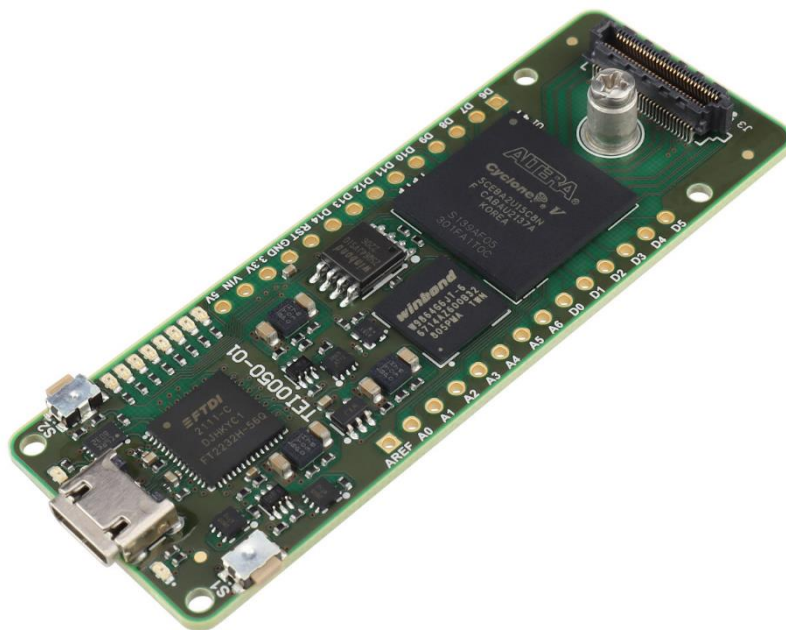




CYC5000

User Guide



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

Revision 1.0



Table of Contents

Table of Figures	4
Chapter 1 - CYC5000 IoT / Maker Board	5
1.1 About Arrow CYC5000 Board	5
1.2 Useful Links	5
1.3 Getting Help.....	6
Chapter 2 - Introduction to the CYC5000 Board.....	7
2.1 Layout and Components	7
2.2 Block Diagram.....	8
Chapter 3 - Connections and Peripherals of the CYC5000 Board	10
3.1 Board Status Elements.....	10
3.2 Clock Circuitry	10
3.3 Peripherals Connected to the FPGA	11
3.3.1 Communication and Configuration.....	11
3.3.2 QSPI Configuration Flash Memory.....	13
3.3.3 SDRAM Memory.....	14
3.3.4 CRUVI HS Connector	16
3.3.5 Arduino Header	18
3.3.6 LEDs	19
3.3.7 Push Buttons	20
3.3.8 Power Tree.....	21
Chapter 4 - Software and Driver Installation.....	23
4.1 Installing Quartus Prime Software	23
4.2 Installing Arrow USB Programmer2.....	25
4.3 License	27
Chapter 5 - New Project with CYC5000	29
5.1 Creating a new Blinky Project with CYC5000	29
5.2 Building a Blinky Project with CYC5000.....	33
5.2.1 Block Diagram.....	33
5.2.2 Components of the Design	34
5.2.3 Catalog IP	34
5.2.4 Create and Configure PLL	34
5.2.5 Create and Configure the Counter	36
5.2.6 Create and Configure the Multiplexer.....	39
5.2.7 Adding the Components to the Schematic.....	41
5.2.8 Connecting the Components	43



5.2.9	Add inputs, outputs to the schematic	46
5.2.10	Analysis and Synthesis	49
5.2.11	Adding Timing Constraints.....	49
5.2.12	Pinning Assignments	51
5.2.13	Compiling the Design	54
5.2.14	Reading the Compilation Report.....	55
Chapter 6 - Configuring the CYC5000.....		57
6.1	Configure the FPGA in JTAG mode	57
6.2	QSPI flash memory programming.....	61
6.2.1	Programming File generation	61
6.2.2	Device Programming	64
6.3	Testing the Design	66
Chapter 7 - Common Issues and Fixes.....		67
Chapter 8 - Appendix		68
8.1	Revision History	68
8.2	Legal Disclaimer.....	69



Table of Figures

Figure 1 – CYC5000 Board (top view)	7
Figure 2 – CYC5000 Block Diagram	8
Figure 3 – Position of Indication LEDs	10
Figure 4 – CYC5000 Clock Tree	11
Figure 5 – FTDI Connections	12
Figure 6 – JTAG Connections.....	13
Figure 7 – Configuration Flash Connections	14
Figure 8 – SDRAM Connections.....	14
Figure 9 – CRUVI HS Connections	16
Figure 10 - Arduino Header Connections.....	18
Figure 11 – LED Connections.....	20
Figure 12 – Button Connections.....	21
Figure 13 – Power Tree Connections	22



Chapter 1 - CYC5000 IoT / Maker Board

1.1 About Arrow CYC5000 Board

The CYC5000 is a customizable IoT / Maker Board ready for evaluation, development, and/or use in a product. The board is based on Cyclone V FPGA, which is optimized to simultaneously accommodate the shrinking power consumption, cost, and time-to-market requirements, furthermore, increasing bandwidth requirements for a wide spectrum of general logic and DSP applications. The ready-to-use hard intellectual property (IP) blocks in the core FPGA fabric, such as variable precision digital signal processing (DSP) blocks, and multiport memory controllers consume less power and free up more logic resources, which provides sufficient resources to implement other functions such as Nios II 32-bit microcontroller IP or various interface controls.

The CYC5000 is equipped with an Arrow USB Programmer2, SDRAM, flash memory, CRUVI HS and ARDUINO MKR connectors making it a fully featured plug-and-play solution without any additional costs.

The CYC5000 board contains all the tools needed to use the board in conjunction with a computer that runs a 64-bit Linux / Microsoft Windows 10 operating system or later.

1.2 Useful Links

A set of useful links that can be used to get relevant information about the CYC5000 board or the Cyclone V FPGA.

- [CYC5000 at Arrow Shop](#)
- [CYC5000 at Trenz Electronic Shop](#)
- [Intel Cyclone V Webpage](#)
- [CYC5000 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 CYC5000 Board

2.1 Layout and Components

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

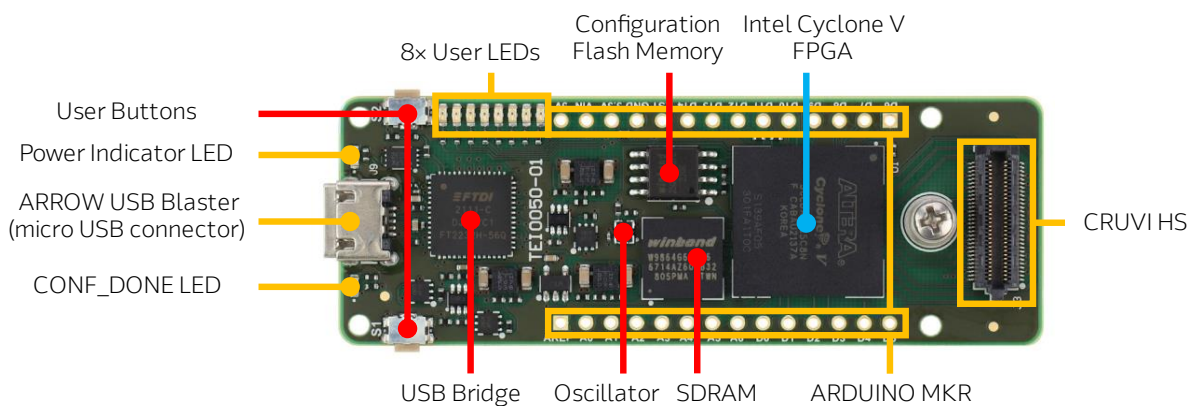


Figure 1 – CYC5000 Board (top view)

The following features are available on the CYC5000 board:

- Intel Cyclone V 5CEBA2U15C8N device
- Arrow USB Programmer2 on-board for programming; JTAG Mode
- 64Mbit SDRAM up to 166MHz
- 64Mbit QSPI Configuration flash memory
- CRUVI HS Connector
- USB-to-JTAG
- GPIO-FTDI
- Arduino MKR Header
- 12MHz MEMS Oscillator
- 8x red user LEDs
- 2x board indication LEDs
- 2x user push buttons

2.2 Block Diagram

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

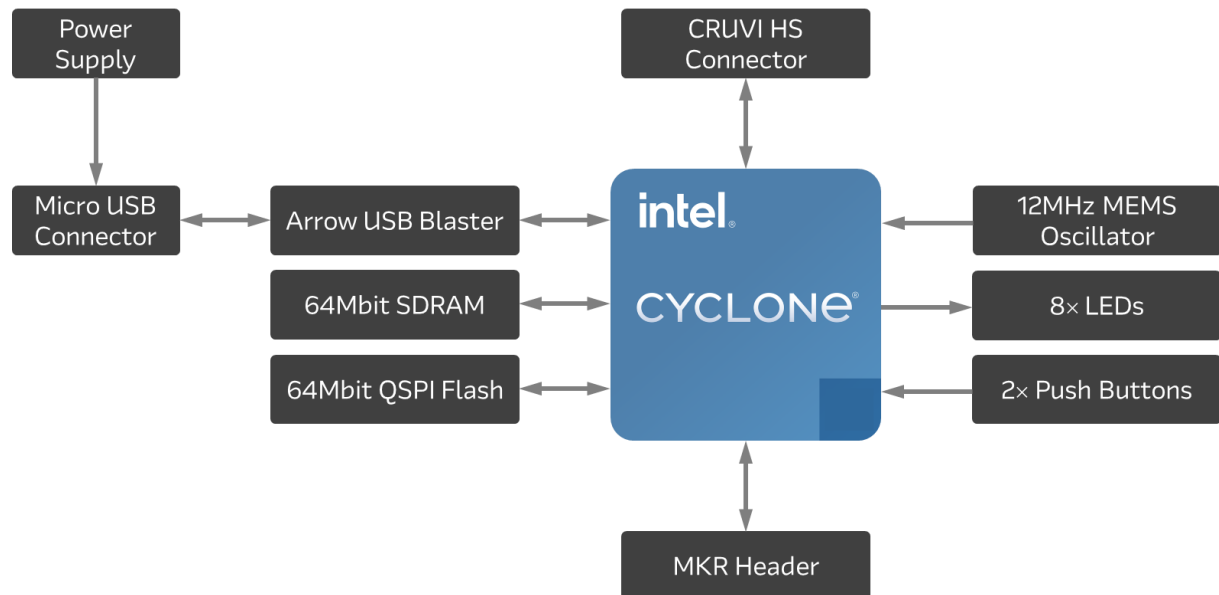


Figure 2 – CYC5000 Block Diagram

FPGA Device

- Intel Cyclone V 5CEBA2U15C8N device.
- Features of the FPGA on the CYC5000 Board:

Resources	Device
	5CEBA2
Logic Elements (kLE)	25
Adaptive Logic Module (ALM)	9,430
M10K Memory (Kb)	1,760
Variable-precision DSP Block	25
18 × 18 Multiplier	50
PLLs	4
I/O	176

Memory Devices

- 64Mbit external SDRAM memory
- 64Mbit external QSPI Flash memory



Configuration and Debug

- On-board Arrow USB Programmer2 (micro-USB type B connector) – JTAG mode

Connectors & Headers

- CRUVI HS Connector
- Arduino MKR compatible Header

Buttons and Indicators

- 2× Side-Buttons
- 8× red user LEDs
- 2× board status LEDs

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 CYC5000 Board

3.1 Board Status Elements

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



Figure 3 – Position of Indication LEDs

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

3.2 Clock Circuitry

The external clock 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 clock input from CRUVI HS Connector, where you can add another preferred clock source to the FPGA (REFCLK). Both clock signals drive the internal PLLs of the FPGA.

For more information on clocks and PLLs of the Cyclone V, please refer to this [document](#).

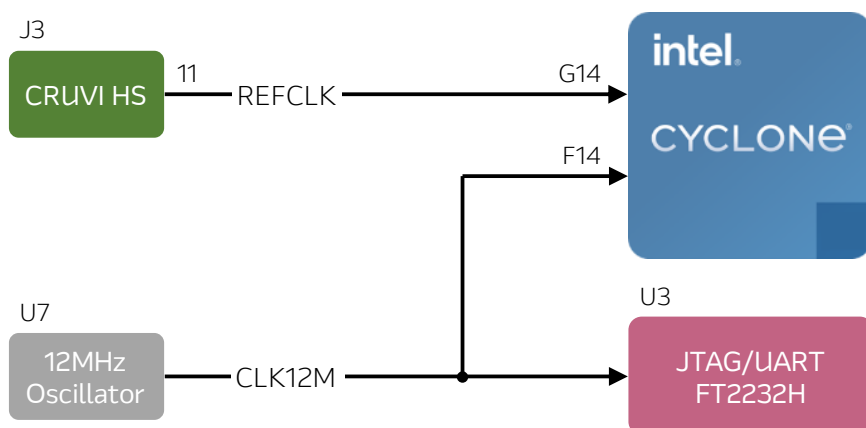


Figure 4 – CYC5000 Clock Tree

Board Reference	FPGA Pin No.	Pin Func.	Description	I/O Std
CLK12M	PIN_F14	Input	12MHz clock input	3.3 V
REFCLK	PIN_G14	Input	Optional clock input	3.3 V

3.3 Peripherals Connected to the FPGA

3.3.1 Communication and Configuration

The CYC5000 board uses a single chip to perform configuration of the device and communication over USB.

3.3.1.1 USB Communication

The FTDI chip converts signals from USB 2.0 to a variety of standard serial and parallel interfaces. Channel A of FTDI chip is used in MPPSE mode for JTAG. Channel B is routed to FPGA and is usable for other standard interfaces.

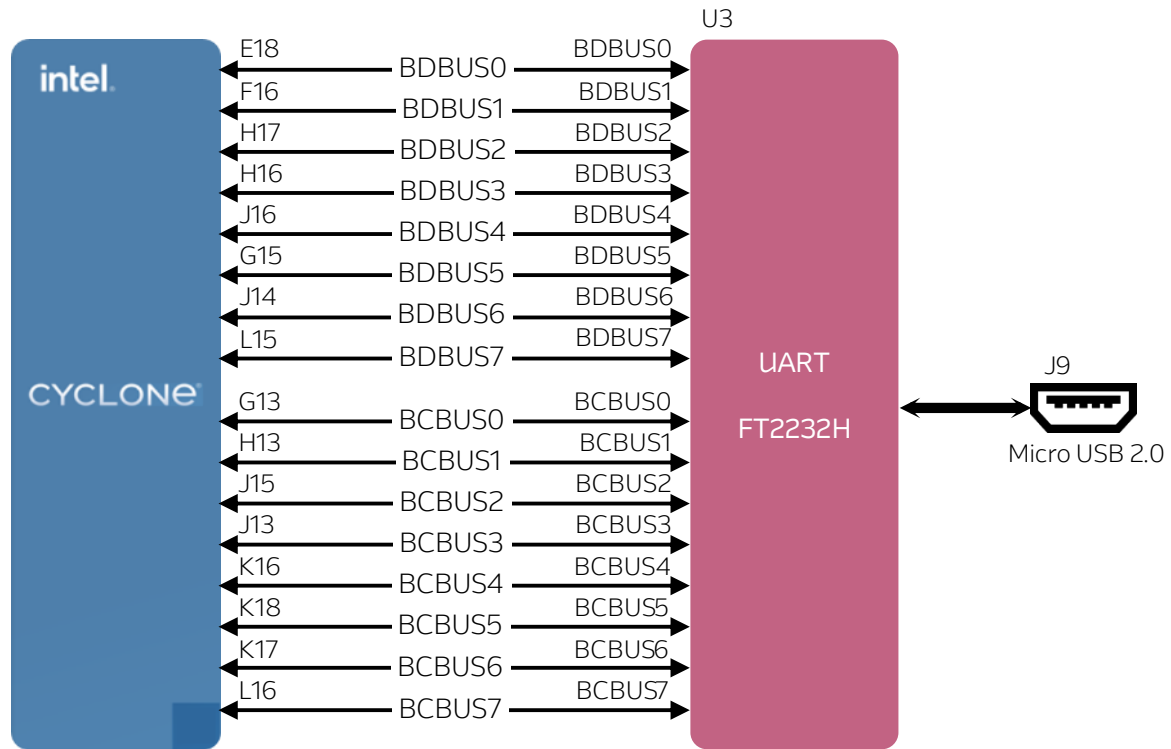


Figure 5 – FTDI Connections

Board Reference	FPGA Pin No.	Pin Func.	Description	I/O Std
BDBUS0	PIN_E18	Bidir	D[0] of bidirectional data bus	3.3 V
BDBUS1	PIN_F16	Bidir	D[1] of bidirectional data bus	3.3 V
BDBUS2	PIN_H17	Bidir	D[2] of bidirectional data bus	3.3 V
BDBUS3	PIN_H16	Bidir	D[3] of bidirectional data bus	3.3 V
BDBUS4	PIN_J16	Bidir	D[4] of bidirectional data bus	3.3 V
BDBUS5	PIN_G15	Bidir	D[5] of bidirectional data bus	3.3 V
BDBUS6	PIN_J14	Bidir	D[6] of bidirectional data bus	3.3 V
BDBUS7	PIN_L15	Bidir	D[7] of bidirectional data bus	3.3 V
BCBUS0	PIN_G13	Bidir	D[0] of bidirectional data bus	3.3 V
BCBUS1	PIN_H13	Bidir	D[1] of bidirectional data bus	3.3 V
BCBUS2	PIN_J15	Bidir	D[2] of bidirectional data bus	3.3 V
BCBUS3	PIN_J13	Bidir	D[3] of bidirectional data bus	3.3 V
BCBUS4	PIN_K16	Bidir	D[4] of bidirectional data bus	3.3 V
BCBUS5	PIN_K18	Bidir	D[5] of bidirectional data bus	3.3 V
BCBUS6	PIN_K17	Bidir	D[6] of bidirectional data bus	3.3 V
BCBUS7	PIN_L16	Bidir	D[7] of bidirectional data bus	3.3 V

3.3.1.2 JTAG Chain Configuration

There are two types of configuration methods supported by CYC5000:

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 V with a download cable through the Quartus Prime Programmer. This function is only available via the On-board Arrow USB Programmer2.
2. **Configuration from QSPI flash:** configuration using external flash. Before configuration, you need to program the configuration data .jic into the configuration flash memory which provides non-volatile storage for the bit stream. The information is retained within flash memory even if the CYC5000 is turned off. When the board is powered on, the configuration data in the flash memory is automatically loaded into the Cyclone V FPGA.

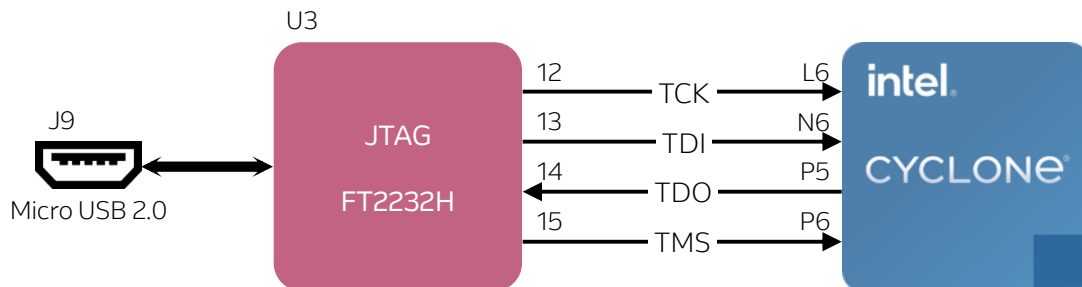


Figure 6 – JTAG Connections

Board Reference	FPGA Pin No.	Pin Func.	Description	I/O Std
TCK	PIN_L6	Input	Test Interface Clock	3.3 V
TDI	PIN_N6	Input	Test Data In	3.3 V
TDO	PIN_P5	Output	Test Data Out	3.3 V
TMS	PIN_P6	Input	Test Mode Select	3.3 V

For detailed information about how to configure the Cyclone V, please refer to [Chapter 6](#).

3.3.2 QSPI Configuration Flash Memory

The CYC5000 board is integrated with a 64MBit of QSPI flash memory that can be used for user data and programming non-volatile storage. The configuration bitstream is downloaded into the configuration device which automatically loads the configuration data into the Cyclone V when the board is powered on. Device memory capacity not consumed storing configuration data can be used as general-purpose non-volatile memory, which with its operation of up to 133MHz is perfect for program and data storage. Several interfaces available with Nios II embedded

processors allow you to access the serial configuration device as a memory module connected to your embedded system.

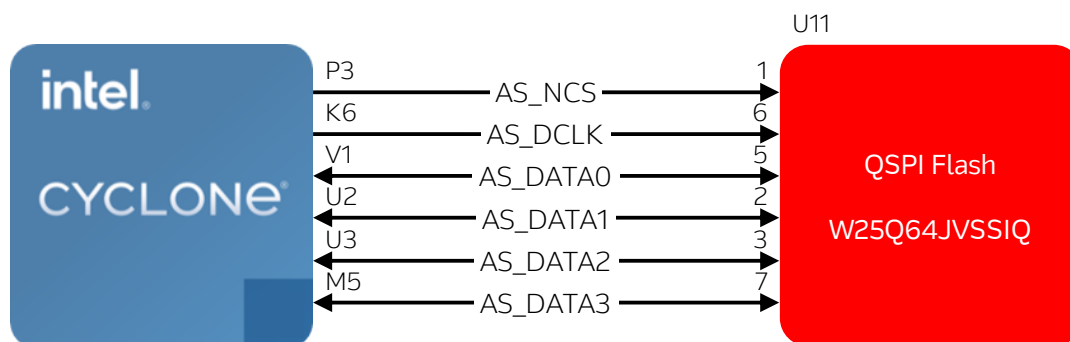


Figure 7 – Configuration Flash Connections

Board Reference	FPGA Pin No.	Pin Func.	Description	I/O Std
AS_NCS	PIN_P3	Output	Chip Select	3.3 V
AS_DCLK	PIN_K6	Output	Clock	3.3 V
AS_DATA0	PIN_V1	Bidir	Data [0]	3.3 V
AS_DATA1	PIN_U2	Bidir	Data [1]	3.3 V
AS_DATA2	PIN_U3	Bidir	Data [2]	3.3 V
AS_DATA3	PIN_M5	Bidir	Data [3]	3.3 V

3.3.3 SDRAM Memory

The CYC5000 board supports single-chip SDRAM with 64Mbit density which can operate up to 166 MHz clock frequency. Below are the connections and pinning of the SDRAM used in the CYC5000.

