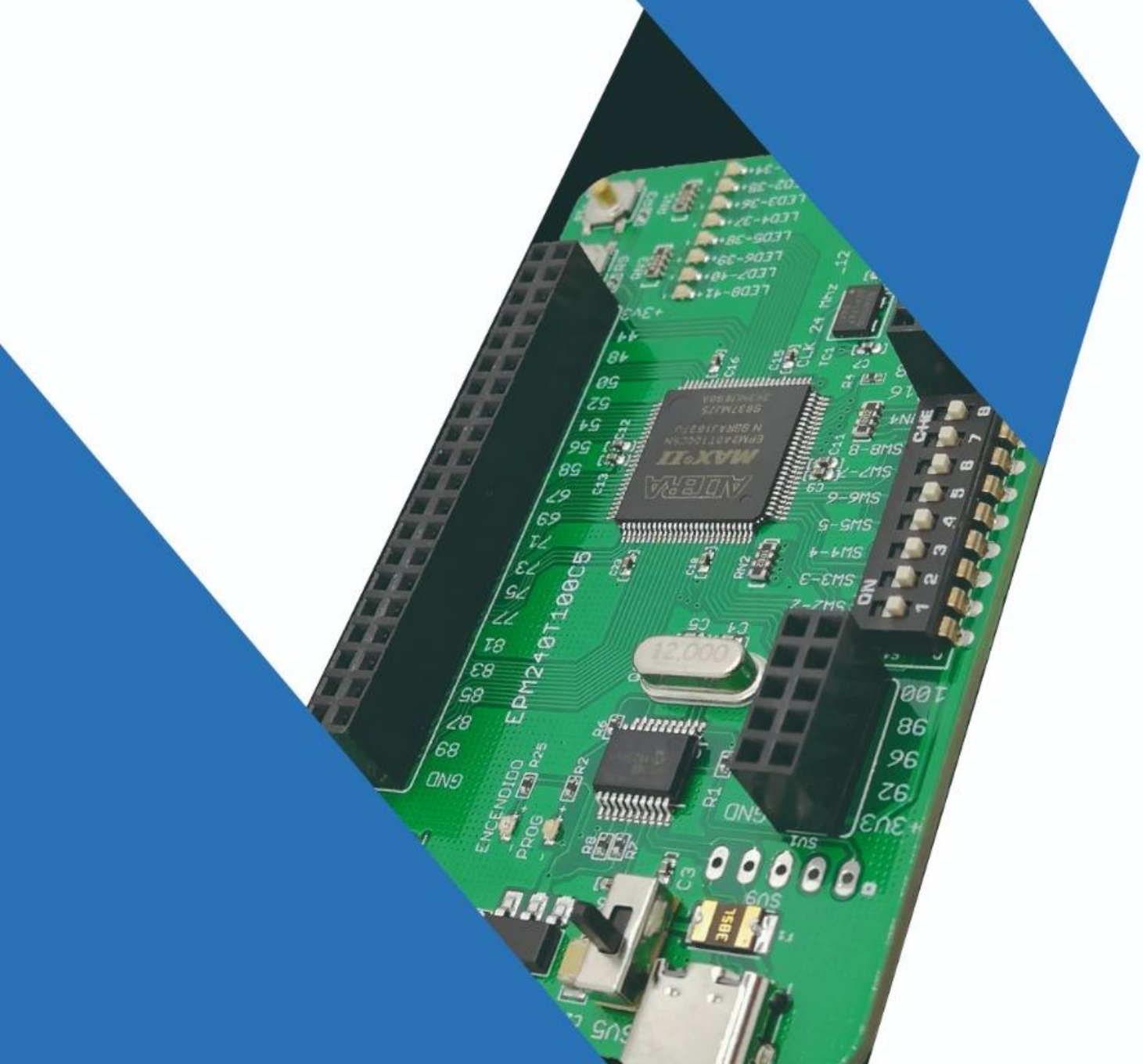
NIBBLUX

DIONE

Especificaciones Tecnicas



DIONE

CONTENIDO

1 Descripción

La tarjeta de desarrollo DIONE consta de un circuito integrado CPLD EPM240T100C5 de Altera, este dispositivo tiene una capacidad de 240 elementos lógicos (Consultar "MAXII DEVICE HANDBOOK" para más información), los cuales pueden ser programados en lenguaje VHDL mediante el entorno Quartus 18.1. DIONE cuenta con un oscilador que genera una señal de reloj a 24MHz conectado en el pin 12. Dione cuenta con un programador integrado con el cual es posible cargar el archivo pof directamente desde el entorno de Quartus 18.1 sin la necesidad de usar programadores externos. DIONE se diseñó en con el objetivo de simplificar el aprendizaje en áreas como electrónica digital, instrumentación y control, sistemas embebidos, mecatrónica, entre otras, ya que este tipo de tecnología se basa en lenguajes de descripción de hardware (HDL por sus siglas en inglés), que consiste crear circuitos lógicos y sus conexiones mediante códigos en una computadora optimizando así el tiempo de construcción de los prototipos al evitar el cableado y conexión física de los circuitos.

