe handling or use of the evaluation board by you, your employees, affiliates, contractors, and designees.

LIMITATION OF LIABILITIES
In no event shall Arrow be liable to you, whether in contract, tort (including negligence), strict liability, or any other legal theory, for any direct, indirect, special, consequential, incidental, punitive, or exemplary damages with respect to any matters relating to this agreement. In no event shall arrow's liability arising out of this agreement in the aggregate exceed the amount paid by you under this agreement for the purchase of the evaluation board.

IDENTIFICATION
You shall, at Your expense, defend Arrow and its Affiliates and Licensors against a claim or action brought by a third party for infringement or misappropriation of any patent, copyright, trade secret or other intellectual property right of a third party to the extent resulting from (1) Your combination of the Evaluation Board with any other component, system, software, or firmware, (2) Your modification of the Evaluation Board, or (3) Your use of the Evaluation Board in a manner not permitted under this Agreement. You shall indemnify Arrow and its Affiliates and Licensors against and pay any resulting costs and damages finally awarded against Arrow and its Affiliates and Licensors or agreed to in any settlement, provided that You have sole control of the defense and settlement of the claim or action, and Arrow cooperates in the defense and furnishes all related evidence under its control at Your expense. Arrow will be entitled to participate in the defense of such claim or action and to employ counsel at its own expense.

RECYCLING
The Evaluation Board is not to be disposed as an urban waste. At the end of its life cycle, differentiated waste collection must be followed, as stated in the directive 2002/96/EC. In all the countries belonging to the European Union (EU Dir. 2002/96/EC) and those following differentiated recycling, the Evaluation Board is subject to differentiated recycling at the end of its life cycle, therefore: It is forbidden to dispose the Evaluation Board as an undifferentiated waste or with other domestic wastes. Consult the local authorities for more information on the proper disposal channels. An incorrect Evaluation Board disposal may cause damage to the environment and is punishable by the law.