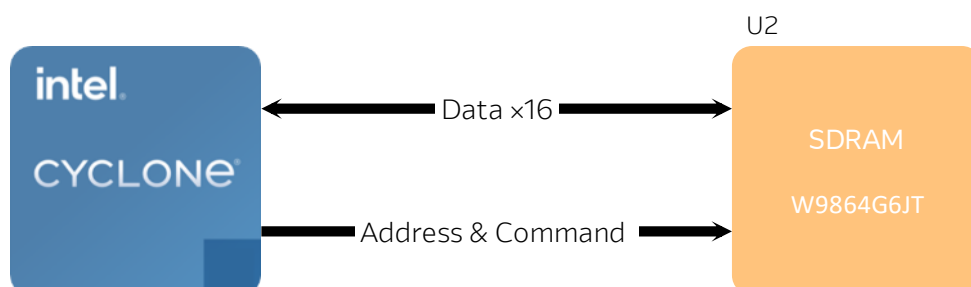


Figure 8 – SDRAM Connections

Board Reference	FPGA Pin No.	Pin Func.	Description	I/O Std
A0	PIN_R13	Output	SDRAM Address [0]	3.3 V
A1	PIN_U12	Output	SDRAM Address [1]	3.3 V
A2	PIN_V12	Output	SDRAM Address [2]	3.3 V
A3	PIN_V13	Output	SDRAM Address [3]	3.3 V
A4	PIN_V15	Output	SDRAM Address [4]	3.3 V
A5	PIN_V16	Output	SDRAM Address [5]	3.3 V
A6	PIN_T16	Output	SDRAM Address [6]	3.3 V
A7	PIN_U15	Output	SDRAM Address [7]	3.3 V
A8	PIN_P14	Output	SDRAM Address [8]	3.3 V
A9	PIN_T15	Output	SDRAM Address [9]	3.3 V
A10	PIN_M13	Output	SDRAM Address [10]	3.3 V
A11	PIN_P15	Output	SDRAM Address [11]	3.3 V
A12	PIN_N16	Output	SDRAM Address [12]	3.3 V
A13	PIN_R16	Output	SDRAM Address [13]	3.3 V
BA0	PIN_T12	Output	SDRAM Bank Address [0]	3.3 V
BA1	PIN_N13	Output	SDRAM Bank Address [1]	3.3 V
RAS	PIN_P13	Output	SDRAM Row Address Strobe	3.3 V
CAS	PIN_M14	Output	SDRAM Column Address Strobe	3.3 V
WE	PIN_N12	Output	SDRAM Write Enable	3.3 V
CS	PIN_L13	Output	SDRAM Chip Select	3.3 V
CLK	PIN_P16	Output	SDRAM Input Clock	3.3 V
CKE	PIN_T14	Output	SDRAM Clock Enable	3.3 V
DQ0	PIN_U4	Bidir	SDRAM Data [0]	3.3 V
DQ1	PIN_T4	Bidir	SDRAM Data [1]	3.3 V
DQ2	PIN_V6	Bidir	SDRAM Data [2]	3.3 V
DQ3	PIN_U5	Bidir	SDRAM Data [3]	3.3 V
DQ4	PIN_V7	Bidir	SDRAM Data [4]	3.3 V
DQ5	PIN_T5	Bidir	SDRAM Data [5]	3.3 V
DQ6	PIN_V8	Bidir	SDRAM Data [6]	3.3 V
DQ7	PIN_U8	Bidir	SDRAM Data [7]	3.3 V
DQ8	PIN_P10	Bidir	SDRAM Data [8]	3.3 V
DQ9	PIN_P9	Bidir	SDRAM Data [9]	3.3 V
DQ10	PIN_T11	Bidir	SDRAM Data [10]	3.3 V
DQ11	PIN_R9	Bidir	SDRAM Data [11]	3.3 V
DQ12	PIN_R11	Bidir	SDRAM Data [12]	3.3 V
DQ13	PIN_T9	Bidir	SDRAM Data [13]	3.3 V
DQ14	PIN_V10	Bidir	SDRAM Data [14]	3.3 V
DQ15	PIN_U9	Bidir	SDRAM Data [15]	3.3 V
DQM0	PIN_U13	Output	SDRAM Lower Data Mask	3.3 V
DQM1	PIN_U14	Output	SDRAM Upper Data Mask	3.3 V

3.3.4 CRUVI HS Connector

The CYC5000 board has one CRUVI HS connector. CRUVI HS is an open ecosystem which makes it possible to add a big variety of interfaces to the system, that can require high-speed signalling as well as low-speed device interface support.

The voltage level of High-Speed interfaces can be selected between 1.8V and 3.3V which ensures that these I/Os are compatible with the high-speed LVDS standard as well as the lower-speed GPIOs.

The CYC5000 board provides +5V and +3.3V power to the mezzanine card through the CRUVI HS connector.

For custom add-on cards with CRUVI HS interface, the recommended counterpart is **ST4-30-1.50-L-D** from Samtec.

Below is the connection diagram and pinning information.

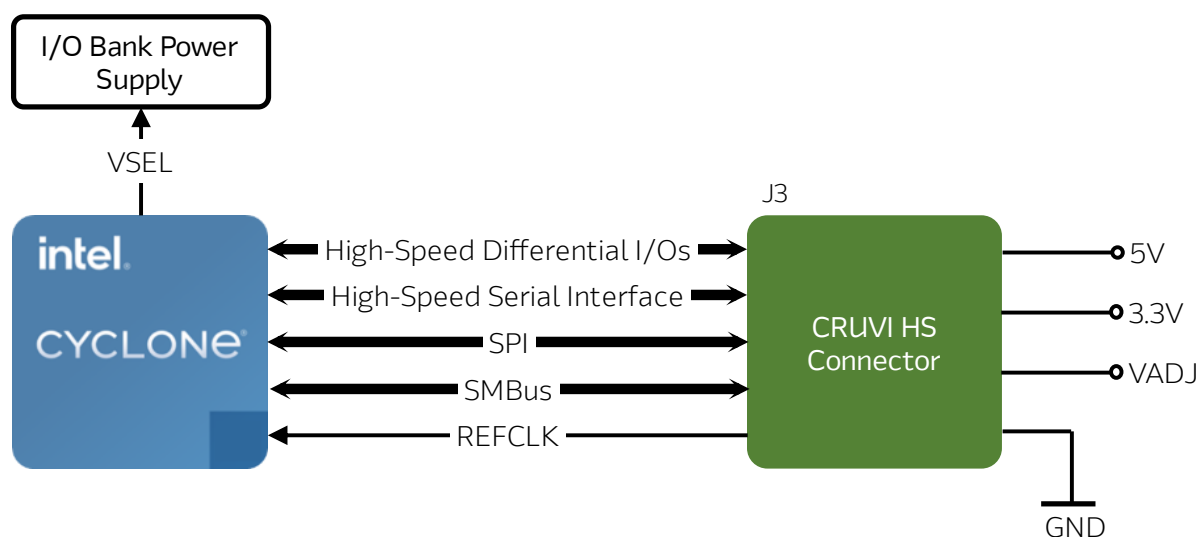


Figure 9 – CRUVI HS Connections

Board Reference	FPGA Pin No.	CRUVI HS No.	Pin Func.	Description	I/O Std
A0_P	PIN_A12	14	Output	High-Speed Differential Transmitter A[0]_p	1.8 V / 3.3 V
A0_N	PIN_B12	16	Output	High-Speed Differential Transmitter A[0]_n	1.8 V / 3.3 V
A1_P	PIN_A11	20	Output	High-Speed Differential Transmitter A[1]_p	1.8 V / 3.3 V
A1_N	PIN_A10	22	Output	High-Speed Differential Transmitter A[1]_n	1.8 V / 3.3 V
A2_P	PIN_A14	26	Output	High-Speed Differential Transmitter A[2]_p	1.8 V / 3.3 V

Board Reference	FPGA Pin No.	CRUVI HS No	Pin Func.	Description	I/O Std
A2_N	PIN_A15	28	Output	High-Speed Differential Transmitter A[2]_n	1.8 V / 3.3 V
A3_P	PIN_A16	32	Output	High-Speed Differential Transmitter A[3]_p	1.8 V / 3.3 V
A3_N	PIN_A17	34	Output	High-Speed Differential Transmitter A[3]_n	1.8 V / 3.3 V
A4_P	PIN_F9	38	Input	High-Speed Differential Receiver A[4]_p	1.8 V / 3.3 V
A4_N	PIN_F10	40	Input	High-Speed Differential Receiver A[4]_n	1.8 V / 3.3 V
A5_P	PIN_B7	44	Input	High-Speed Differential Receiver A[5]_p	1.8 V / 3.3 V
A5_N	PIN_B8	46	Input	High-Speed Differential Receiver A[5]_n	1.8 V / 3.3 V
B0_P	PIN_B17	15	Output	High-Speed Differential Transmitter B[0]_p	1.8 V / 3.3 V
B0_N	PIN_B18	17	Output	High-Speed Differential Transmitter B[0]_n	1.8 V / 3.3 V
B1_P	PIN_A7	21	Output	High-Speed Differential Transmitter B[1]_p	1.8 V / 3.3 V
B1_N	PIN_A6	23	Output	High-Speed Differential Transmitter B[1]_n	1.8 V / 3.3 V
B2_P	PIN_C13	37	Input	High-Speed Differential Receiver B[2]_p	1.8 V / 3.3 V
B2_N	PIN_C12	29	Input	High-Speed Differential Receiver B[2]_n	1.8 V / 3.3 V
B3_P	PIN_E8	33	Input	High-Speed Differential Receiver B[3]_p	1.8 V / 3.3 V
B3_N	PIN_F7	35	Input	High-Speed Differential Receiver B[3]_n	1.8 V / 3.3 V
B4_P	PIN_G6	39	Input	High-Speed Differential Receiver B[4]_p	1.8 V / 3.3 V
B4_N	PIN_F6	41	Input	High-Speed Differential Receiver B[4]_n	1.8 V / 3.3 V
B5_P	PIN_B4	45	Input	High-Speed Differential Receiver B[5]_p	1.8 V / 3.3 V
B5_N	PIN_B5	47	Input	High-Speed Differential Receiver B[5]_n	1.8 V / 3.3 V
HSI	PIN_D16	10	Bidir	Serial In or High-Speed Differential Transmitter A[6]_p	1.8 V / 3.3 V
HSMIO	PIN_C16	2	Bidir	Serial Data I/O or High-Speed Differential Transmitter A[6]_n	1.8 V / 3.3 V
HSO	PIN_B14	6	Output	Serial Out or High-Speed Differential Transmitter A[7]_p	1.8 V / 3.3 V
HSRST	PIN_B15	8	Output	Serial Reset or High-Speed Differential Transmitter A[7]_n	1.8 V / 3.3 V
SMB_ALERT	PIN_G17	3	Input	SMBus interrupt signal	3.3 V
SMB_SDA	PIN_H18	5	Bidir	SMBus Data Line	3.3 V
SMB_SCL	PIN_F17	7	Output	SMBus Data Clock Line	3.3 V

Board Reference	FPGA Pin No.	CRUVI HS No	Pin Func.	Description	I/O Std
SDI	PIN_F4	51	Input	Master Input Slave Output	3.3 V
SDO	PIN_J3	53	Output	Master Output Slave Input	3.3 V
SEL	PIN_E3	55	Output	Chip Select	3.3 V
MODE	PIN_J4	57	Bidir	Mode pin, User IO	3.3 V
SCK	PIN_E2	59	Output	Clock	3.3 V
REFCLK	PIN_G14	11	Input	Clock Input	3.3 V
5V	-	60	PWR	5V power to the connector	-
3.3V	-	4, 9	PWR	3.3V power to the connector	-
ADJ	-	36	PWR	HS IO Bank voltage	-
GND	-	12, 13, 18, 19, 24, 25, 30, 31, 36, 37, 42, 43, 48, 49, 54	PWR	Ground to the connector	-
n.c.	-	1, 50, 52, 56, 58	-	Not connected	-
VSEL	PIN_N10	-	Output	HS IO Bank voltage select: - VSEL = '0'; ADJ = 1.8V - VSEL = '1'; ADJ = 3.3V	3.3 V

3.3.5 Arduino Header

The CYC5000 board offers connectivity to classic Arduino MKR compatible shields that could also alternatively be used as GPIOs. The MKR connectors offer up to 23 digital I/Os.

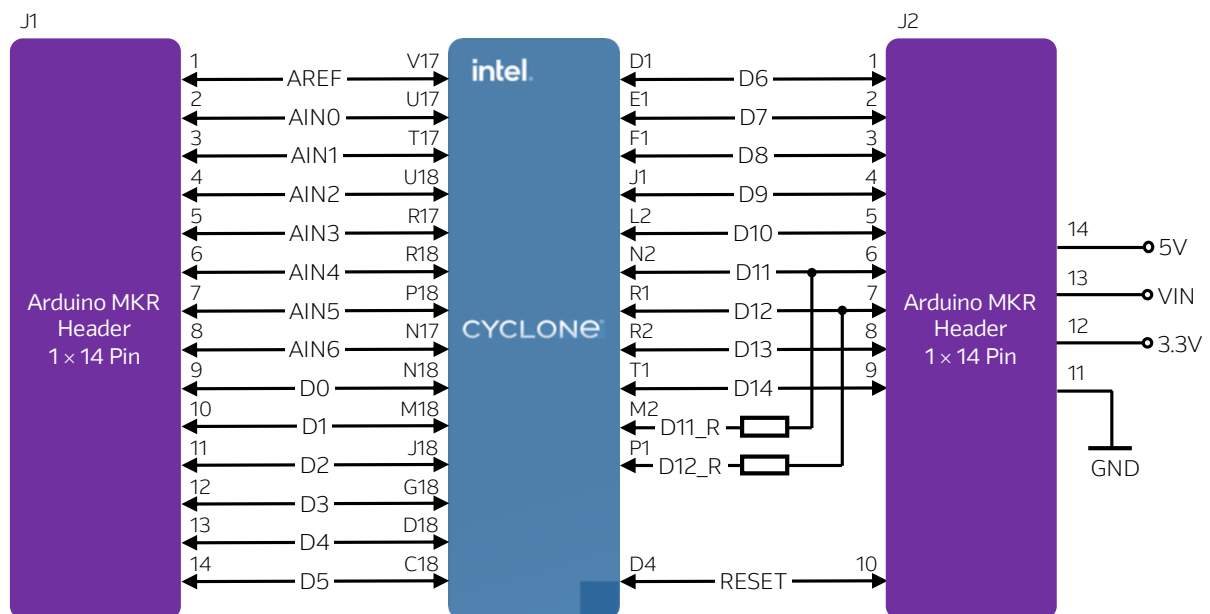


Figure 10 - Arduino Header Connections

Board Reference	FPGA Pin No.	Arduino Header	Pin Func.	Description	I/O Std
AREF	PIN_V17	J1 / 1	Bidir	Input reference voltage or GPIO	3.3 V
AIN0	PIN_U17	J1 / 2	Bidir	GPIO [0]	3.3 V
AIN1	PIN_T17	J1 / 3	Bidir	GPIO [1]	3.3 V
AIN2	PIN_U18	J1 / 4	Bidir	GPIO [2]	3.3 V
AIN3	PIN_R17	J1 / 5	Bidir	GPIO [3]	3.3 V
AIN4	PIN_R18	J1 / 6	Bidir	GPIO [4]	3.3 V
AIN5	PIN_P18	J1 / 7	Bidir	GPIO [5]	3.3 V
AIN6	PIN_N17	J1 / 8	Bidir	GPIO [6]	3.3 V
D0	PIN_N18	J1 / 9	Bidir	Digital I/O [0]	3.3 V
D1	PIN_M18	J1 / 10	Bidir	Digital I/O [1]	3.3 V
D2	PIN_J18	J1 / 11	Bidir	Digital I/O [2]	3.3 V
D3	PIN_G18	J1 / 12	Bidir	Digital I/O [3]	3.3 V
D4	PIN_D18	J1 / 13	Bidir	Digital I/O [4]	3.3 V
D5	PIN_C18	J1 / 14	Bidir	Digital I/O [5]	3.3 V
D6	PIN_D1	J2 / 1	Bidir	Digital I/O [6]	3.3 V
D7	PIN_E1	J2 / 2	Bidir	Digital I/O [7]	3.3 V
D8	PIN_F1	J2 / 3	Bidir	Digital I/O [8]	3.3 V
D9	PIN_J1	J2 / 4	Bidir	Digital I/O [9]	3.3 V
D10	PIN_L2	J2 / 5	Bidir	Digital I/O [10]	3.3 V
D11	PIN_N2	J2 / 6	Bidir	Digital I/O [11]*	3.3V
D12	PIN_R1	J2 / 7	Bidir	Digital I/O [12]*	3.3 V
D13	PIN_R2	J2 / 8	Bidir	Digital I/O [13]	3.3 V
D14	PIN_T1	J2 / 9	Bidir	Digital I/O [14]	3.3 V
D11_R	PIN_M2	J2 / 6	Bidir	Digital I/O [11] with resistor*	3.3 V
D12_R	PIN_P1	J2 / 7	Bidir	Digital I/O [12] with resistor*	3.3 V
Reset	PIN_H4	J2 / 10	Bidir	System reset of the board, controlled by the S1 user button	3.3 V
GND	-	J2 / 11	PWR	Ground of the connector	-
3.3V	-	J2 / 12	PWR	3.3V power to the connector	-
VIN	-	J2 / 13	PWR	User power into to the CYC5000 Board	-
5V	-	J2 / 14	PWR	5V power to the connector	-

*Can only choose one, hence same name pinning

3.3.6 LEDs

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

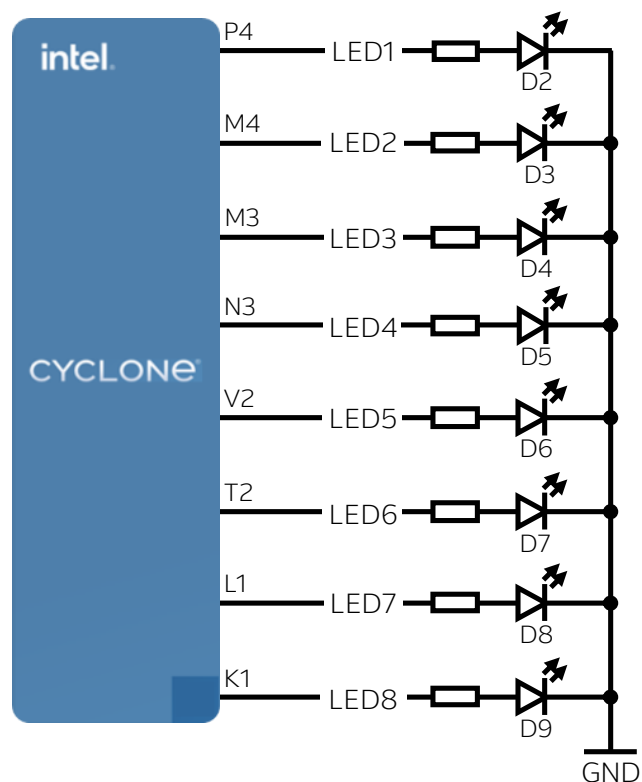


Figure 11 – LED Connections

Board Reference	FPGA Pin No.	Pin Func.	I/O Std
LED1	PIN_P4	Output	3.3 V
LED2	PIN_M4	Output	3.3 V
LED3	PIN_M3	Output	3.3 V
LED4	PIN_N3	Output	3.3 V
LED5	PIN_V2	Output	3.3 V
LED6	PIN_T2	Output	3.3 V
LED7	PIN_L1	Output	3.3 V
LED8	PIN_K1	Output	3.3 V

3.3.7 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.

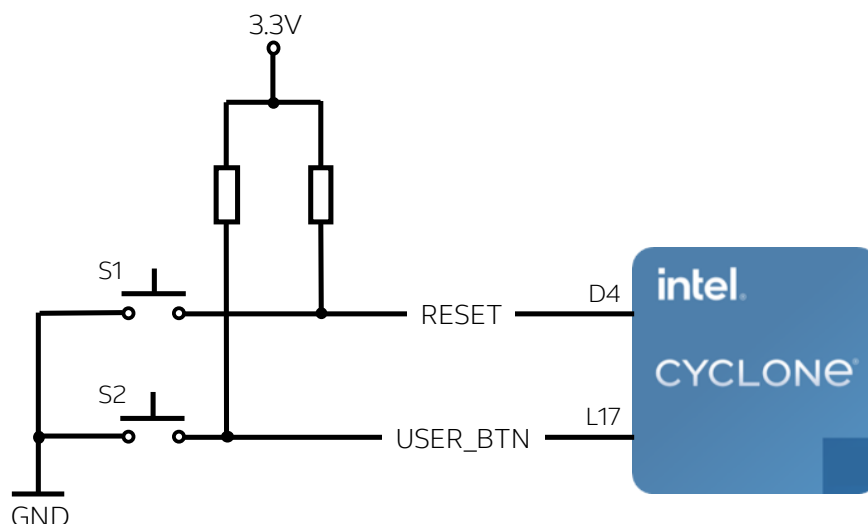


Figure 12 – Button Connections

Board Reference	FPGA Pin No.	Pin Func.	Description	I/O Std
RESET	PIN_D4	Input	nCONFIG	3.3 V
USER_BTN	PIN_L17	Input	User button	3.3 V

3.3.8 Power Tree

The CYC5000 is powered by MPM Power Modules which provide high efficiency on a small layout. 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 a 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 V FPGA is powered by 3 power modules, while a Microchip LDO provides auxiliary voltage.

Because of the 'Power Good' and 'Power Enable' signals of modules, there is no need for a separate power supply controller to manage the power sequencing.

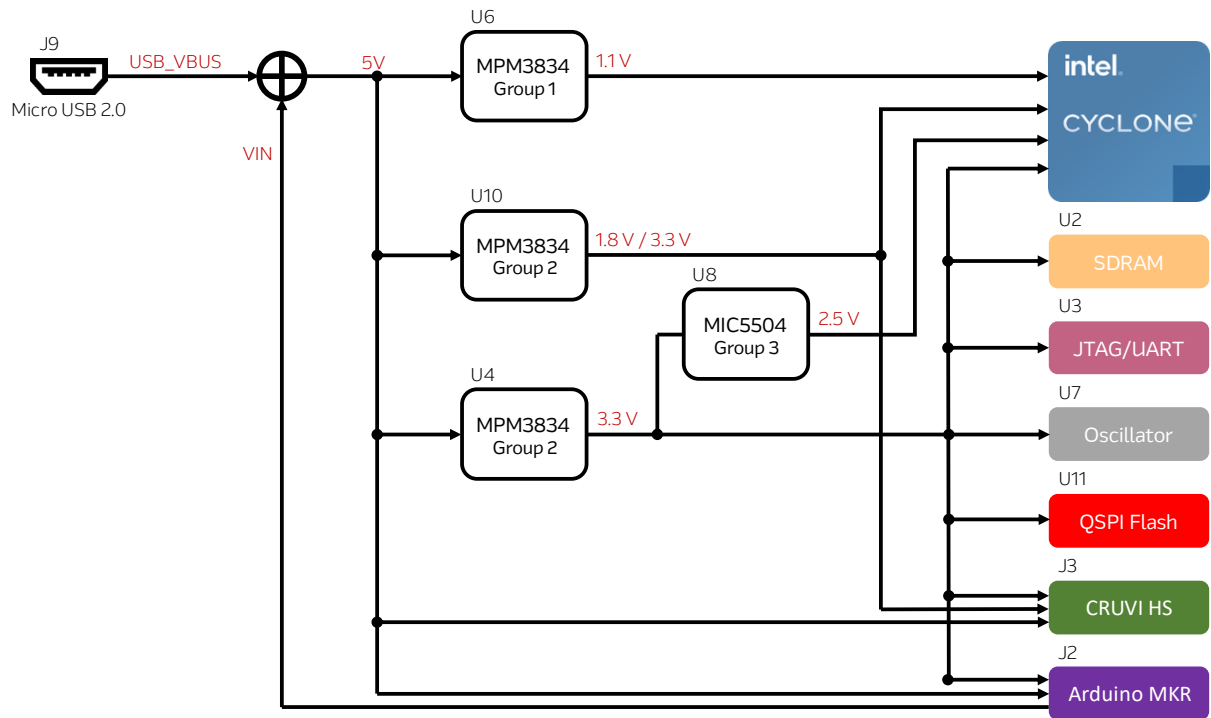


Figure 13 – Power Tree Connections

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 Make sure that **22.1.1** is selected, or your preferred version (highlighted in red).



4.1.3 Download the following files from the “Individual Files” tab (highlighted in yellow) and save them in the same folder:

- Intel Quartus Prime (included Nios II EDS)
- Questa – Intel FPGA Edition
- Intel Cyclone V device support

If the download page redirects you to the Software License Agreement page, accept the Legal Disclaimer, and the downloading will start automatically.