Figura 1. Tarjeta DIONE

En su cara superior DIONE cuenta con letreros que indican los pines a los que están conectados los periféricos, estos letreros son muy útiles a la hora de ubicar las entradas y salidas, ver figura 1.

- CPLD MAX II EPM240T100C5, programable en lenguaje VHDL, entorno Quartus 18.1.
- Interruptor de encendido - apagado.
- Fusible térmico, protege a la fuente de voltaje del USB ante un cortocircuito.
- Salida de voltaje de +3.3V y +5V para alimentación de módulos o circuitos externos.
- Programador integrado en la tarjeta mediante conector USB tipo C.
- 52 pines de entrada/salida libres.
- Oscilador MEMS 24MHz para señales de reloj.
- 8 interruptores conectados como entradas.
- 8 leds color azul conectados como salidas.
- 2 botones (push botton) conectados como entradas.

A continuación, se muestra una tabla con los parámetros del dispositivo EPM240T100C5, toda esta información es muy útil a la hora de diseñar circuitos externos,

Parámetro	Símbolo	Condiciones	Mínimo	Típico	Máximo	Unidad
Voltaje de alimentación	Vcc	-		3.3	3.6	V
Voltaje de entrada en alto	V_{inH}	-	1.7	3.3	4	V
Voltaje de entrada en Bajo	V_{inL}		-0.5	0	0.8	V
Corriente de entrada del pin (corriente de fuga)	I_{inH}	Vi=3.3V	-10	-	10	µA
Voltaje de salida en alto	V_{oH}	-	2.4	3.3	-	V
Voltaje de salida en bajo	V_{oL}	-	-	0	0.45	V
Corriente de salida máxima en alto (1)	I_{omax}	Vo=3.3V	-	-	16	mA
Corriente de salida recomendada	I_{orec}	-	-	-	10	mA
Corriente de pull up recomendada	$I_{pull\ up}$	-	-	-	300	µA
Resistor de pull up recomendado	$R_{pull\ up}$	Vcc=3.3V	5		25	kΩ
Temperatura de operación recomendada.	T_j	-	0	-	85	°C

(1) No se debe exceder el límite de 16mA en la corriente de salida, si esto sucede el dispositivo se dañará.

3 Alimentación

La tarjeta DIONE se alimenta a través del conector USB tipo C, el voltaje en el conector es de +5V, es regulado a +3.3V para la alimentación del CPLD y las fuentes de cada banco de entrada/salida. DIONE cuenta con dos salidas de fuete de alimentación, una salida es de +3.3V y la otra salida es de +5V, estas fuentes de alimentación cuentan con un fisible térmico de protección anti corto circuito, la corriente máxima que pueden proporcionar las fuentes de alimentación juntas es de 700mA.

Es posible alimentar la tarjeta con una fuente externa de +5V en caso de no querer usar el cable USB C, para esto conecte las terminales de la nueva fuente a las terminales +5V0 y GND. No usar voltajes diferentes a +5V o el dispositivo se puede dañar.

El interruptor que se encuentra junto al conector USB funciona como interruptor de encendido y apagado. Nunca conectar al mismo tiempo el cable USB C y una fuente externa en las terminales +5V0 y GND.

Nota: Cualquier duda con respecto a la tarjeta DIONE favor de comunicarte con nuestro equipo de soporte técnico, Facebook/Nibblux, WhatsApp 5583725122 o por correo electrónico ventas@nibblux.com.