Intel® Quartus® Software

Intel® Quartus® Prime (includes Nios II EDS)

Download

QuartusLiteSetup-22.1std.1.917-windows.exe

Size: 1.6 GB

SHA1: 3694a64cc8253450ad7682a552396f7e292dad09

** Nios® II EDS on Windows requires Ubuntu 18.04 LTS on Windows Subsystem for Linux (WSL), which requires a manual installation.

** Nios® II EDS requires you to install an Eclipse IDE manually.

** Installation size: 8.86 GB

Questa® - Intel® FPGA Edition

Download

QuestaSetup-22.1std.1.917-windows.exe

Size: 780.1 MB

SHA1: 61219c4ba8cd88d51a87dfe5623ff6bacf346185

** Installation size: 2.73 GB

Devices

Intel® Arria® II device support

Download

arria_lite-22.1std.1.917.qdz

Size: 499.1 MB

SHA1: e9d3ce3a3a8581576f1a33c63a306c922fdd617d

** Installation size: 0.52 GB

Intel® Cyclone® IV device support

Download

cyclone-22.1std.1.917.qdz

Size: 465.8 MB

SHA1: cbbfc3ffdcce8a2535b9e129bd7444f3fa18b71f

** Installation size: 0.50 GB

Intel® Cyclone® 10 LP device support

Download

cyclone10lp-22.1std.1.917.qdz

Size: 265.5 MB

SHA1: a26747672b0e8f48c0e6691760760b3ce60cba42

** Installation size: 0.29 GB

Intel® Cyclone® V device support

Download

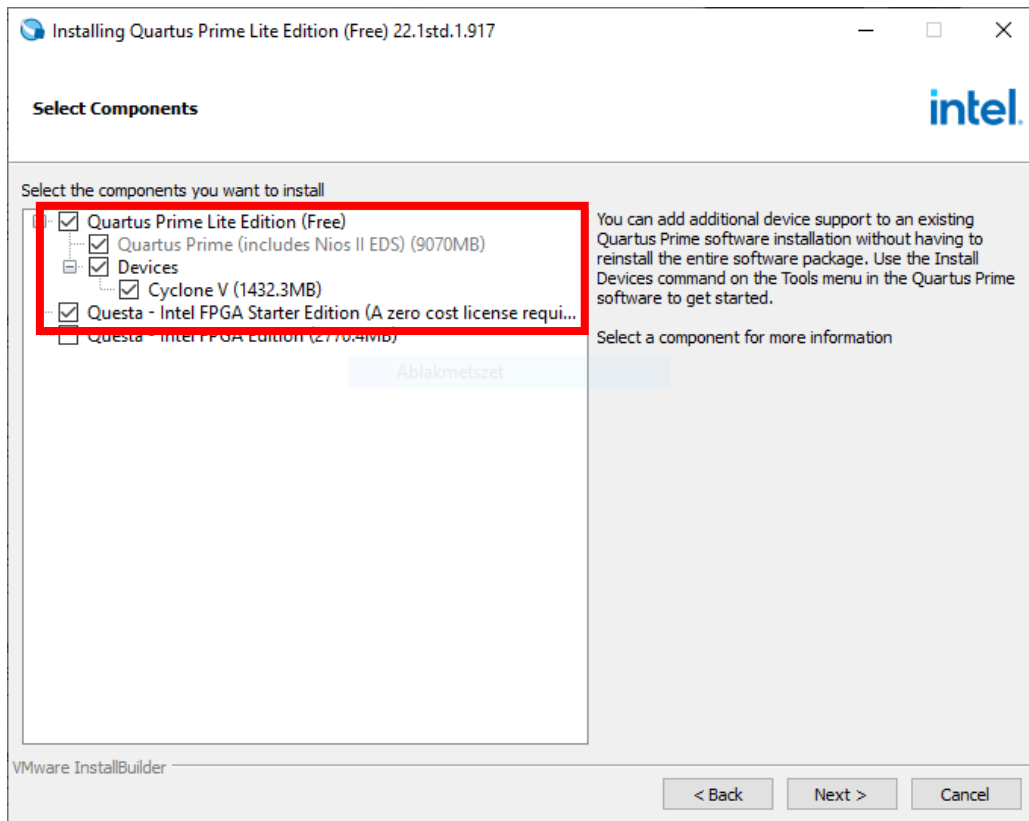
cyclonev-22.1std.1.917.qdz

Size: 1.3 GB

SHA1: 379e51b9e908cd43b9515f93f42f2a230a405a60

** Installation size: 1.40 GB

- 4.1.4 After the download is finished, run the Quartus Prime installer.
- 4.1.5 When prompted to select the components, the installer will automatically detect the Cyclone V device support and Questa packages when they are in the same folder. Make sure these components are selected:



- 4.1.6 Finish the installation of the Quartus Lite and proceed to the next section to install Arrow USB Programmer2 to be able to connect to the CYC5000 board.

4.2 Installing Arrow USB Programmer2

The CYC5000 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 CYC5000 work.

- 4.2.1 Download the appropriate version¹ of Arrow USB Programmer2 for CYC5000 from Trenz Electronic Wiki page or alternatively this direct [link](#).

¹Modules produced after June 2020 are no longer compatible with older drivers. Please install driver version 2.4 or newer.

Diligent

OHO-Elektronik

SunDance

Trenz_Electronic

-corporate

-obsolete_products

Accessories

CPCIS_Cards

CRUVI

Development_Boards

Digital_IO

FMC_Cards

JTAG_Programmer

Modules_and_Module_Carriers

Motherboards_and_Carriers

PCle_Cards

Pinout

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](mailto:support[at]trenz-electronic.de)).

- 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 2.3 Libraries
- Arrow_USB_Programmer_2.4 - Arrow USB Programmer 2.4 Libraries
- Arrow_USB_Programmer_2.5 - Arrow USB Programmer 2.5 Libraries

Files

↓ **Documents (1 Files)**

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

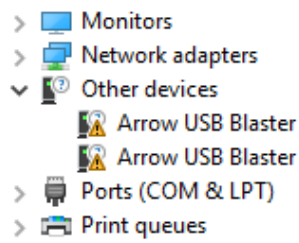
↓ **Diagnose Tool for Win OS (1 Files)**

Arrow_USB_Programmer2-Diagnostic_Program_for_Win_OS.zip
Size 218,36 KB / Modified 15.04.2020 - 16:32:58

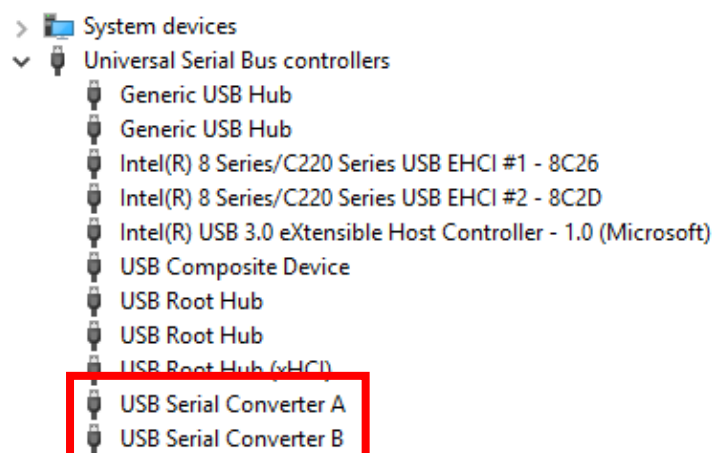
↓ **Other Files (0 Files)**

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.

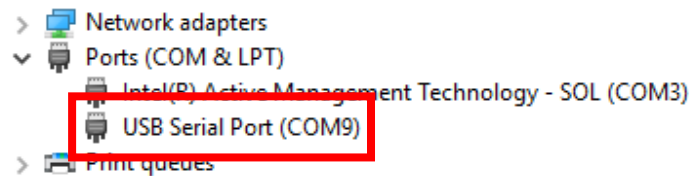
4.2.3 After connecting the CYC5000 board to the PC, two unknown devices might appear in the “Other devices” section of device manager of the PC.



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”:



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



Note that 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.

4.3 License

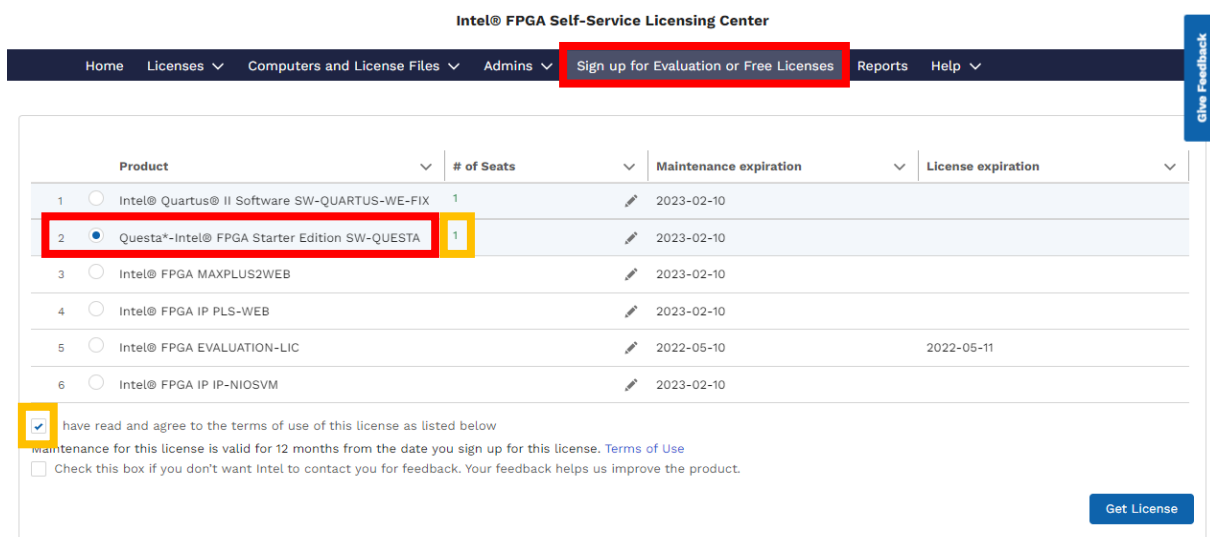
Quartus Lite does not require a license, its use is completely free. However, even though Questa Starter Edition can be used free of charge, you need to generate a free license for it.

4.3.1 Log in to [Intel FPGA Self-Service Licensing Center](#)

4.3.2 Go to Sign up for **Evaluation or Free Licenses** tab.

4.3.3 Select **Questa*-Intel® FPGA Starter Edition SW-QUESTA** option.

4.3.4 Set the seats and accept the terms of use this license.



Product	# of Seats	Maintenance expiration	License expiration
1 <input type="radio"/> Intel® Quartus® II Software SW-QUARTUS-II-FIX	1	2023-02-10	
2 <input checked="" type="radio"/> Questa*-Intel® FPGA Starter Edition SW-QUESTA	1	2023-02-10	
3 <input type="radio"/> Intel® FPGA MAXPLUS2WEB		2023-02-10	
4 <input type="radio"/> Intel® FPGA IP PLS-WEB		2023-02-10	
5 <input type="radio"/> Intel® FPGA EVALUATION-LIC		2022-05-10	2022-05-11
6 <input type="radio"/> Intel® FPGA IP IP-NIOSVM		2023-02-10	

☒ I have read and agree to the terms of use of this license as listed below

Maintenance for this license is valid for 12 months from the date you sign up for this license. [Terms of Use](#)

☐ Check this box if you don't want Intel to contact you for feedback. Your feedback helps us improve the product.

[Get License](#)

4.3.5 Click on Get License button.

4.3.6 In the pop-up window select **+New computer** under Create a New Computer.



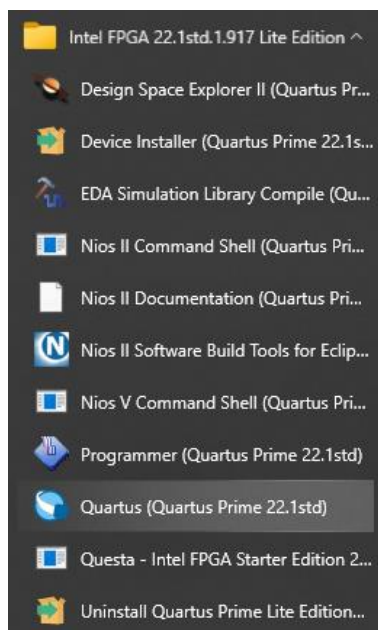
- 4.3.7 In the Create Computer window, fill in the fields with your computer details and click on Generate License.

The license file will be provided by email, or you can also download it under Intel® FPGA Self-Service Licensing Center.

Chapter 5 - New Project with CYC5000

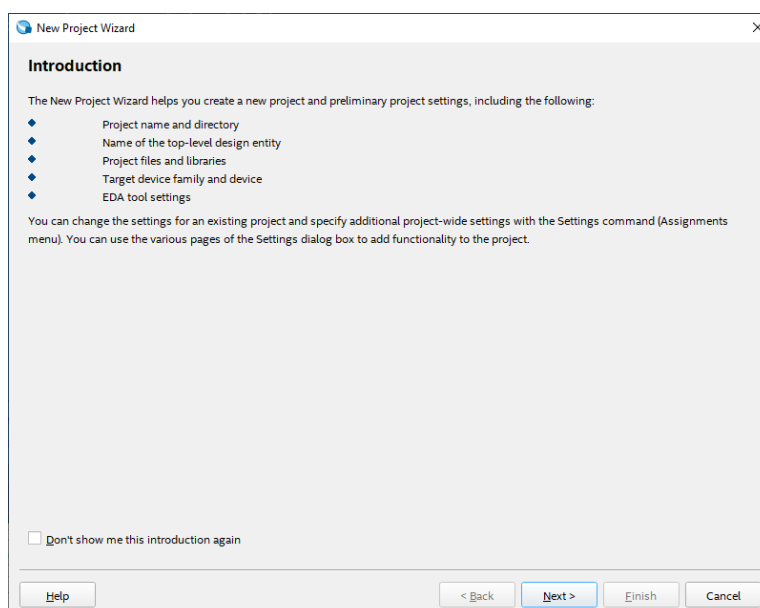
5.1 Creating a new Blinky Project with CYC5000

5.1.1 Launch Quartus Prime Lite Edition from the Start Menu.



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

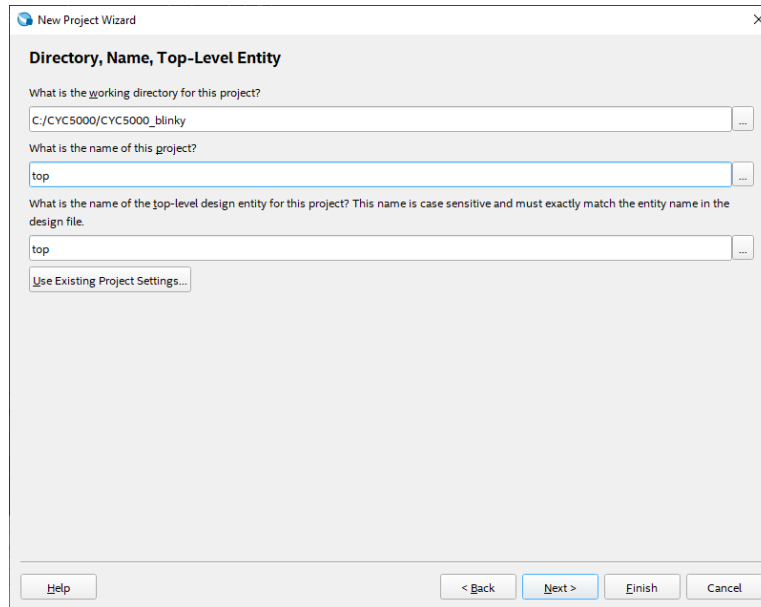
The New Project Wizard walks you through the project settings, such as the name, directories, files, directories, device family and other settings. These settings can be changed later if needed.



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 **CYC5000_blinky**. This will be the folder containing all the project files.

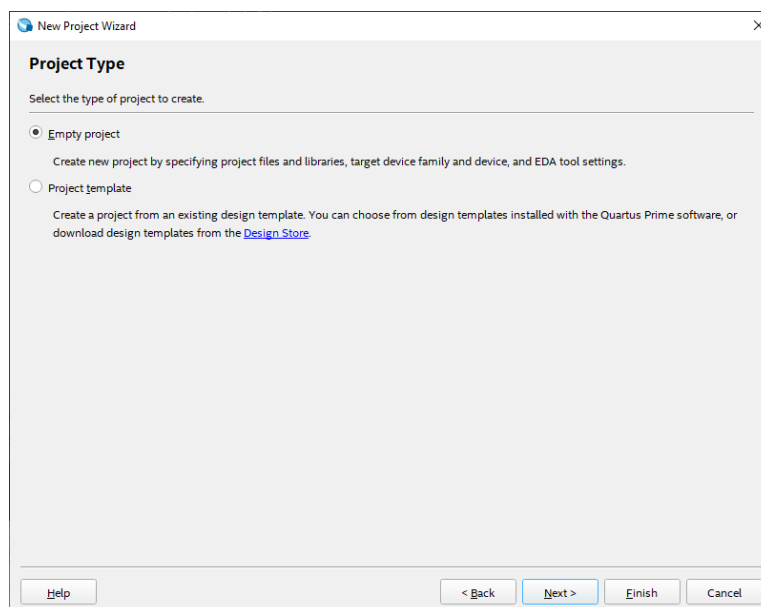
5.1.5 Enter the project name: **“top”** and click **“next”**.



The screenshot shows the 'New Project Wizard' dialog box, specifically the 'Directory, Name, Top-Level Entity' step. It contains three text input fields: 'What is the working directory for this project?' with the value 'C:/CYC5000/CYC5000_blinky', 'What is the name of this project?' with the value 'top', and '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.' with the value 'top'. There is a 'Use Existing Project Settings...' button below the third field. At the bottom, there are buttons for 'Help', '< Back', 'Next >', 'Finish', and 'Cancel'.

5.1.6 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.

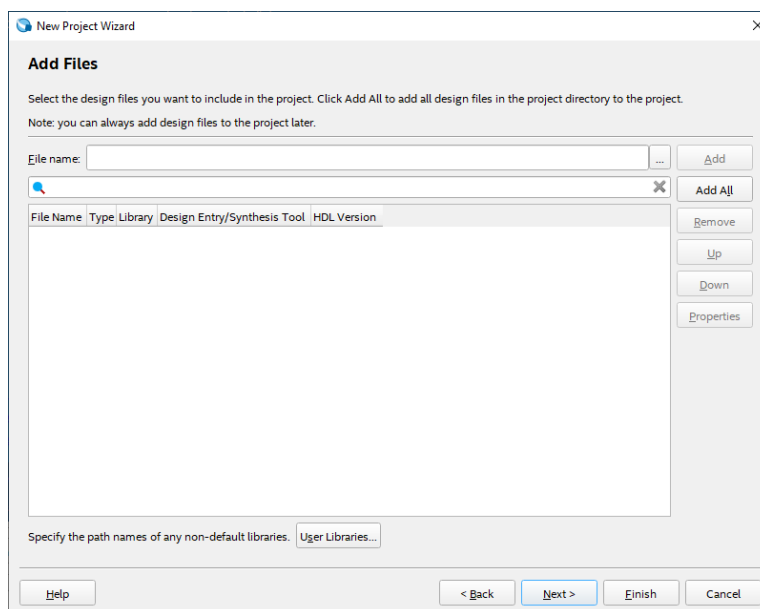


The screenshot shows the 'New Project Wizard' dialog box, specifically the 'Project Type' step. It contains two radio button options: 'Empty project' (selected) and 'Project template'. Below 'Empty project' is the text: 'Create new project by specifying project files and libraries, target device family and device, and EDA tool settings.' Below 'Project template' is the text: '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](#).' At the bottom, there are buttons for 'Help', '< Back', 'Next >', 'Finish', and 'Cancel'.

5.1.7 Click “Next”.

5.1.8 Add Project Files

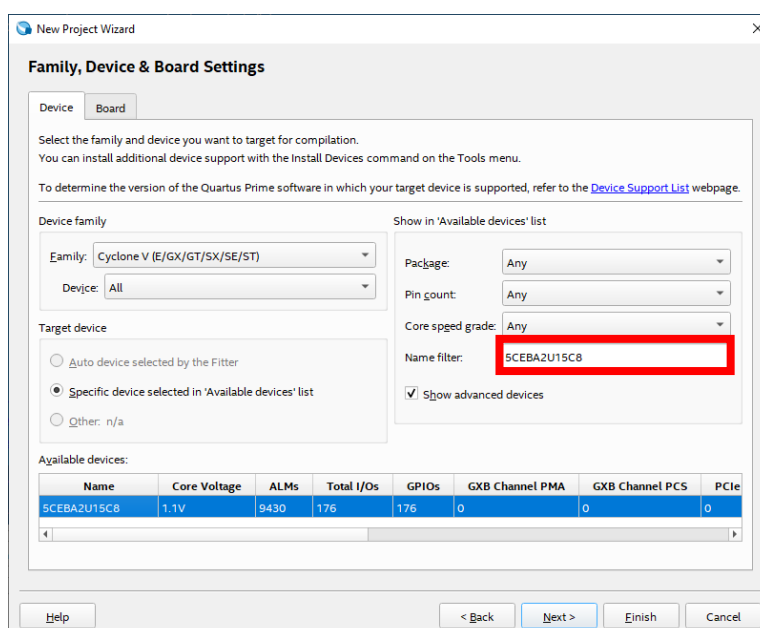
The Add File window will appear. For this tutorial, new design files will be created so no files will be added. For other designs, files could be added here.



5.1.9 Click “Next”.

5.1.10 Select the Device Part Number of the CYC5000 Board

In the Family and Device Settings, use the pull-down menu to select the family as Cyclone V. Then in the Name Filter enter **5CEBA2U15C8**.

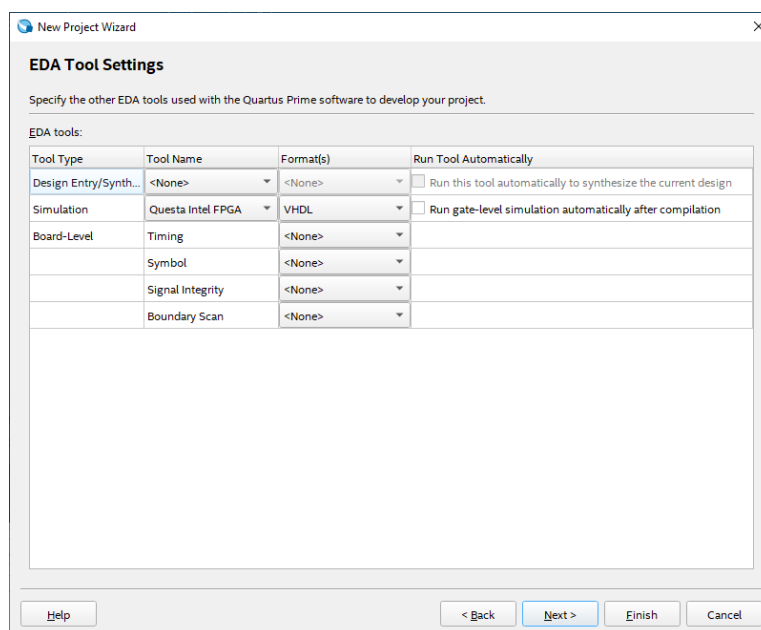


Rather than entering the exact part number, the pull-down menus can be used to select the correct family, package, pin count, and speed grade. Quartus Prime will use these settings to compile the design, and also provide the programming file that you will use later to program the device.

5.1.11 Click “Next”.

5.1.12 EDA Tool Settings

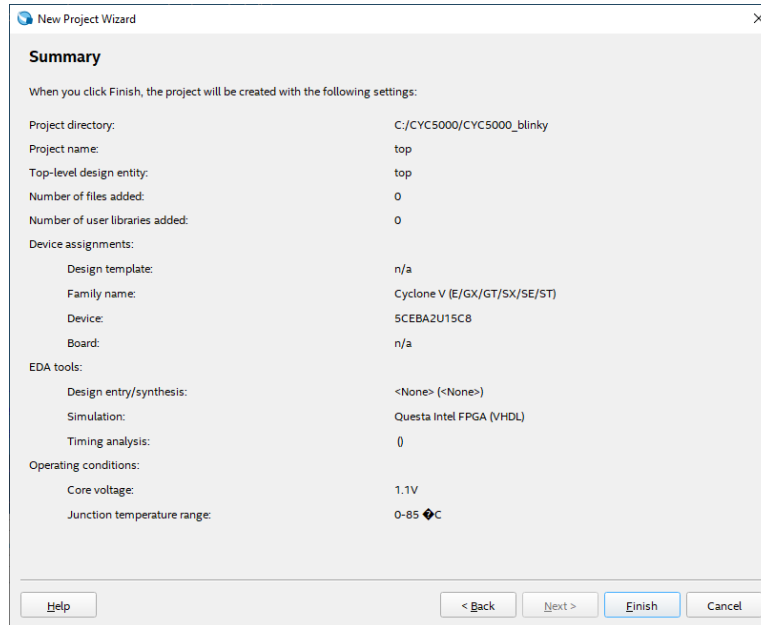
In the EDA tool Settings window, disable any EDA tools, if there are any present. EDA tools are third party tools that work with Quartus Prime for design entry, simulation, verification, and board-level timing. For this tutorial, no EDA software will be used, as only Quartus Prime will be used.



5.1.13 Click “Next”.

5.1.14 Project Summary Page

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



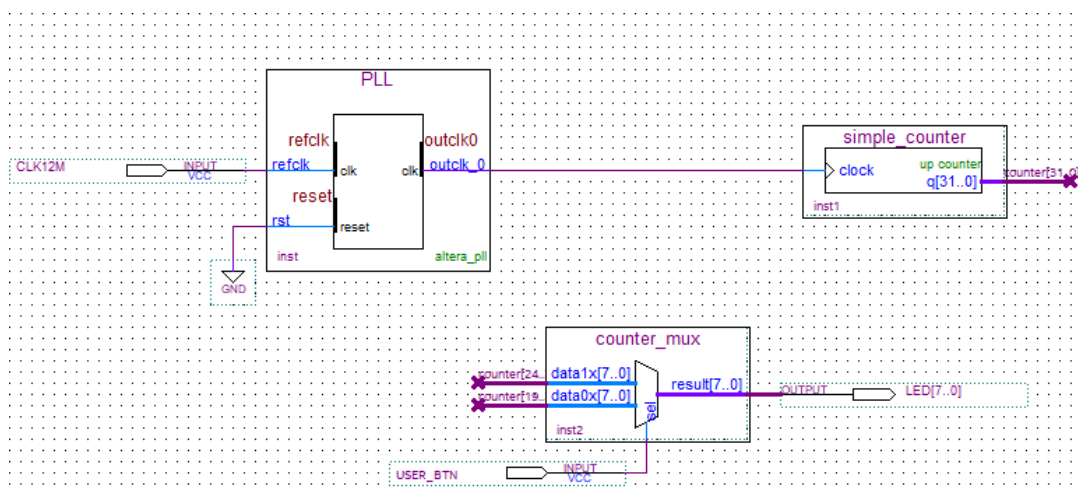
5.1.15 Click “Finish”.

5.2 Building a Blinky Project with CYC5000

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 CYC5000.

5.2.1 Block Diagram

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

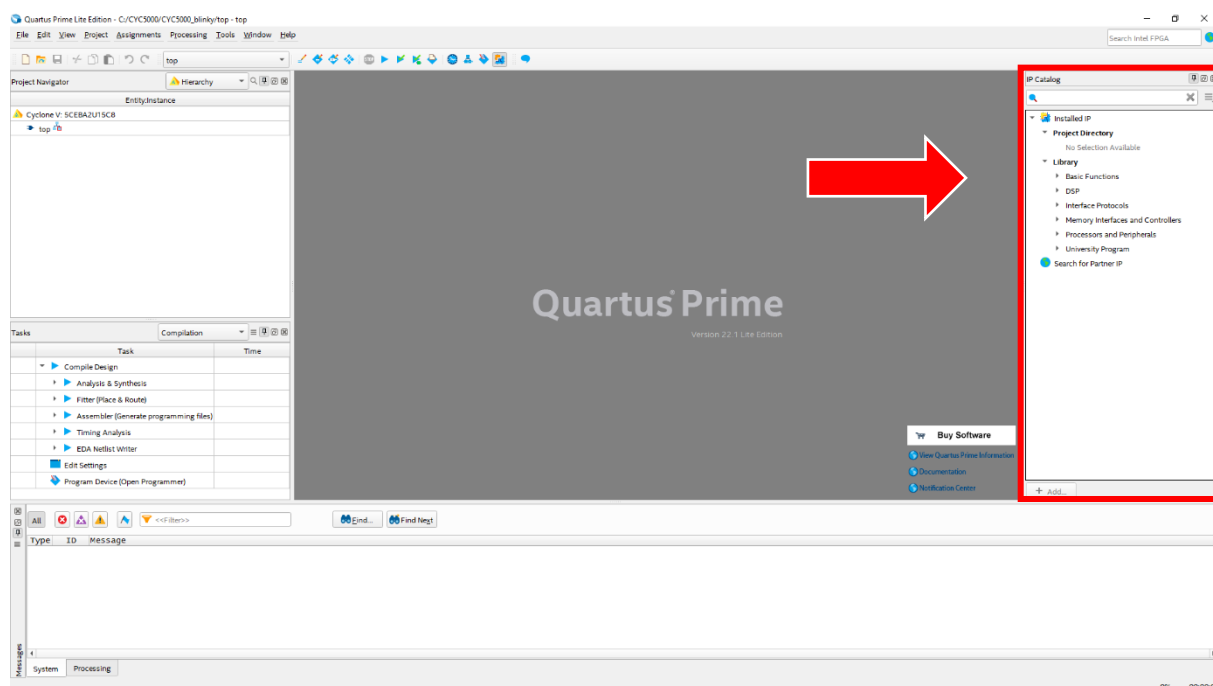


5.2.2 Components of the Design

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

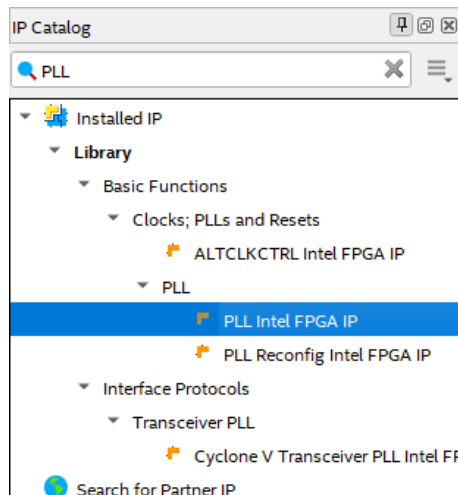
The IP Catalog allows you to create and modify design files with custom variations. The IP Catalog window is open by default when you open Quartus Prime. If it's not present, you can open it by going to the tab **Tool** → **IP Catalog**.



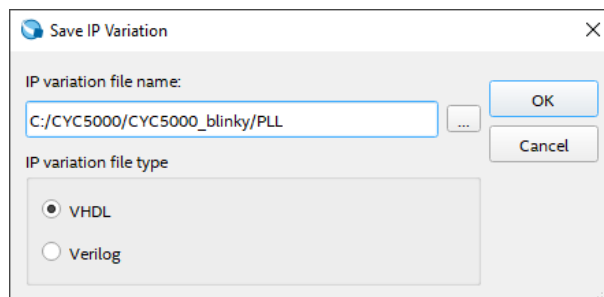
5.2.4 Create and Configure PLL

In the IP Catalog, browse for PLL Intel FPGA IP, 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 **PLL Intel FPGA IP** 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.

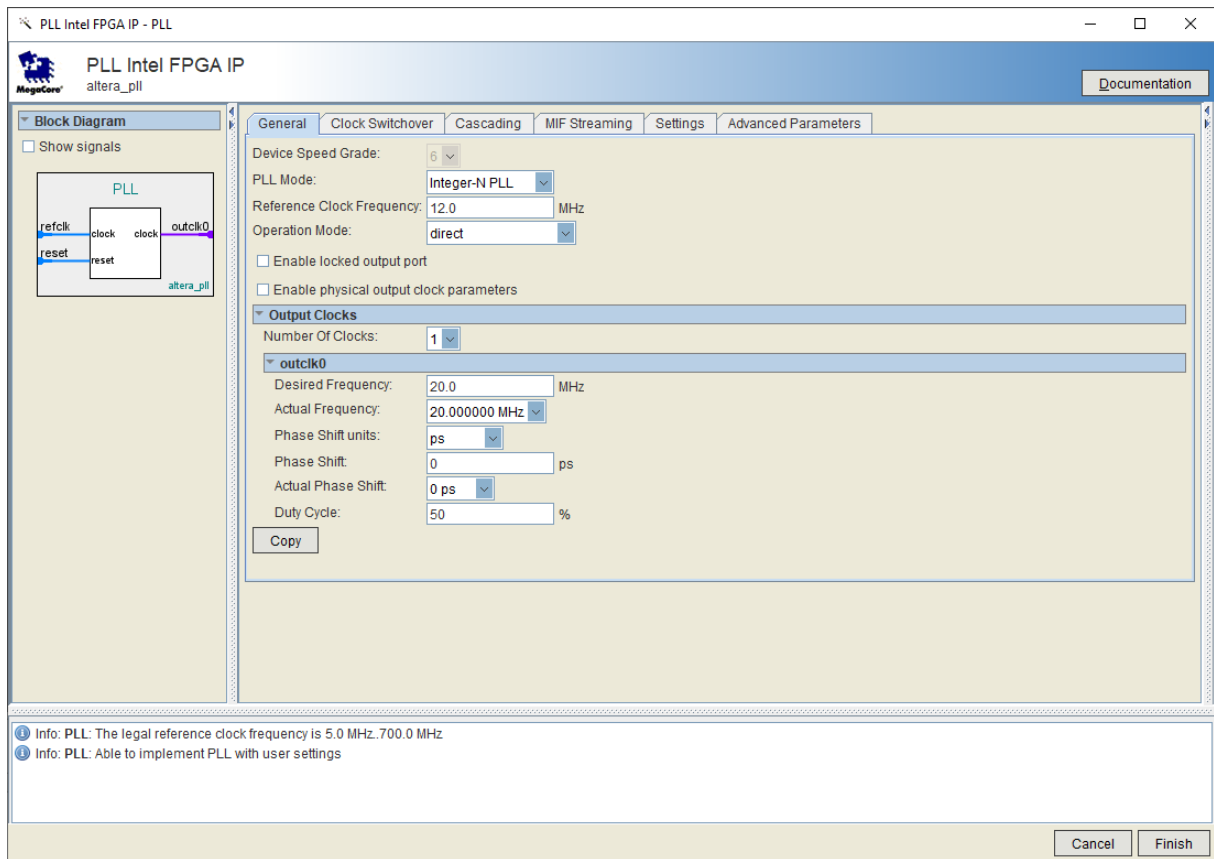


- 5.2.4.3 Click “OK”.

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

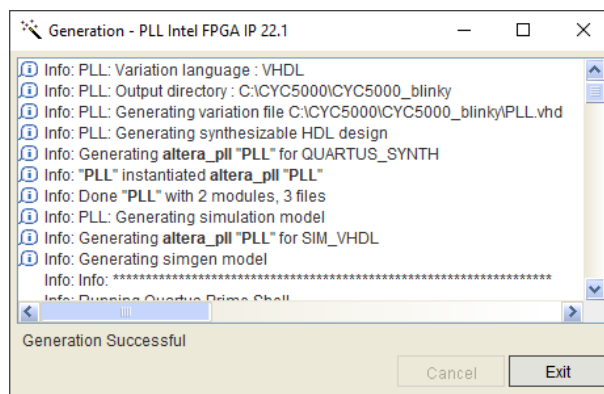
- 5.2.4.4 On the General tab, make sure that “**Integer-N PLL**” is set for PLL mode and enter the PLL reference clock frequency to match the clock input on the CYC5000 Board. We have 12 MHz clock signal coming into the FPGA, so we will use **12MHz** for the Reference Clock Frequency.
- 5.2.4.5 Simplify the PLL by disabling ‘locked output port’.
- 5.2.4.6 Check if the number of clocks set to **1** and set the desired frequency to **20MHz** for outclk0. For simplification, there are two inputs to the PLL (12 MHz and Reset), and one output of the PLL (20 MHz)

The setting should look like this:



5.2.4.7 Leave the rest as default and click “Finish”. The PLL (1st component) will now be created.

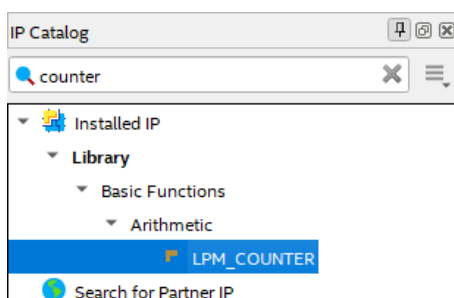
5.2.4.8 If the PLL has been generated correctly, the following window will appear.
Click “Exit” to close it.



5.2.5 Create and Configure the Counter

The next step is to create the counter which will drive the LEDs on the CYC5000 board.

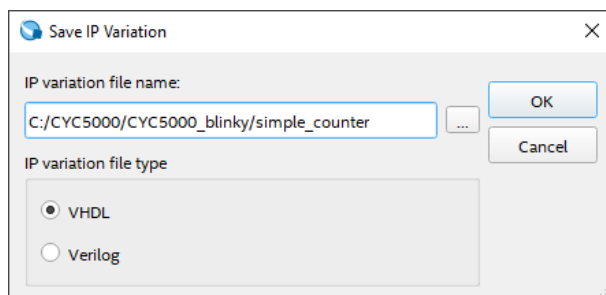
5.2.5.1 To create this counter, select the IP Catalog and expand the Basic Functions → Arithmetic and select the **LPM_COUNTER** or type “counter” in the search field.



Note that the LPM stands for Library of Parameterized Modules

5.2.5.2 Double click on it to add.

5.2.5.3 When the Save IP Variation pop up appears, enter “**simple_counter**” and select VHDL as below:

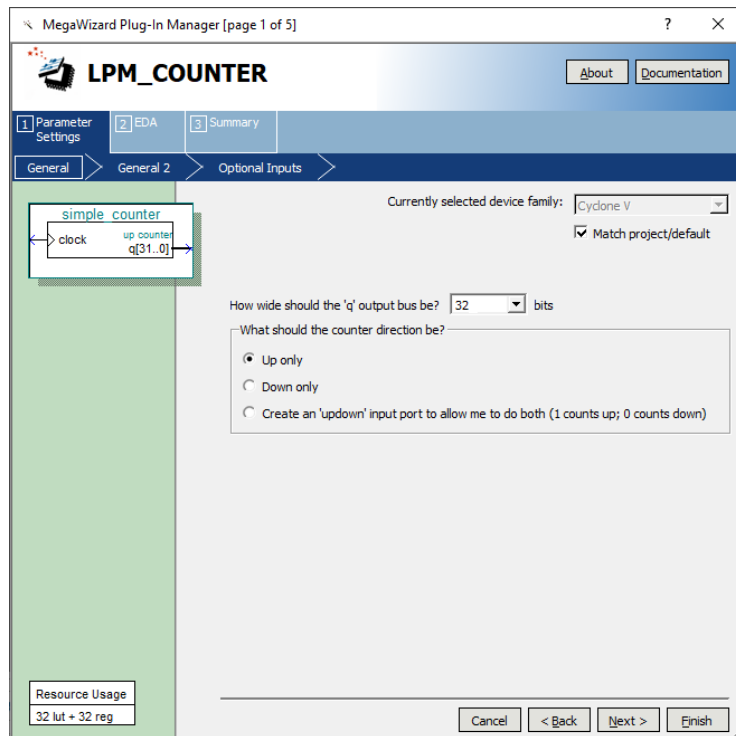


5.2.5.4 Click “**OK**”.

5.2.5.5 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.

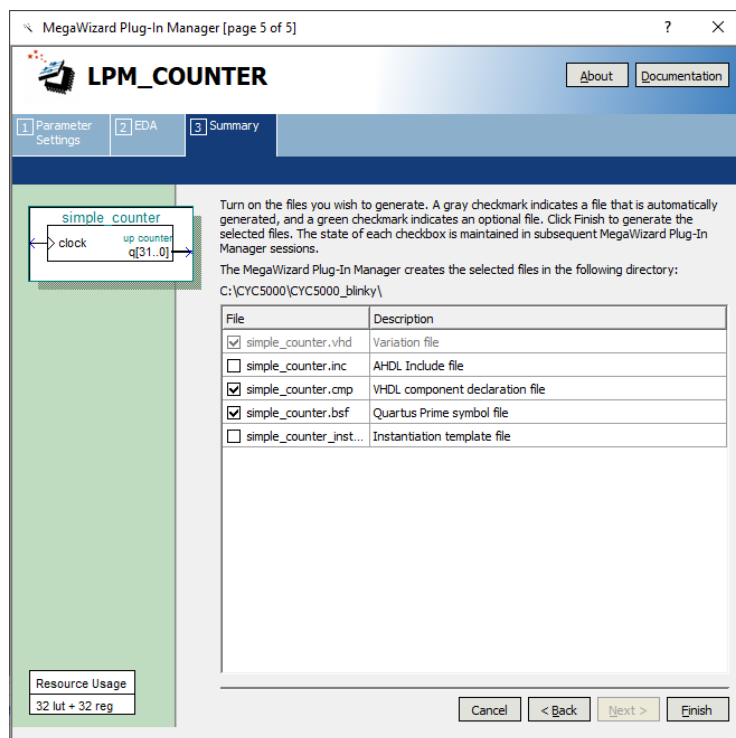
5.2.5.6 Change this number to **32**.

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



5.2.5.8 Select “**Next**” until reaching Page 5.

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



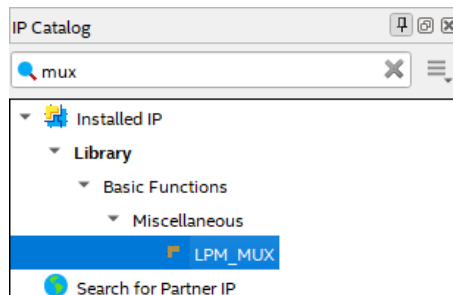
5.2.5.10 Click “**Finish**”.

The counter is now created.

5.2.6 Create and Configure the Multiplexer

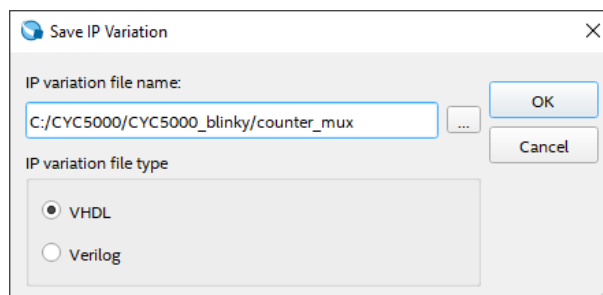
The next step is to create a mux component. This mux will be used along with a push button on the CYC5000 board to control the speed of the counter, where the counter outputs will be seen on the LEDs.

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



- 5.2.6.2 Click **“Add”**.

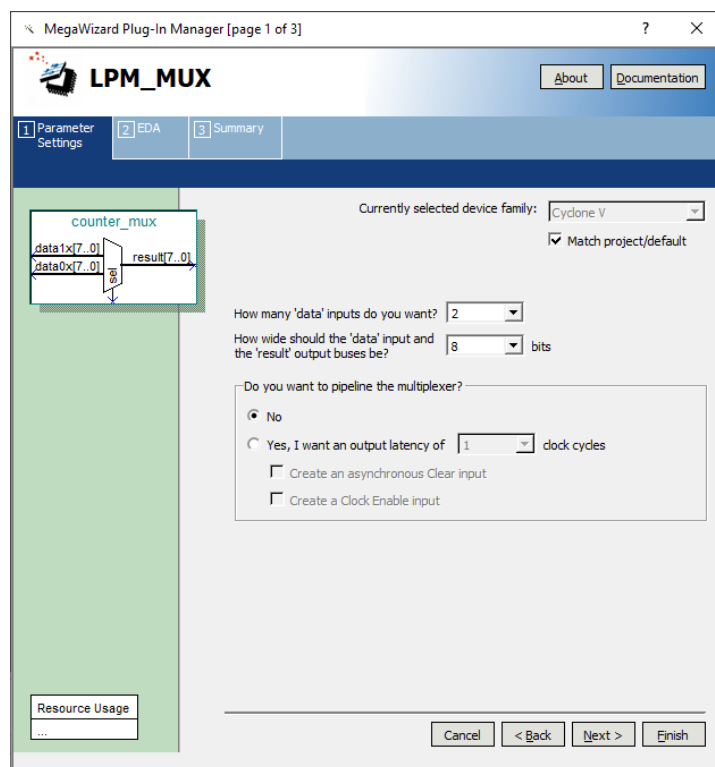
- 5.2.6.3 Enter the name of the **“counter_mux”** and the file type to be VHDL.



- 5.2.6.4 Click **“OK”**.

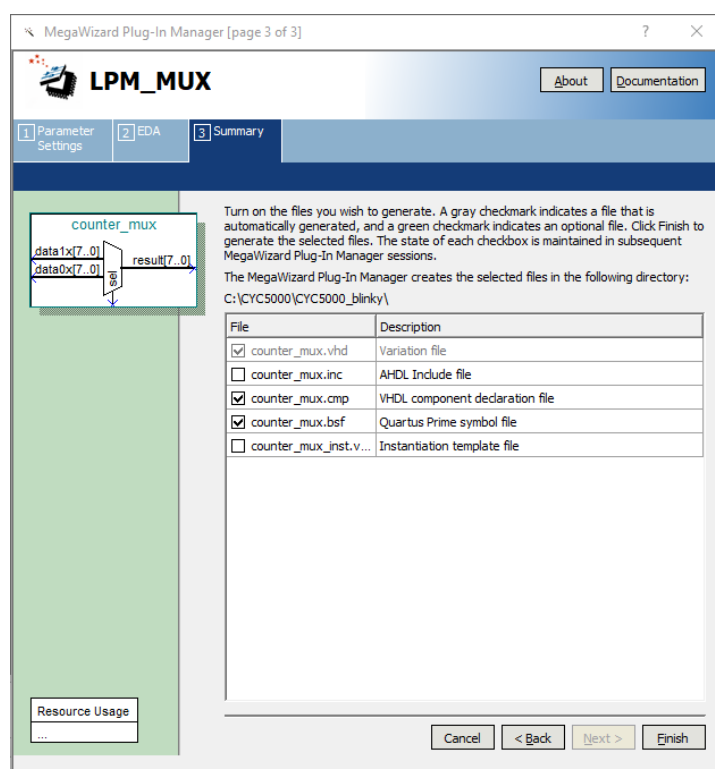
- 5.2.6.5 Select 2 data inputs and the width of the input to be 8 bits. The reason for 8 bits is that there are 8 LEDs to be toggled (showing count values).