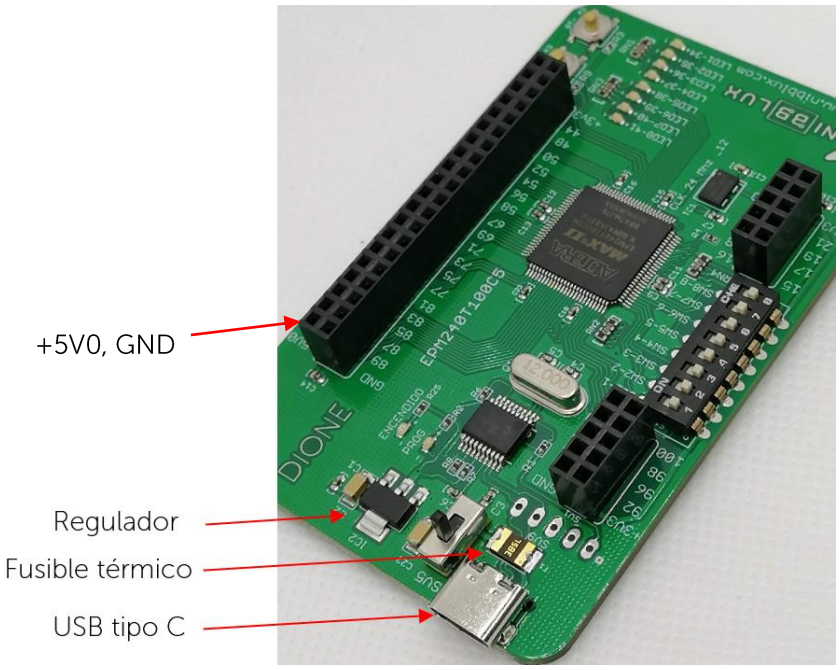
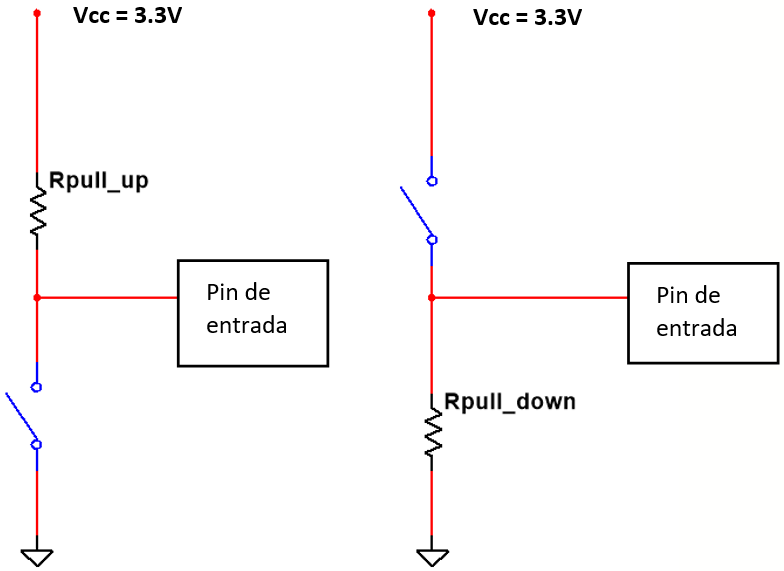


Figura 7. Etapa de alimentación.

Para el correcto funcionamiento de los pines de entrada y salida se debe tomar en cuenta lo siguiente.

Cuando se configuran los pines como entradas estos presentan una alta impedancia, lo que significa que la corriente que circulará por el pin será muy pequeña, 10 µA para ser exactos, este valor se puede despreciar. Todos los pines de entrada deben tener un resistor de pull up o pull down conectado de la siguiente forma.



(Figura 2 a). Conexión de resistores de pull up y pull down.

La corriente de pull up recomendada por el Altera es de 300µA, con base en esto se calcula el resistor de pull up o pull down usando la ley de ohm como se muestra a continuación.

$$R_{pull\ up} = \frac{Vcc}{I_{pull\ up}}$$

$$R_{pull\ up} = \frac{3.3V}{300\mu A} = 11.3k\Omega$$

Se selecciona un valor comercial de 10kΩ.

Para las salidas la situación es diferente, en los pines de salida sí circula corriente eléctrica, LA CORRIENTE MÁXIMA QUE PUEDE DAR UN PIN DE SALIDA ES DE 16mA, por lo que el resistor que se coloque en la salida debe controlar el flujo de corriente por debajo de este valor. Si en la salida se usa un resistor de pull up, éste se calcula de la misma forma que para las entradas, en cambio, si la salida requiere de una corriente mayor, por ejemplo en el caso de un optoacoplador, hay que tener cuidado de que la corriente de salida (corriente del led del optoacoplador) no sobrepase los 16mA. Se recomienda una corriente de salida de operación ($I_{sal\ rec}$) de 10mA ya que 16mA es el límite máximo y si se trabaja el pin a esa corriente el dispositivo se dañará en poco tiempo.