The screen should look like this now:



5.2.6.6 Click “Next” until Page 3.

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

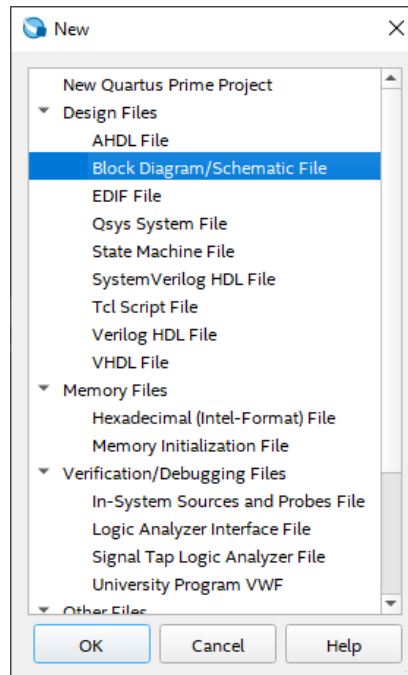


5.2.6.8 Click “Finish”.

5.2.7 Adding the Components to the Schematic

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

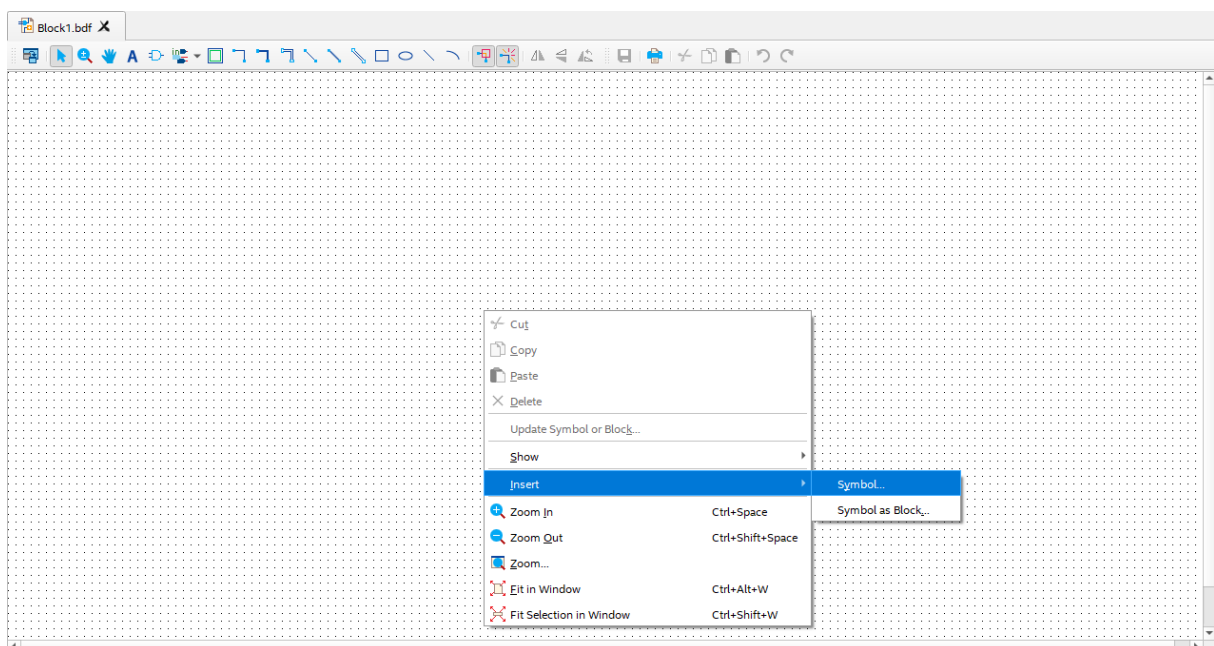
5.2.7.1 To do so, select File menu, then click on New and select **Block Diagram/Schematic File**.



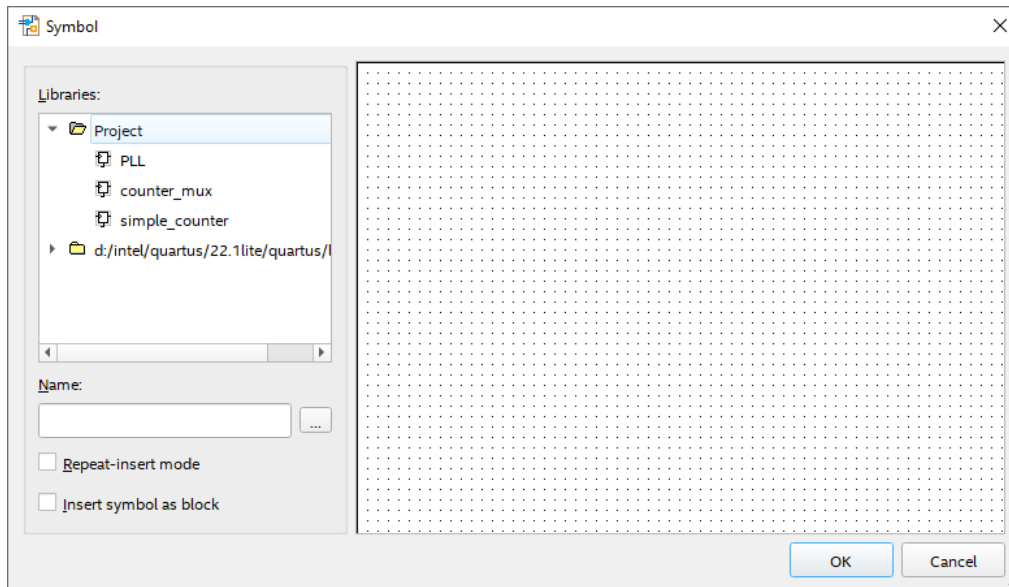
5.2.7.2 Click "OK".

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

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



- 5.2.7.4 In the new window, expand “Project” and the three components that were created can now be seen.



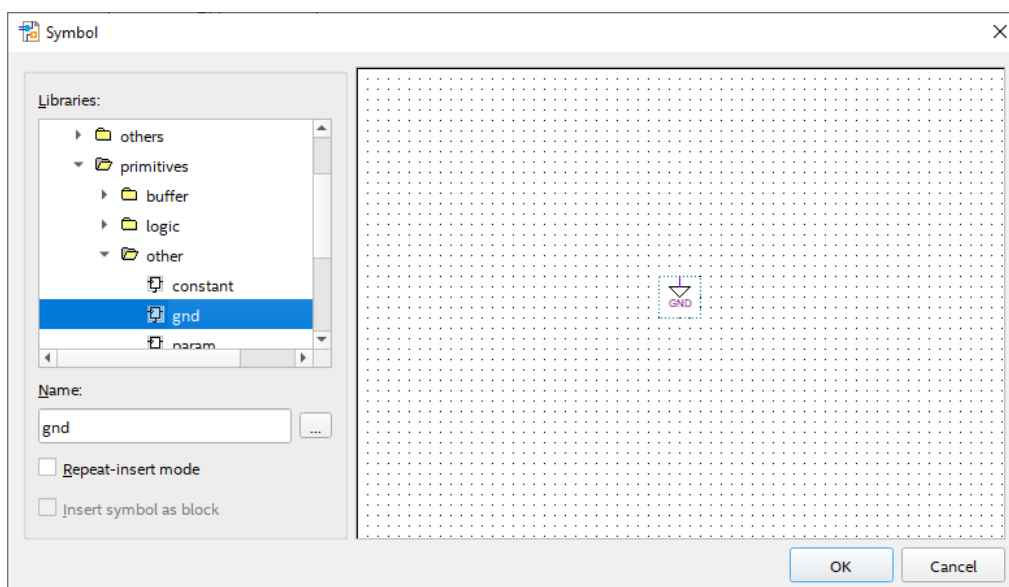
- 5.2.7.5 Select “PLL”.

- 5.2.7.6 Click “OK”.

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

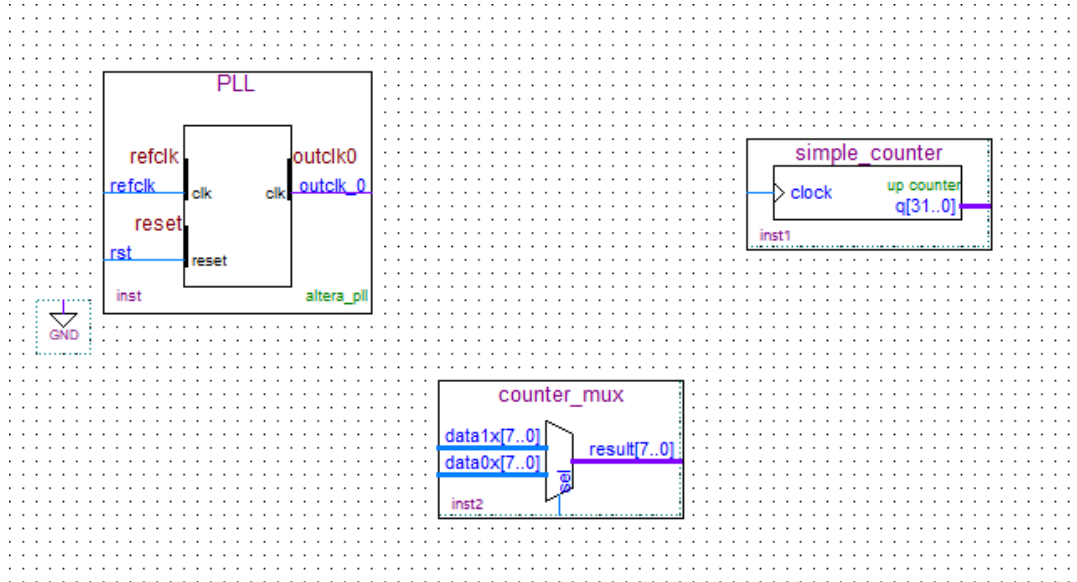
- 5.2.7.8 Just like in the steps from 5.2.7.3 to 5.2.7.6, do the same for **counter_mux** and **simple_counter** to add them to the schematic page. The order of adding the components does not matter, as the connections between them will happen in the following steps.

- 5.2.7.9 Repeat the step 5.2.7.3 but this time expand the Quartus folder, and search for primitives → other, and select “gnd”.



5.2.7.10 Click “OK”.

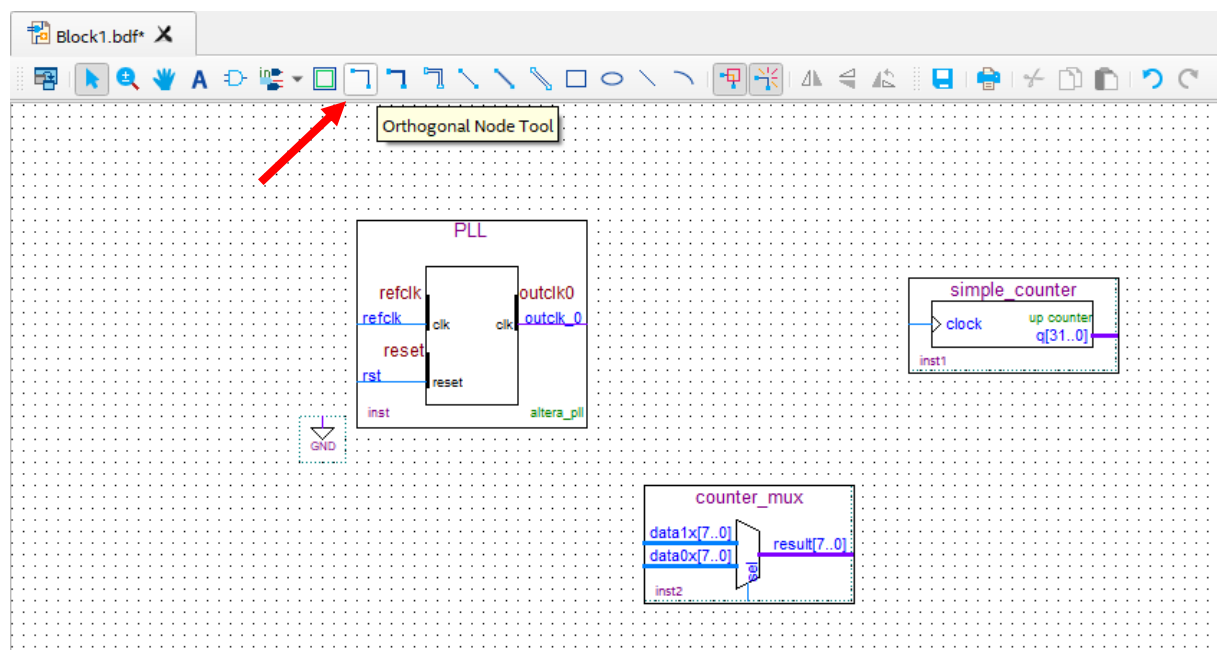
5.2.7.11 After adding the components, your schematic should look similar to the following. To place them similarly, simply drag the components to the appropriate locations.



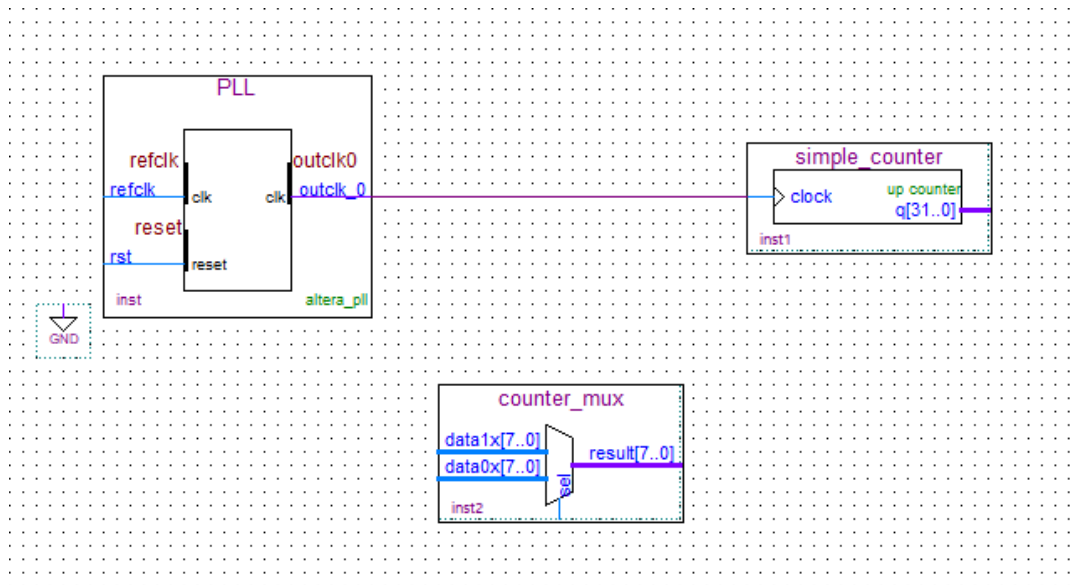
5.2.8 Connecting the Components

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

5.2.8.1 Select the “Node Tool”.



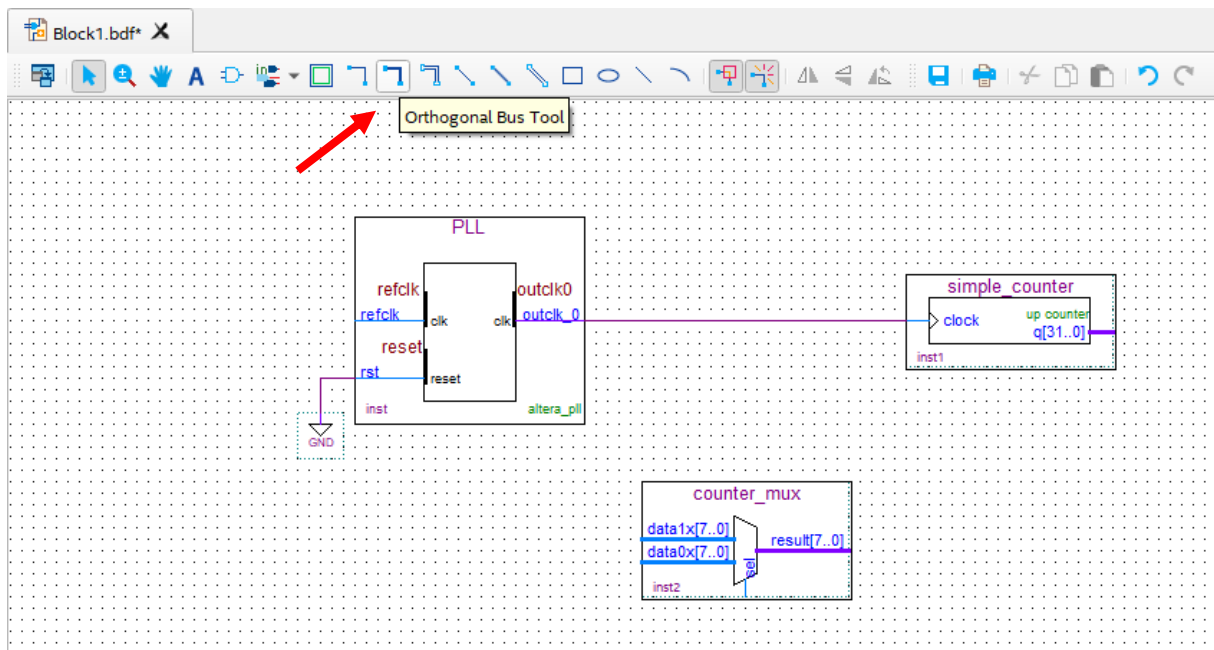
5.2.8.2 Connect the outclk0 of the PLL to the clock input of the simple_counter as shown below:



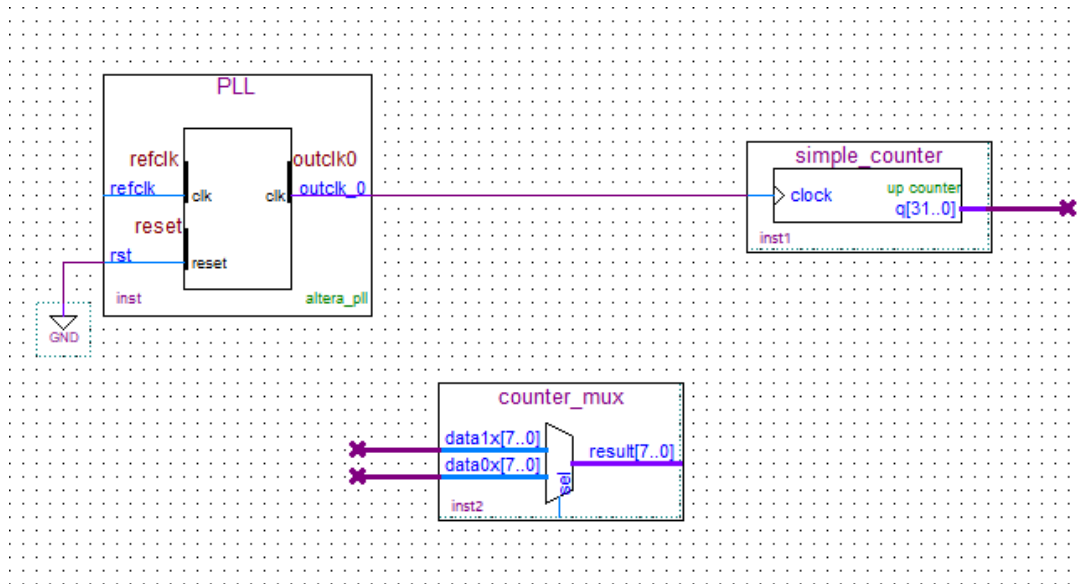
This will mean that a single signal (c0) is connected to the simple_counter (clock).

5.2.8.3 Repeat the 5.2.8.1 step and connect the reset of PLL to the GND.

5.2.8.4 Select the “Bus Tool”.



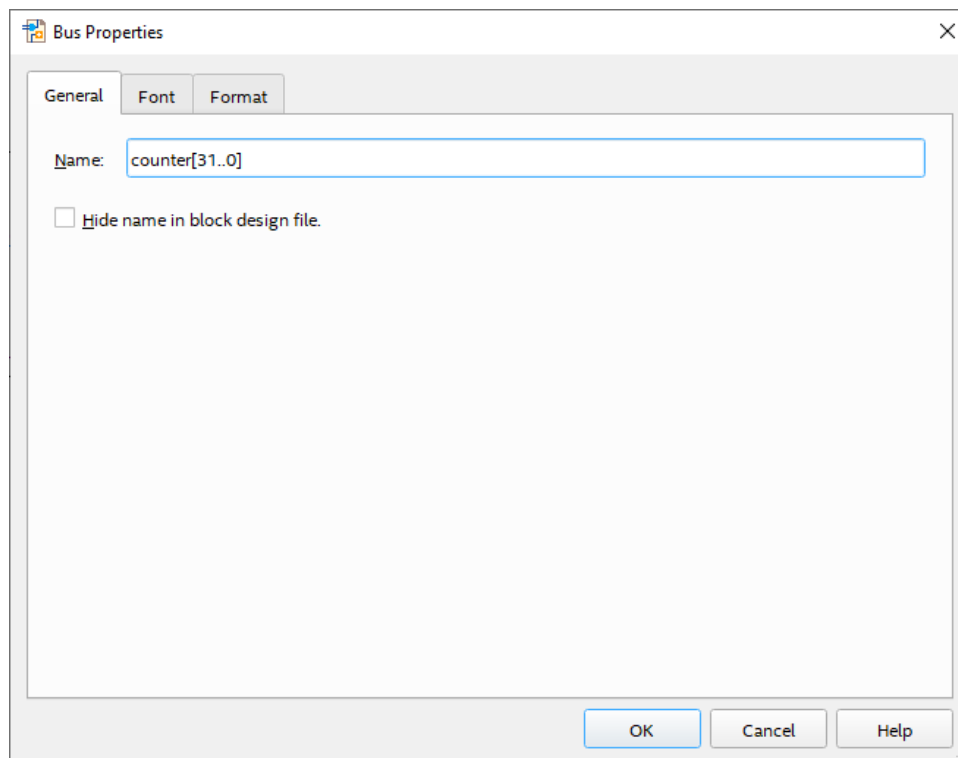
5.2.8.5 Using the bus tool create a connection coming out of the simple_counter and one connection for each of the inputs of the counter_mux as show below.



5.2.8.6 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:



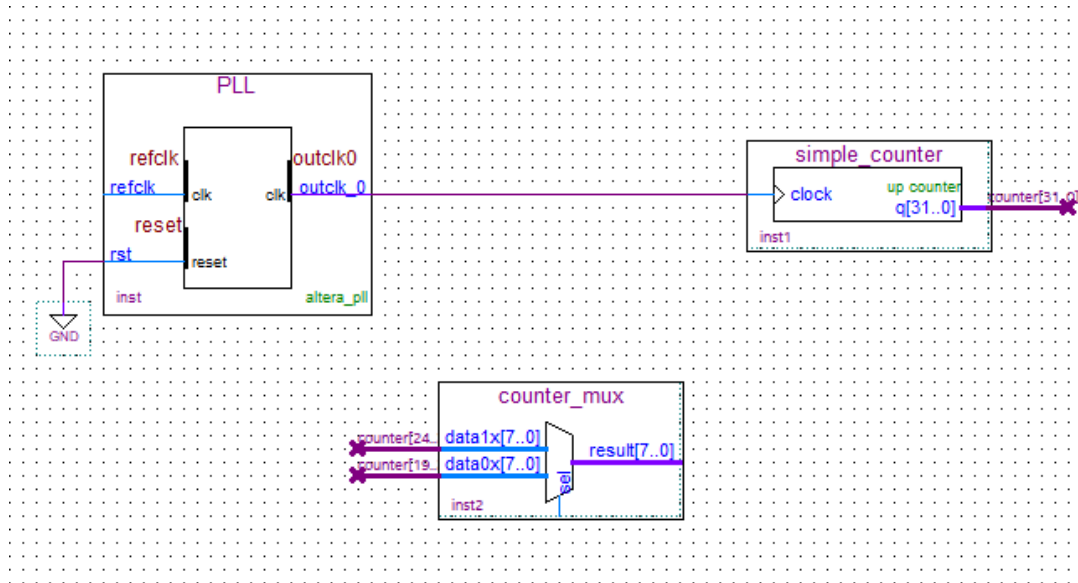
5.2.8.7 Click “OK”.

5.2.8.8 Do the same for input buses of the mux:

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

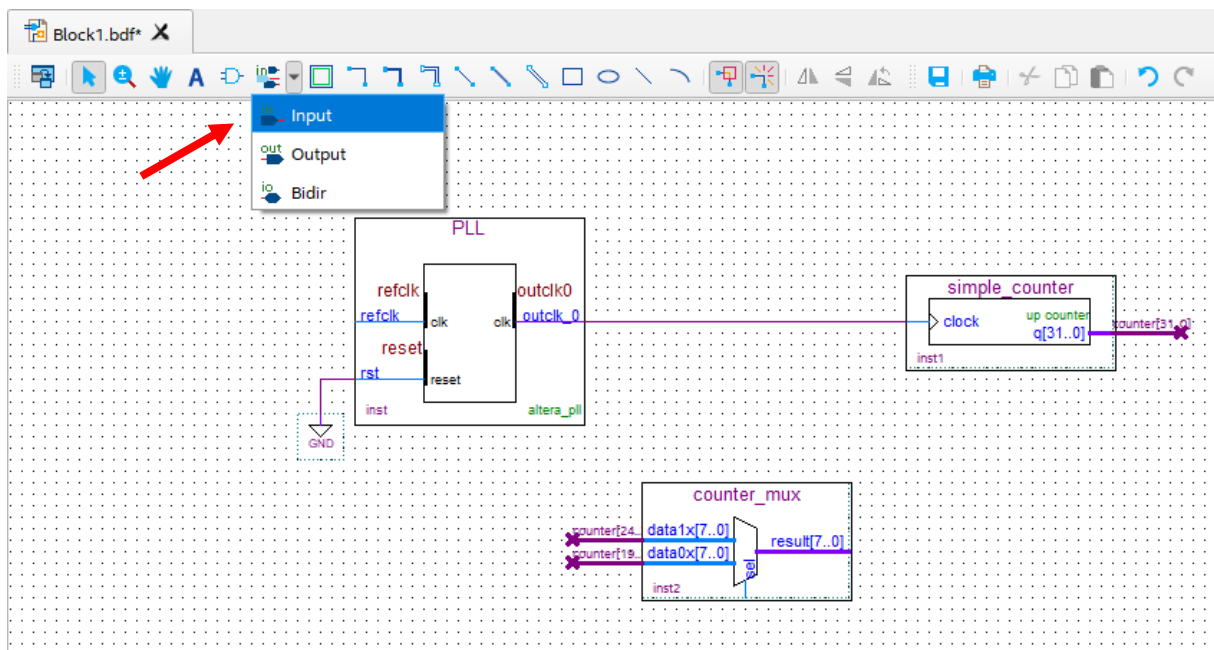
Name the bottom bus input: **data0x[7..0] → counter[19..26]**

Schematic should look like this:



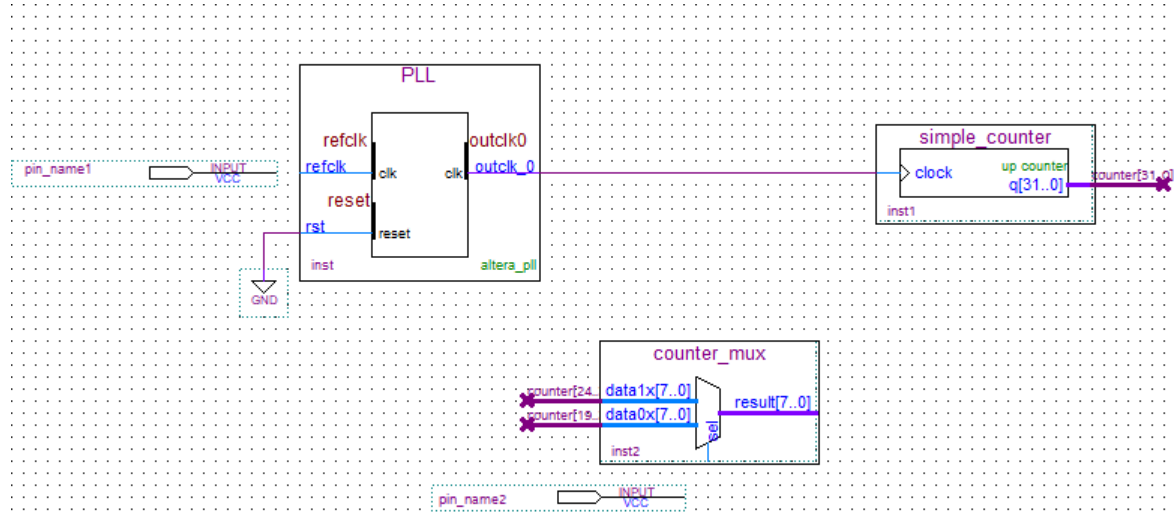
5.2.9 Add inputs, outputs to the schematic

5.2.9.1 Click on the “Pin Tool” as show below and select “Input”.



- 5.2.9.2 Add one input pin for refclk of the PLL and add other one input pin for sel of counter_mux.

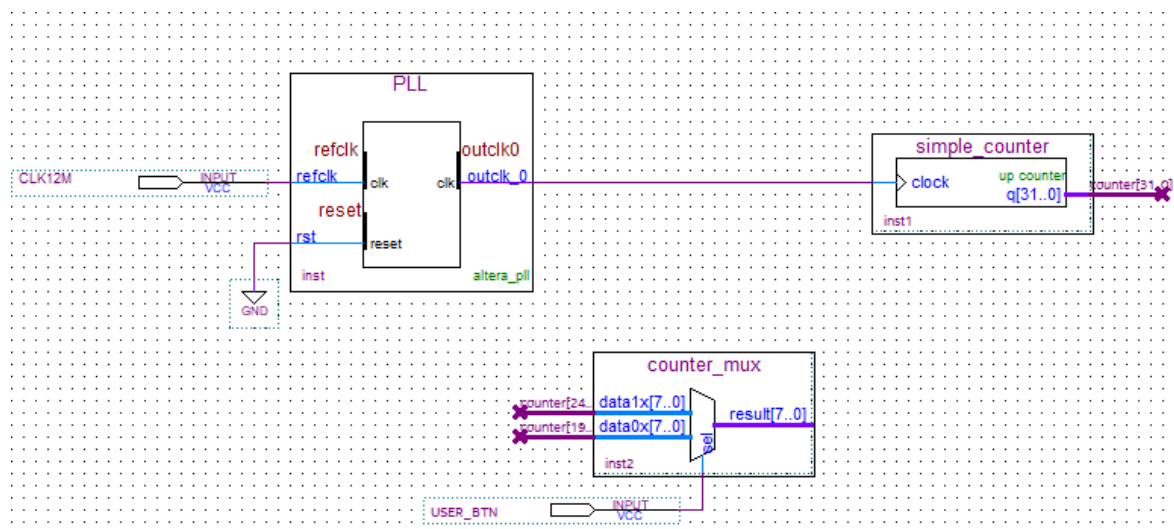
Your schematic should look like this:



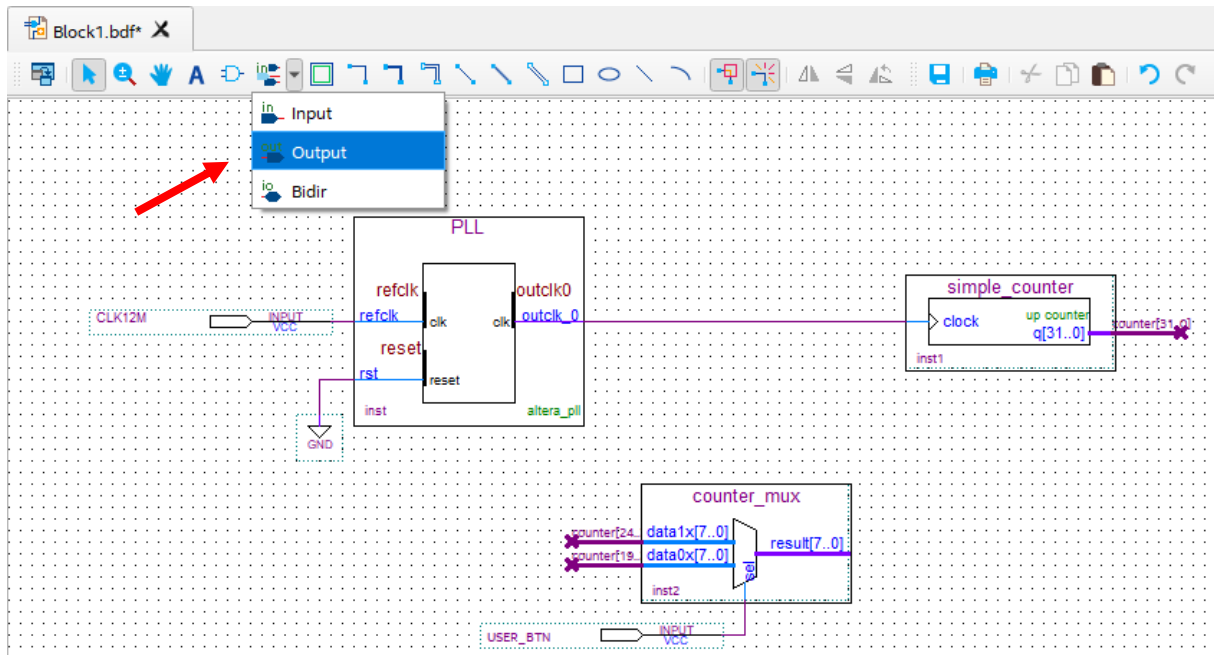
- 5.2.9.3 Rename the pin_name1 to **CLK12M** by double clicking its current name. This is going to be the clock signal coming into the FPGA.
- 5.2.9.4 Rename the pin_name2 to **USER_BTN** by double clicking its current name. This is going to be one of the user buttons of the CYC5000 board to select the mux.
- 5.2.9.5 Using the “Node Tool” connect:

CLK12M → refclk (of the PLL component)
 USER_BTN → sel (of the counter_mux component)

Your schematic should look like this now:



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



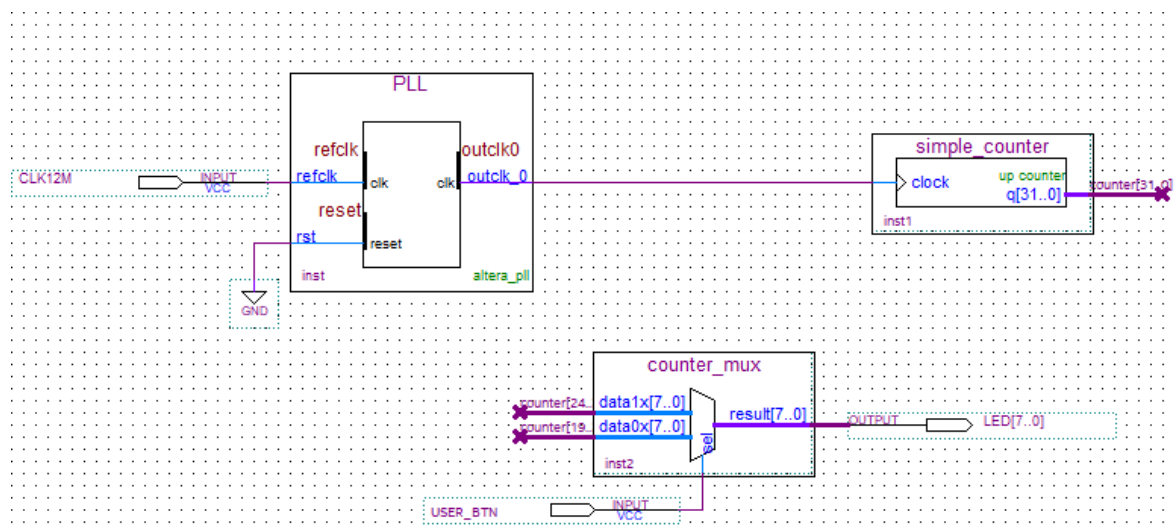
5.2.9.7 Add one output pin for the LEDs.

5.2.9.8 Rename the pin to LED[7..0].

5.2.9.9 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.

5.2.9.10 Save your design. Open the File Menu and select “**Save**”. Save it as **top.bdf**.

5.2.10 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 clicking  button on the top toolbar.

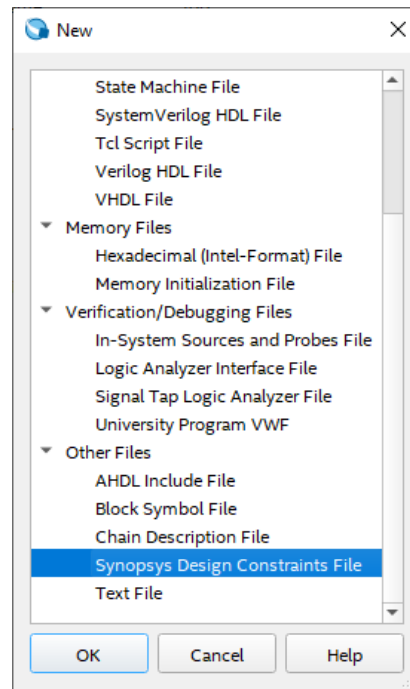
There should be no errors. If there are errors, they should be fixed before continuing and Analysis and Synthesis run again.

Tasks			Compilation				
	Task		Time				
	▶	Compile Design					
✓	▶	Analysis & Synthesis	00:00:24				
	▶	Fitter (Place & Route)					
	▶	Assembler (Generate programming files)					
	▶	Timing Analysis					
	▶	EDA Netlist Writer					
		Edit Settings					
		Program Device (Open Programmer)					

5.2.11 Adding Timing Constraints

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

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



5.2.11.2 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.

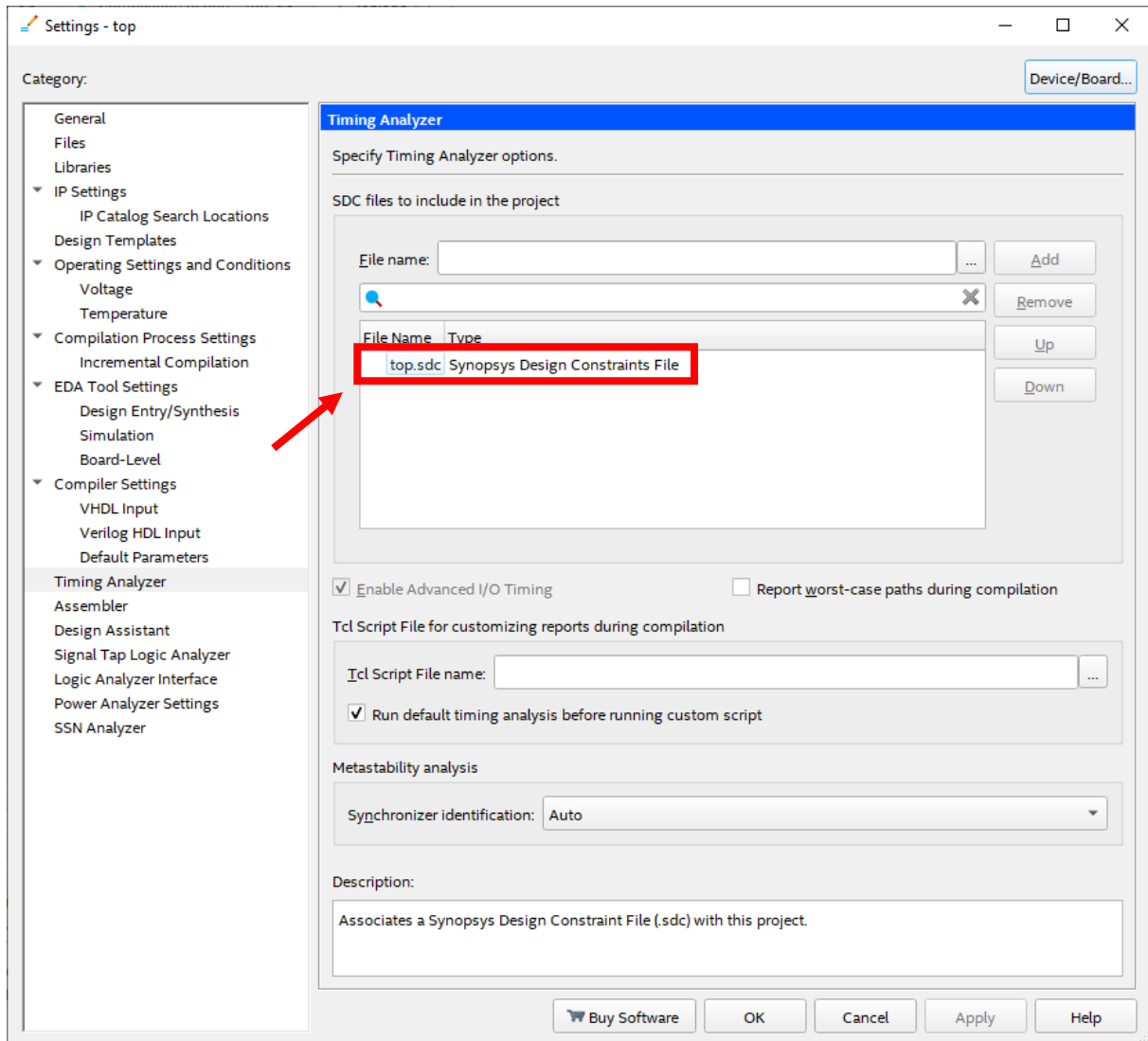
The second line “derive_pll_clocks” tells the software to look if there are any PLLs, and if so, automatically derive the clock multiplication/division of the outputs of the PLL even if they are used internally within the CPLD/FPGA.

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.

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

- 5.2.11.4 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.



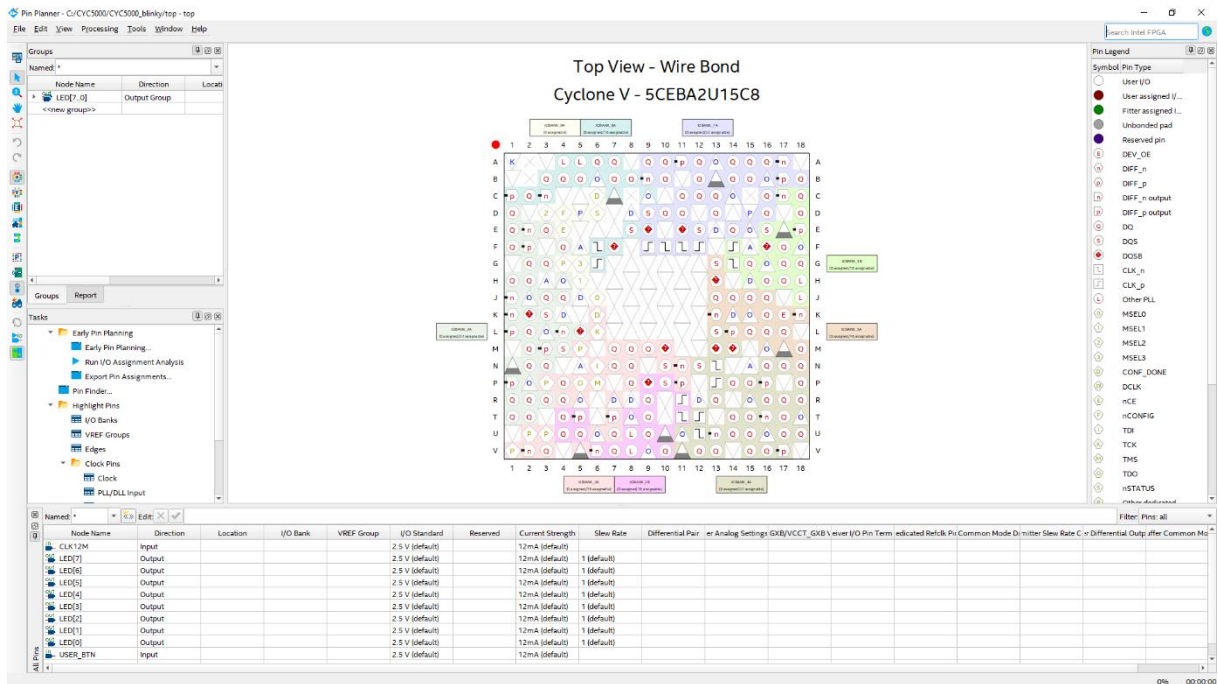
5.2.12 Pinning Assignments

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.

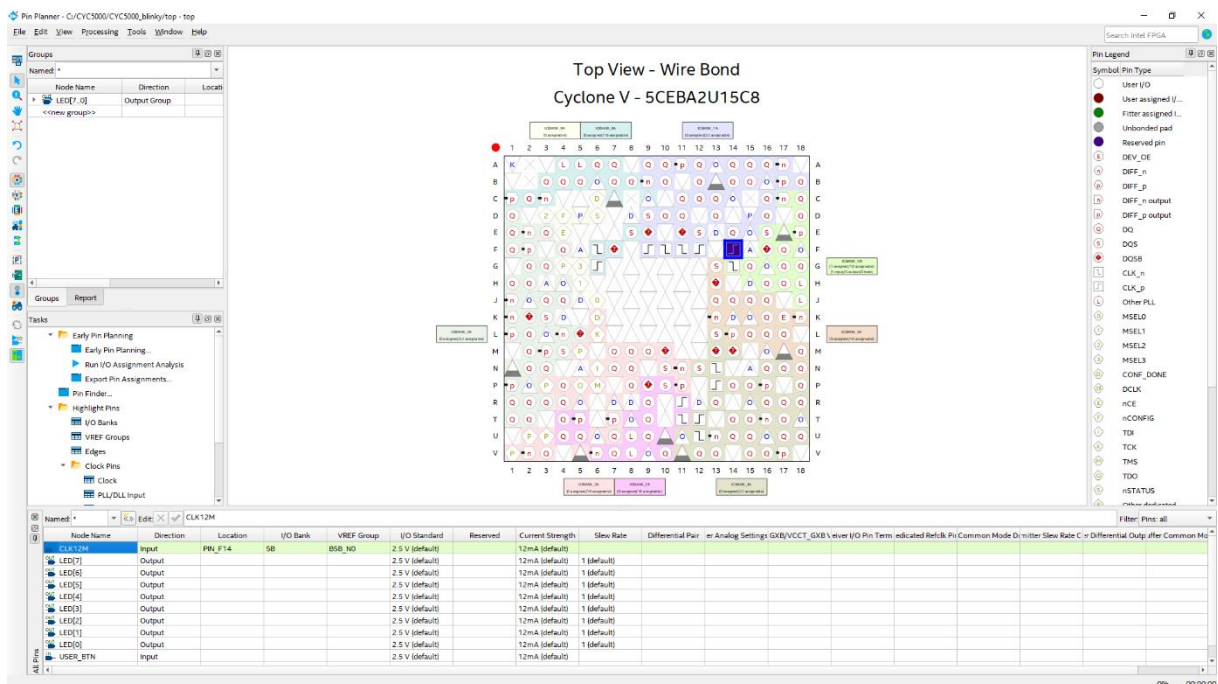
The following steps will show one of these ways, the Pin Planner. Since there are only 10 pins that need to be assigned, the Pin Planner can be used. If many pins are needed, other ways can be used such as the Quartus Assignment Editor, or by importing constraints from a text file or spreadsheet.

- 5.2.12.1 Open the Pin Planner: **Assignments** → **Pin Planner**.

A new window will open as seen below:



5.2.12.2 To make pin assignments, select the CLK12M (node name) on the bottom portion and drag and drop it to pin F14 of the Top View of the FPGA or alternatively set the Location field of the CLK12M to **PIN_F14**.

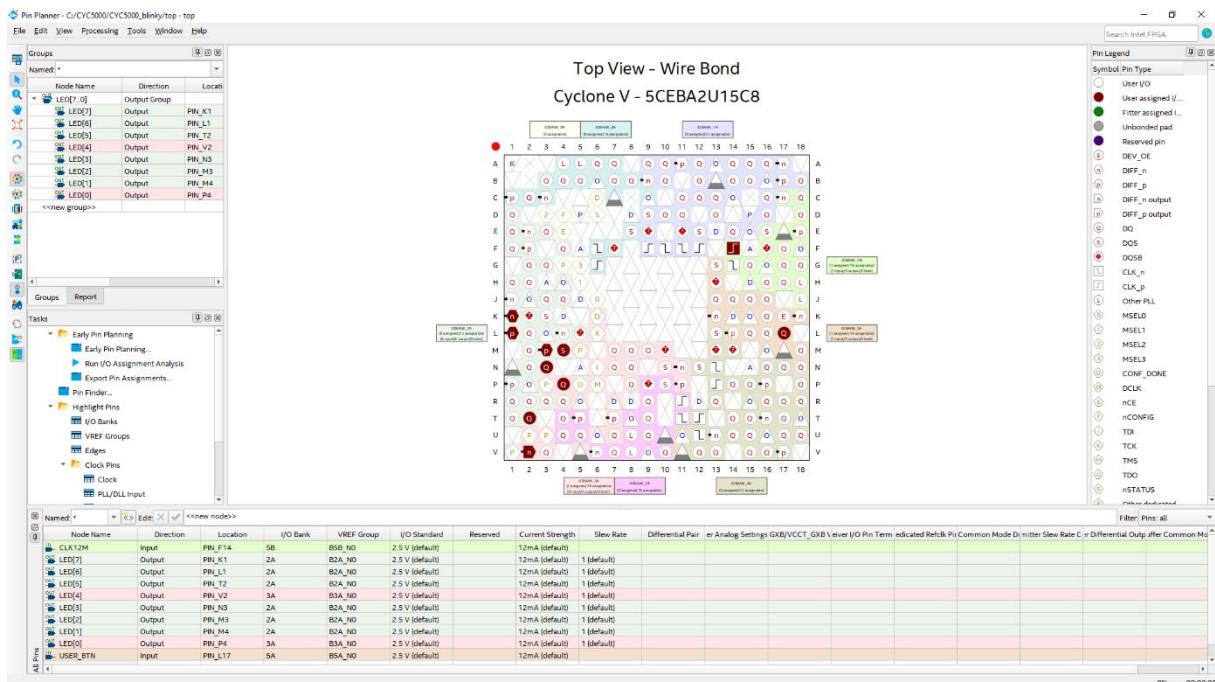


Note that the Location of the CLK12M is now set to Location PIN_F14 (as seen in blue colour in the top view of the FPGA).

5.2.12.3 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.

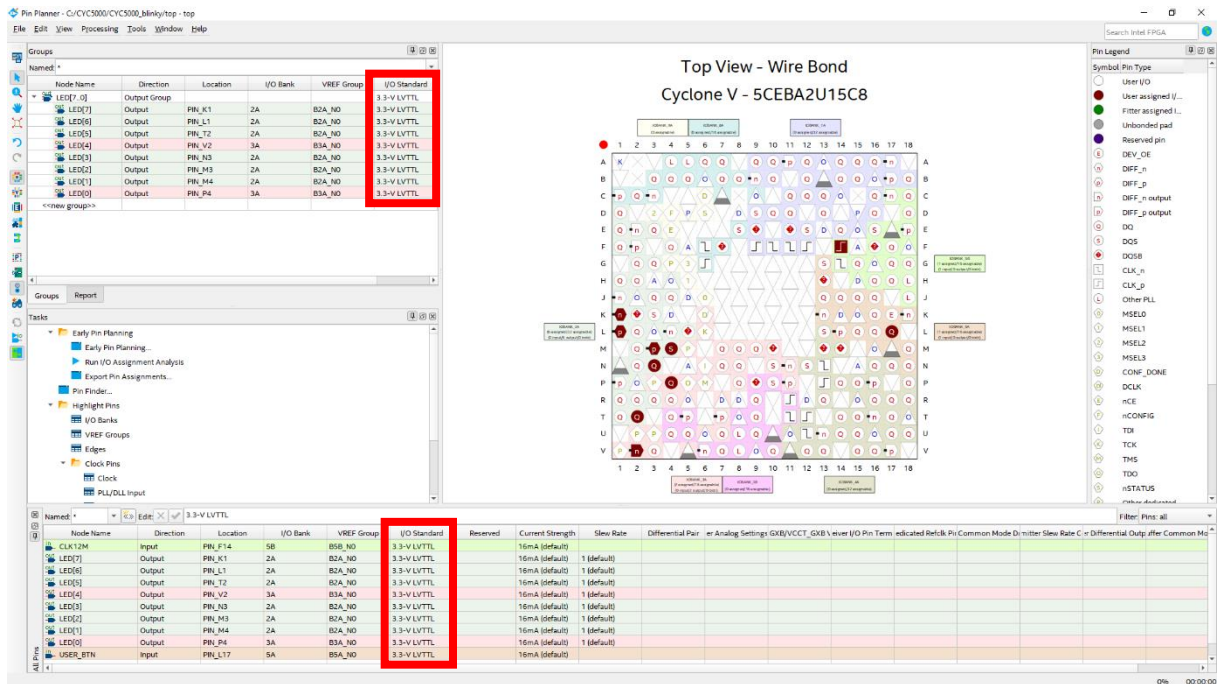
Node Name	Pin Location
LED[7]	PIN_K1
LED[6]	PIN_L1
LED[5]	PIN_T2
LED[4]	PIN_V2
LED[3]	PIN_N3
LED[2]	PIN_M3
LED[1]	PIN_M4
LED[0]	PIN_P4
USER_BTN	PIN_L17

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



5.2.12.5 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 CYC5000 since all banks* and peripherals are powered by 3.3V. The USER_BTN, the LEDs and clock pins are 3.3-V LVTTTL. 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:

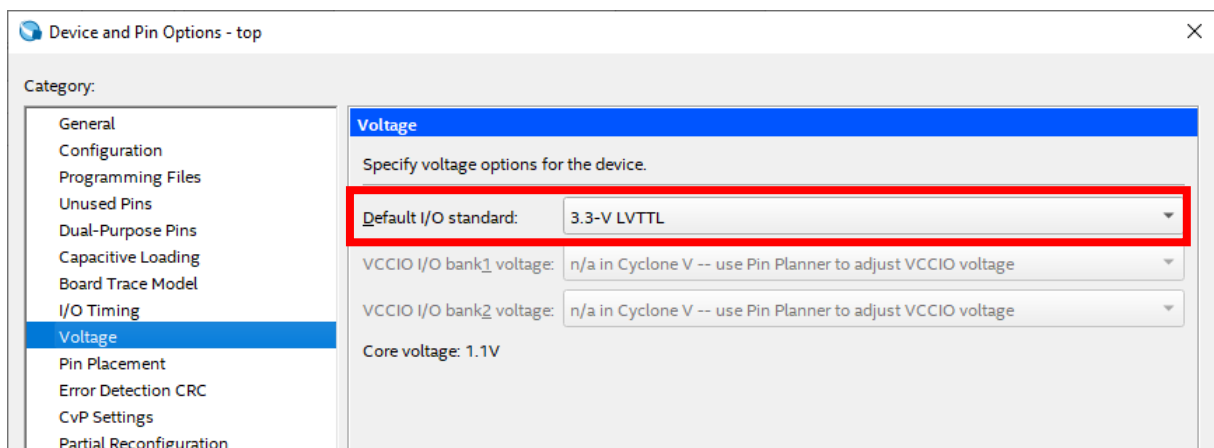


5.2.12.6 Close the Pin Planner. The settings are automatically saved.


5.2.13 Compiling the Design

5.2.13.1 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 LVTTTL” and press “OK” to all the windows.



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.

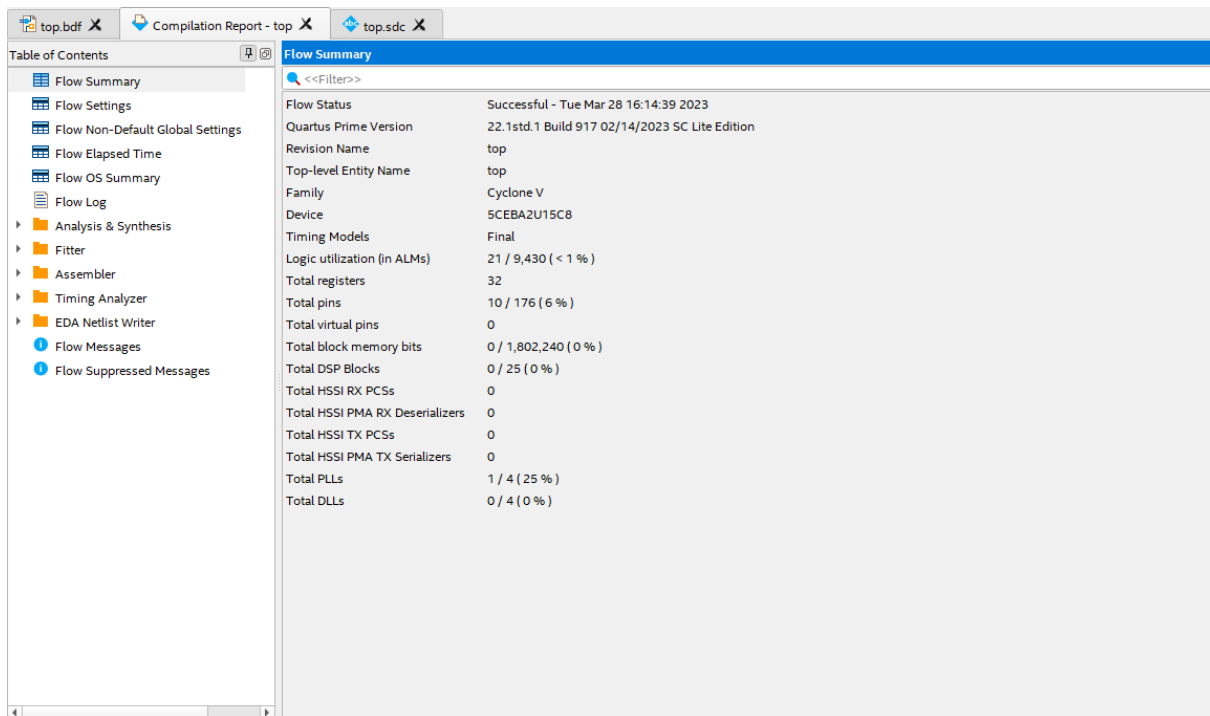
5.2.13.2 To compile the design, select **Processing** → **Start Compilation** or push the  button on the toolbar.

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.

Tasks			Compilation				
	Task		Time				
✓	▶	Compile Design	00:01:13				
✓	▶	▶ Analysis & Synthesis	00:00:25				
✓	▶	▶ Fitter (Place & Route)	00:00:26				
✓	▶	▶ Assembler (Generate programming files)	00:00:08				
✓	▶	▶ Timing Analysis	00:00:09				
✓	▶	▶ EDA Netlist Writer	00:00:05				
		■ Edit Settings					
		🔧 Program Device (Open Programmer)					

5.2.14 Reading the Compilation Report

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



Flow Summary	
Flow Status	Successful - Tue Mar 28 16:14:39 2023
Quartus Prime Version	22.1std.1 Build 917 02/14/2023 SC Lite Edition
Revision Name	top
Top-level Entity Name	top
Family	Cyclone V
Device	5CEBA2U15C8
Timing Models	Final
Logic utilization (in ALMs)	21 / 9,430 (< 1 %)
Total registers	32
Total pins	10 / 176 (6 %)
Total virtual pins	0
Total block memory bits	0 / 1,802,240 (0 %)
Total DSP Blocks	0 / 25 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	1 / 4 (25 %)
Total DLLs	0 / 4 (0 %)

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:

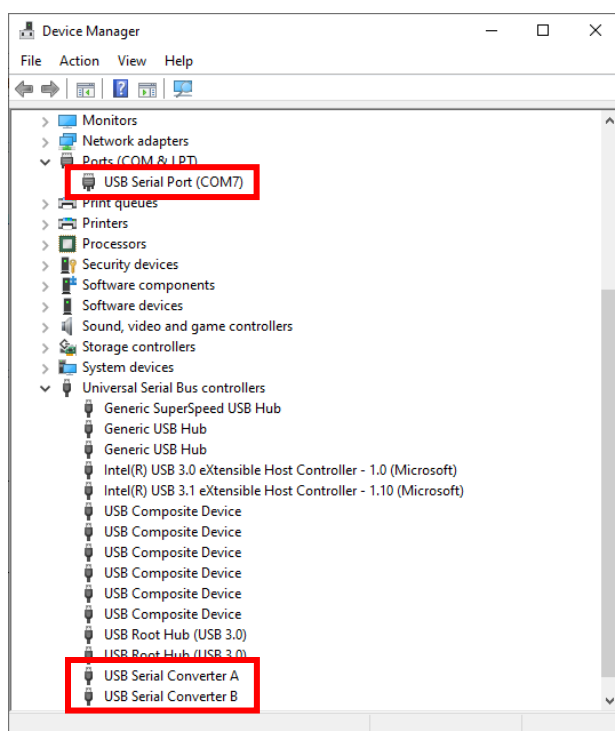
- In the Flow Summary, it can be seen how many logic elements the whole design took, along with total PLLs, registers, pins, etc.
- In Analysis and Synthesis, more detailed information about the resources used can be seen in Resource Usage Summary, as well how many LEs were used for each component in 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 CYC5000

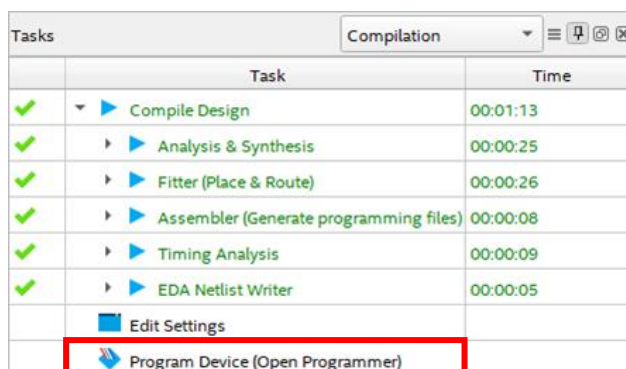
After successfully compiling your project, there should new files be generated. In case of Cyclone V devices, only the .sof file is generated automatically.

6.1 Configure the FPGA in JTAG mode

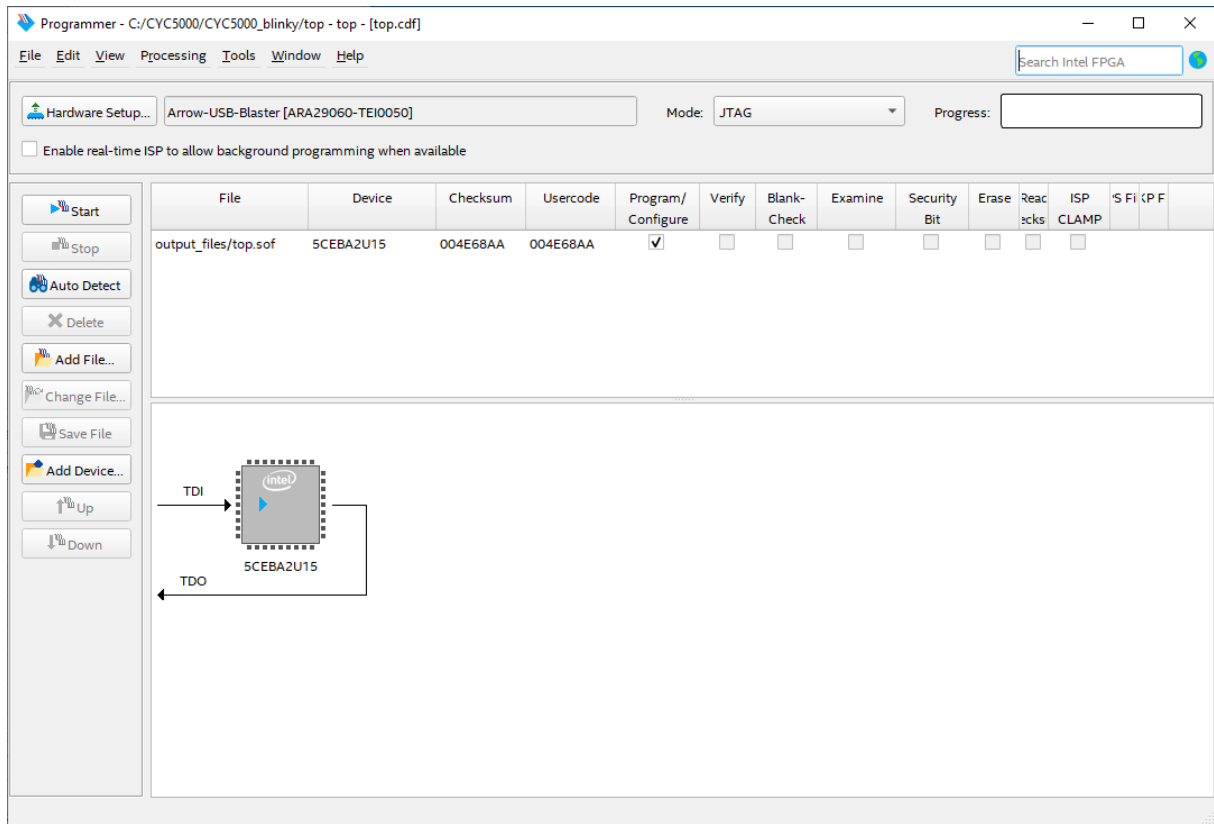
- 6.1.1 Connect your CYC5000 board to your PC using a USB cable. Since the Arrow USB Blaster should be already installed, the Window's Device Manager should display the following entries are highlighted in red (port number may differ depending on your PC). If the Arrow USB Blaster is not installed, please 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.

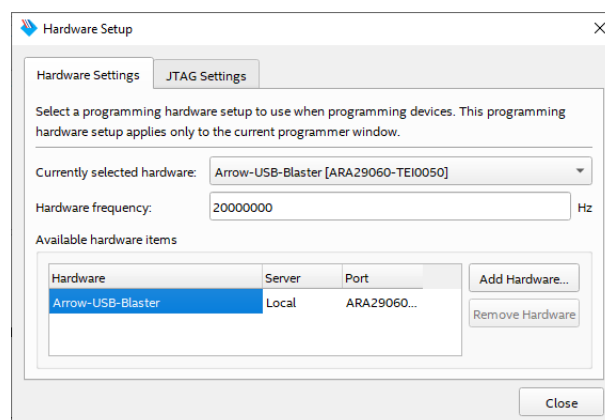


6.1.3 The programmer should add the programming file automatically. After opening the program this should be the view of the new window:



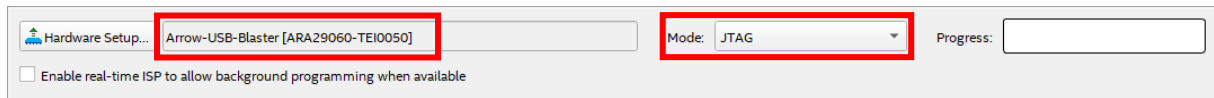
6.1.4 If the hardware or programming file was not found automatically, continue with the steps below to add the missing parts. Otherwise, continue from step 6.1.12.

6.1.5 To add hardware, 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 vary).

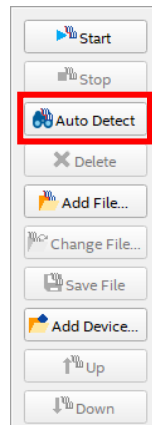


6.1.6 Click "Close".

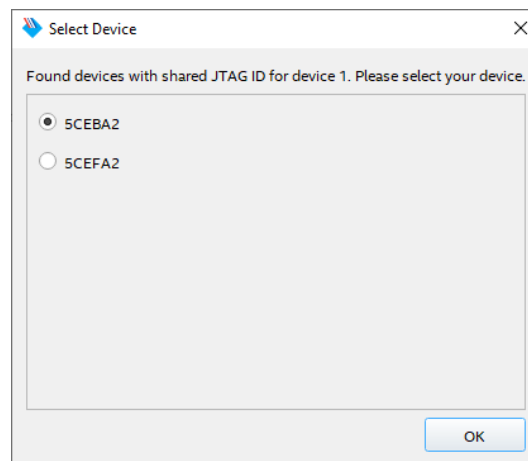
- 6.1.7 Make sure the hardware setup is **Arrow-USB-Blaster [USB0]** and the mode is **JTAG**. If the Mode is not set to JTAG, click on it, and select JTAG from the drop-down menu.



- 6.1.8 To add the device, click “**Auto Detect**” on the left side of the Programmer.



- 6.1.9 Select **5CEBA2** device and click “**OK**” on the Select Device window.

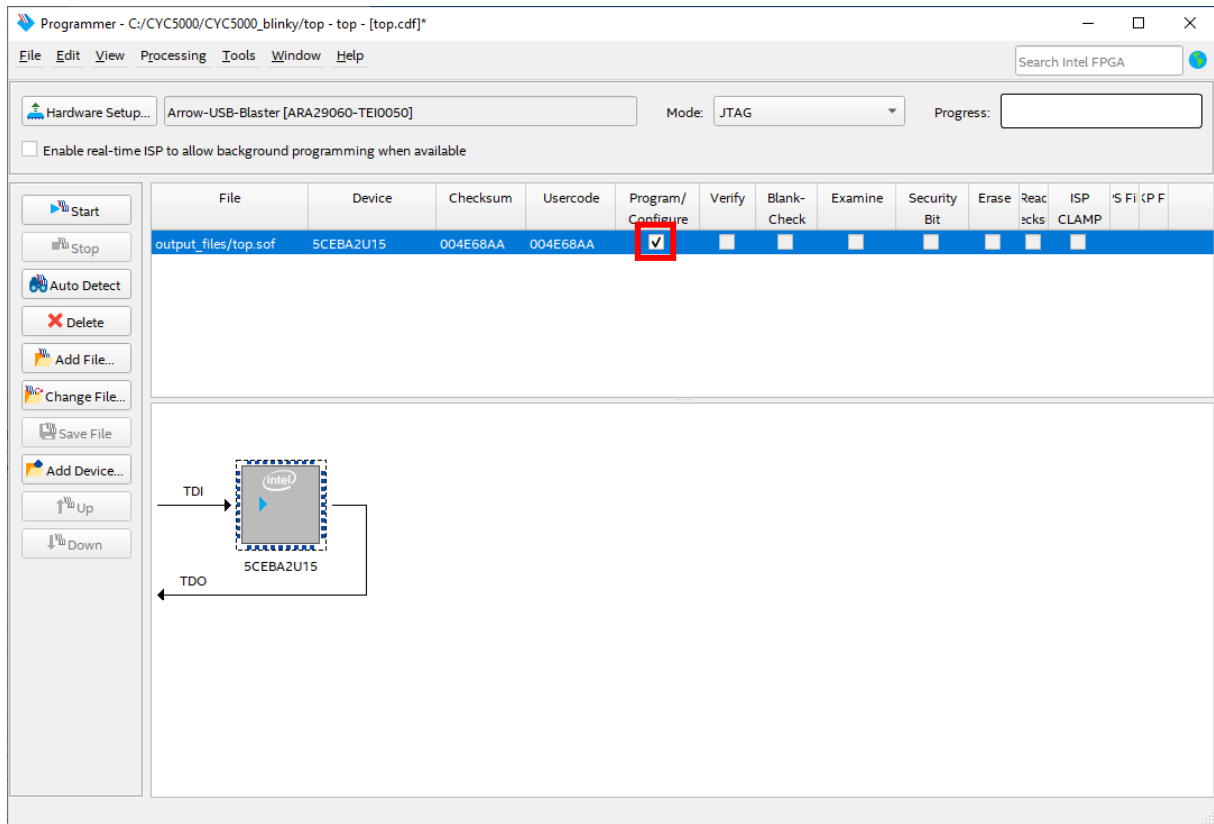


- 6.1.10 To add programming file, double click **<none>** to choose programming file.

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	Reac tacks	ISP CLAMP	S Fi (P F
<none>	5CEBA2	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.



- 6.1.13 Click Start to program the CYC5000. When the configuration is complete, the Progress bar should reach 100% (Successful).

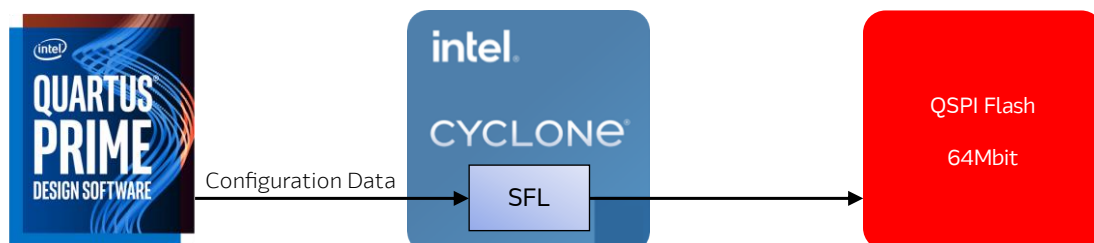
Progress: 100% (Successful)

The design is now programmed to the FPGA.

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

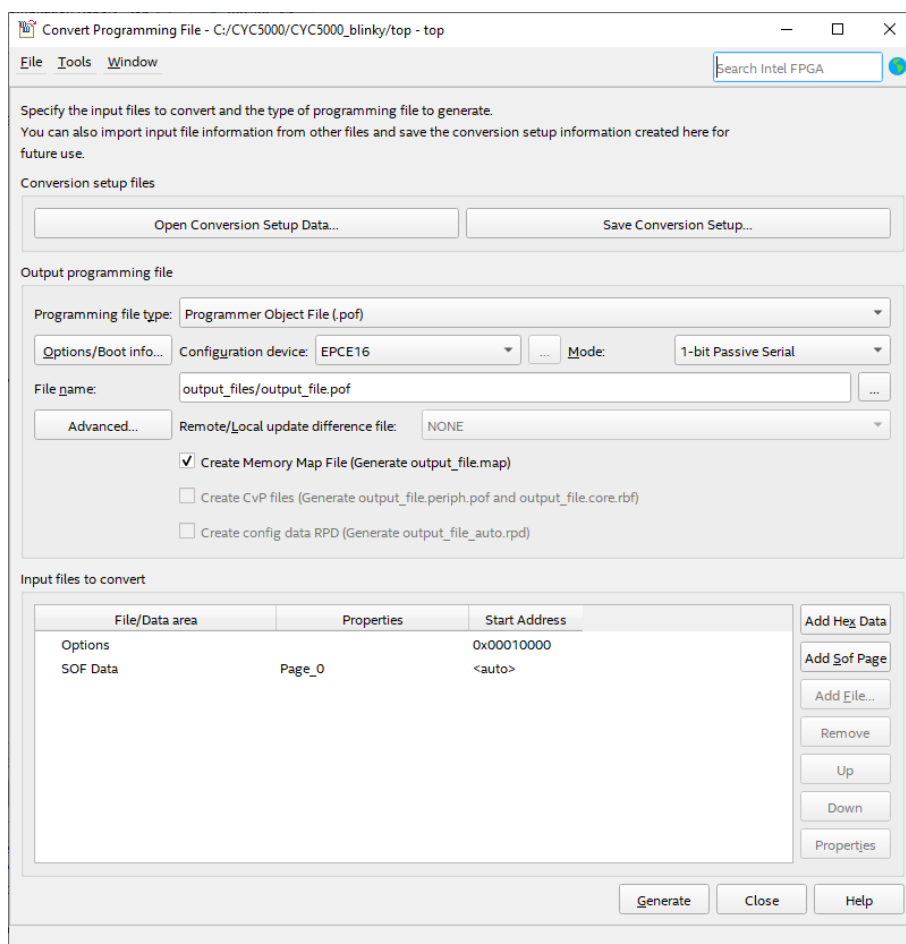
6.2 QSPI flash memory programming

The configuration data to be written to QSPI Flash will be part of the JTAG indirect configuration file (.jic). This configuration data is automatically loaded from the serial configuration flash into the Cyclone V 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...



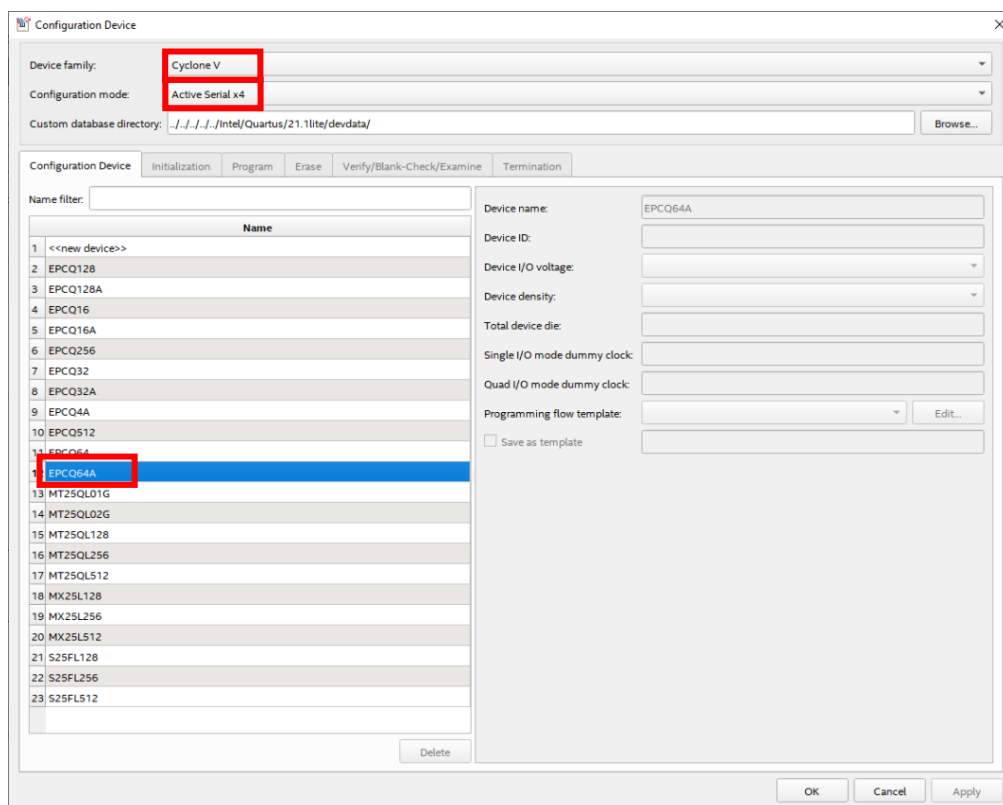
6.2.1.2 Set the programming file type to **JTAG Indirect Configuration File (.jic)**.

6.2.1.3 Click on the  button for configuration device.

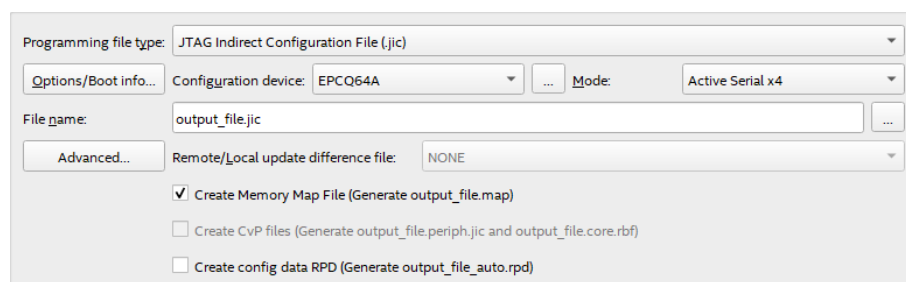
6.2.1.4 Select **Cyclone V** for the Device family, **Active Serial x4** for Configuration mode, and choose **EPCQ64A** from the Configuration device tab.

Although the CYC5000 board has a Winbond QSPI flash memory, the use and control of this memory are completely identical to Intel EPCQ-A. Accordingly, instead of adding a new Configuration device to Quartus, simply selecting EPCQ64A is sufficient.

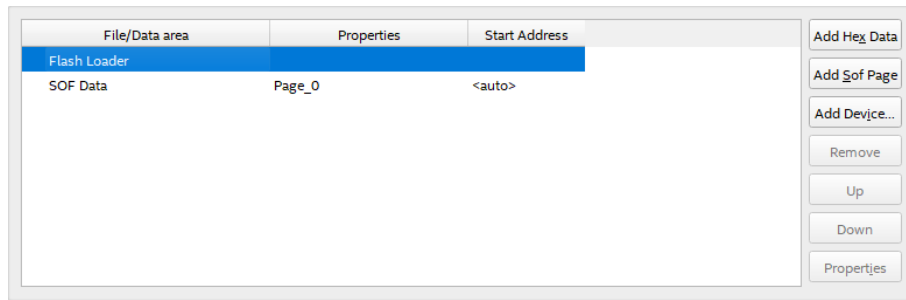
The Active Serial (AS) configuration scheme is supported in the 1-bit data width while AS x4 supports a 4-bit of data width. The Cyclone V FPGA and the QSPI flash memory on the CYC5000 board support both configuration schemes, but AS x4 allows for much faster configuration, which can significantly reduce the start-up time.



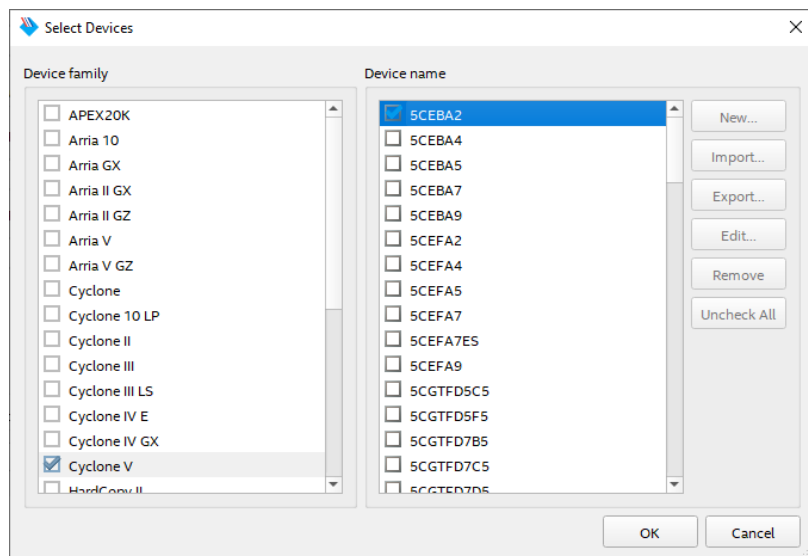
6.2.1.5 Click "OK". Now the output programming file settings should look like this:



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



6.2.1.7 On the new window select **Cyclone V** as Device family and **5CEBA2** as Device name.

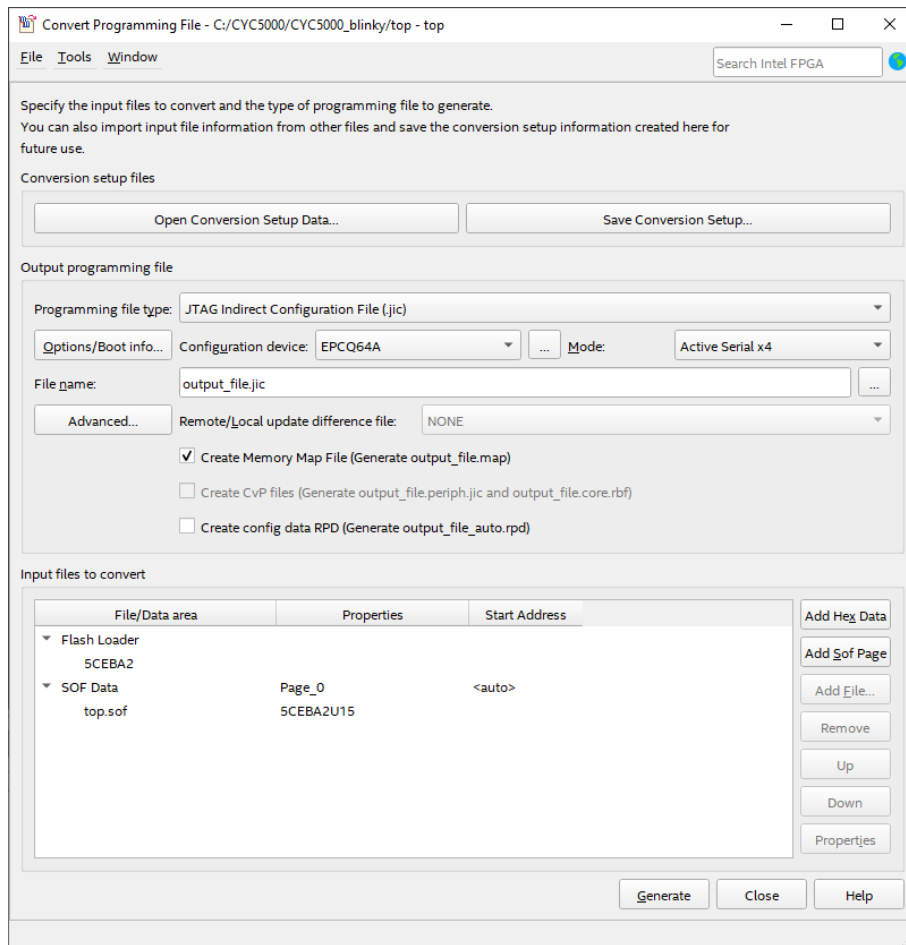


6.2.1.8 Click **“OK”** to add device to Flash Loader.

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

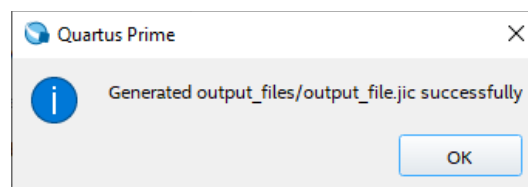
6.2.1.10 Go to **<project_directory>/output_files/** and open **top.sof**.

6.2.1.11 Make sure that your settings are same as the picture below and if everything is correct.



6.2.1.12 Click **“Generate”**.

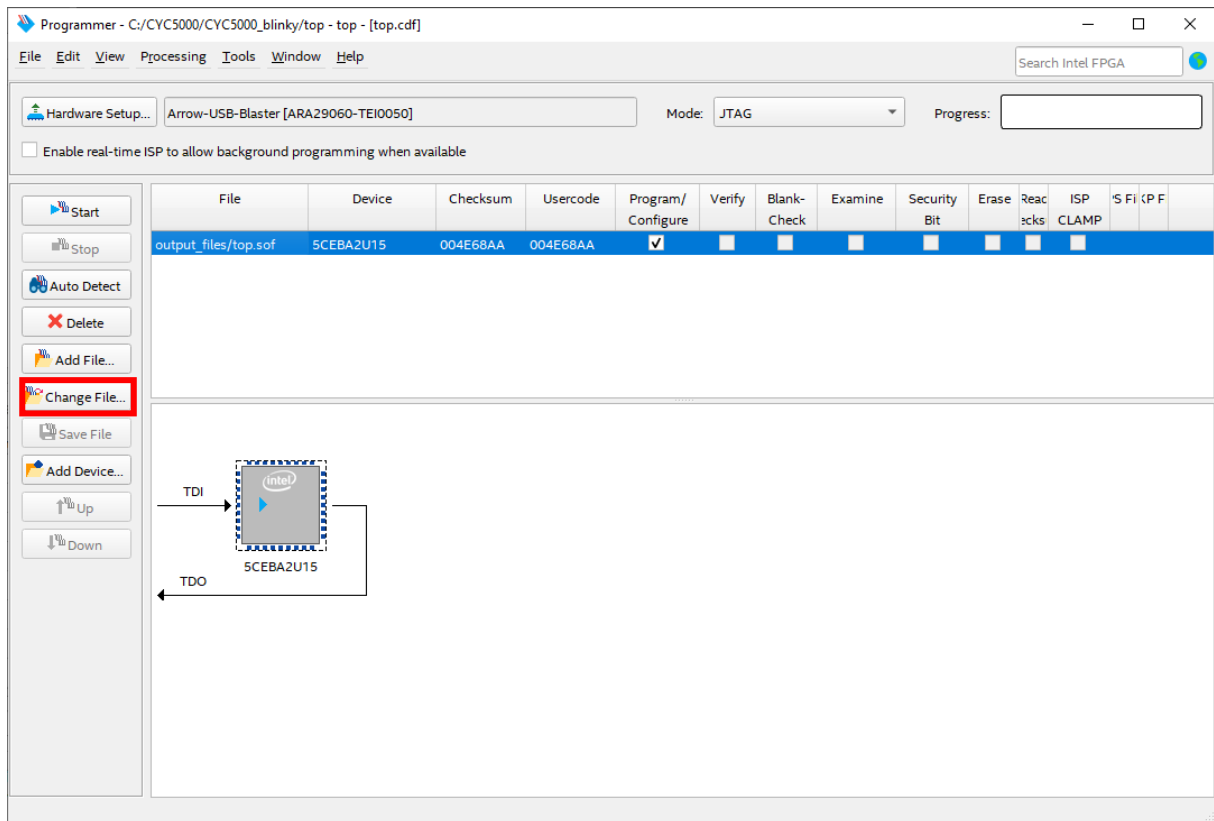
6.2.1.13 Click **“OK”** on the successful file generation notification and **close** Convert Programming File window.



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.

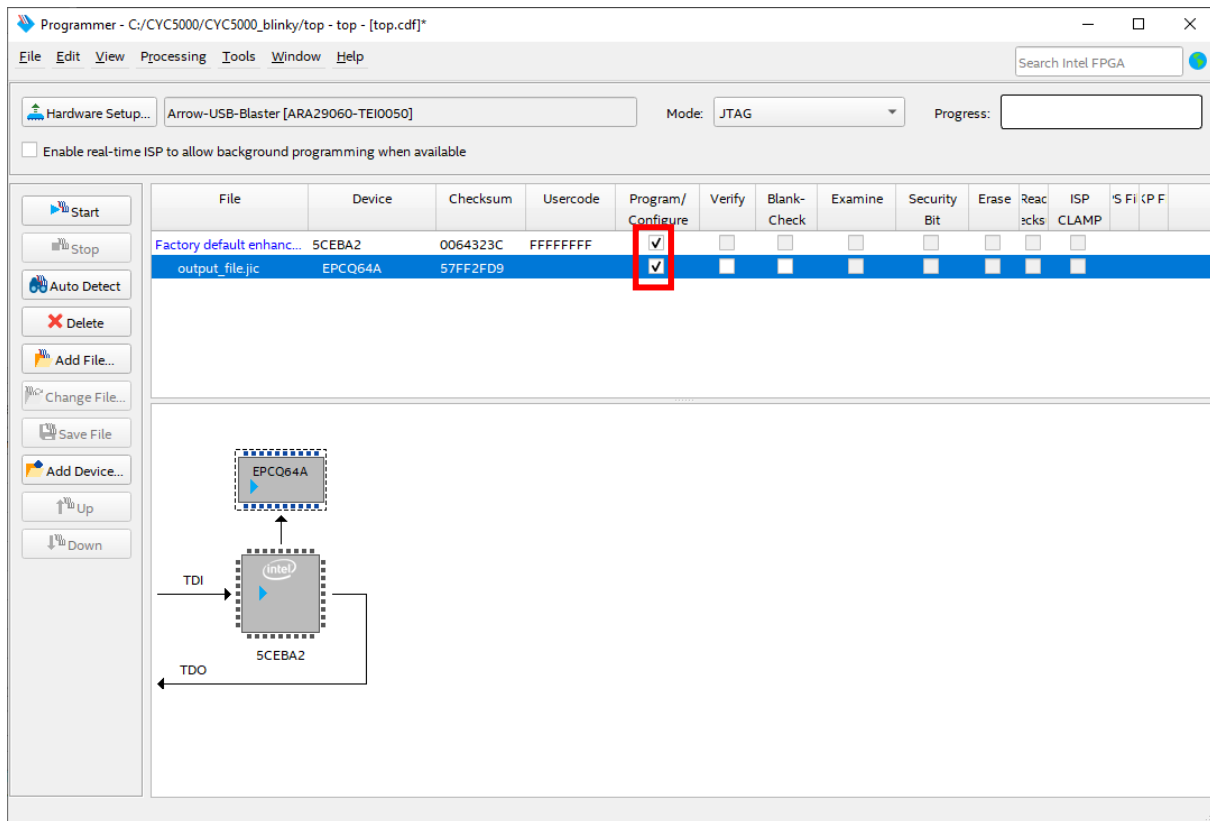


6.2.2.3 Go to <project_directory>/ and open output_file.jic.

If you do not find the programming file, it may be that the default save location is <project_directory>/output_files/ so check that folder as well.

When you add the .jic file, the Programmer will automatically update the JTAG chain and put the configuration flash memory.

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



6.2.2.5 Click **Start** to configure the configuration memory. The programming could take a while.

6.2.2.6 When the programming is finished, the CYC5000 should be able to keep its configuration data even after powered off.

At this point our program is stored in the QSPI flash memory, but the Cyclone V 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 S1 RESET button which will reset the FPGA and automatically loads the configuration from QSPI.

6.3 Testing the Design

Does not matter which way the CYC5000 board 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.

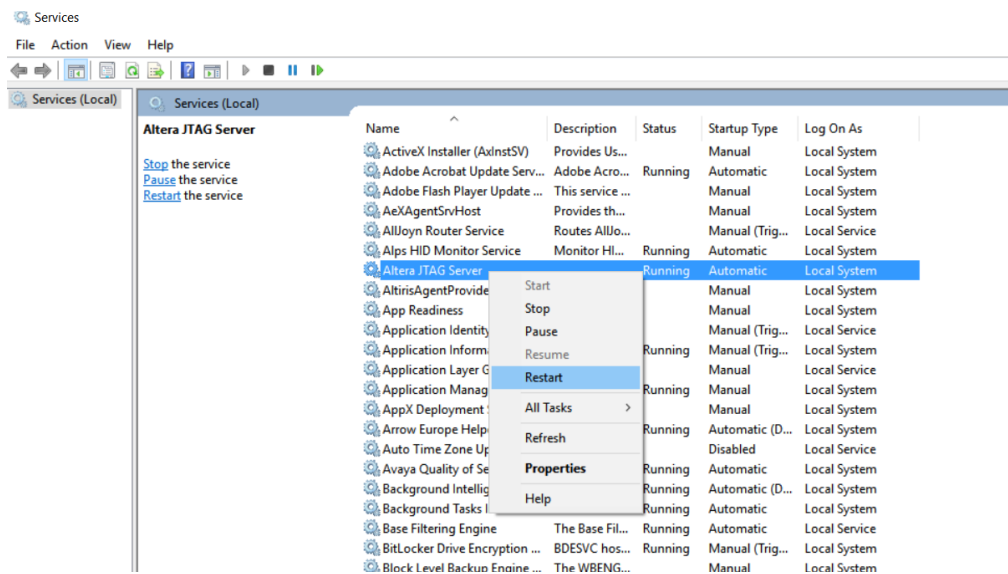
Push and hold the S2 USER_BTN to see that the LEDs will now toggle in a very fast counting sequence. USER_BTN is on the side of the LEDs.

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.





Chapter 8 - Appendix

8.1 Revision History

Version	Change Log	Date of Change
V1.0	Initial Version	29/03/2023



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 its 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.