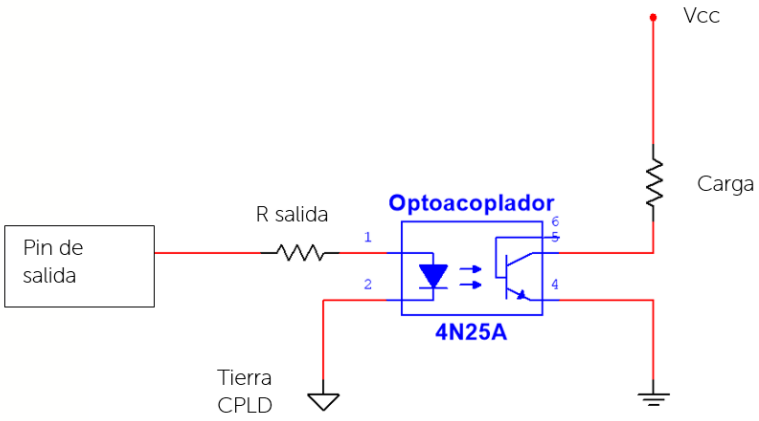


Figura 2 b). Conexión de resistores de salida para el uso de un optoacoplador.

El voltaje V_{led} se obtiene de la hoja de especificaciones del optoacoplador. El valor mínimo de resistencia recomendado en el caso de usar un optoacoplador se obtiene de la siguiente expresión:

$$R_{salida} = \frac{V_{sal\ pin} - V_{led}}{I_{sal\ rec}}$$

$$R_{salida} = \frac{3.3V - 1.15V}{10\text{mA}} = 215\Omega$$

DIONE cuenta con periféricos de entrada y salida integrados a la tarjeta. Cada uno de estos periféricos tiene conexiones permanentes con pines de la CPLD. En la cara superior de la tarjeta DIONE podemos encontrar información que nos indica a qué pin del dispositivo EPM240T100C5 está conectado cada periférico, ver figura 3. Esta información es muy útil a la hora de hacer las conexiones de los pines de entrada y salida con circuitos externos.

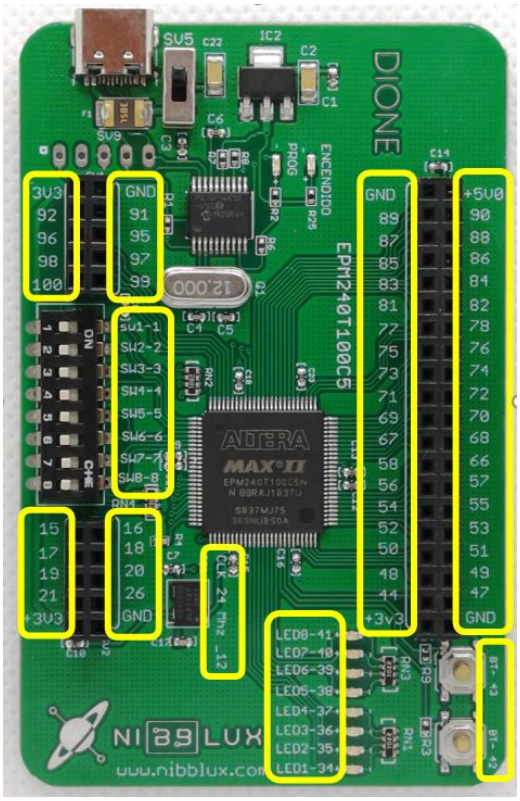


Figura 3. La información encerrada los rectángulos amarillos indica el número de pin del dispositivo EPM240T100C5 al que está asignada cada entrada y salida de la tarjeta. Este número se tiene que ingresar en el software Quaruts para poder usar dichos pines.

La tarjeta DIONE cuenta con 52 pines de entrada – salida de propósito general, estos pines se encuentran en los conectores "header" negros de la tarjeta. Cada bloque de pines cuenta con una fuente de alimentación de +3.3V y GND, esta fuente se puede usar para alimentar circuitos externos que se utilicen como entradas, recuerde siempre utilizar resistores de pull up o pull down en los pines de entrada y salida. Revise la sección anterior para calcular el valor de dichos resistires.

La tarjeta cuenta además con una fuente de alimentación de +5V que se puede usar para circuitos externos.

Nota importante: Nunca conectar la fuente de +5V a las entradas o salidas de la tarjeta, recordemos que funciona a +3.3V, el voltaje máximo de las entradas y salidas es de +3.6V y de la fuente es +4V si se supera este valor el dispositivo se destruirá.

La corriente máxima que pueden dar las fuentes de alimentación es de 750mA se debe tomar en cuenta este valor para el diseño de sus proyectos.

Interruptores de entrada

DIONE tiene 8 interruptores de dos posiciones que pueden ser usados como entradas, conectado en configuración pull down. Estos interruptores están conectados a los pines 1, 2, 3, 4, 5, 6, 7, 8 del dispositivo EPM240T100C5. Estos pines no pueden ser usados como salidas.

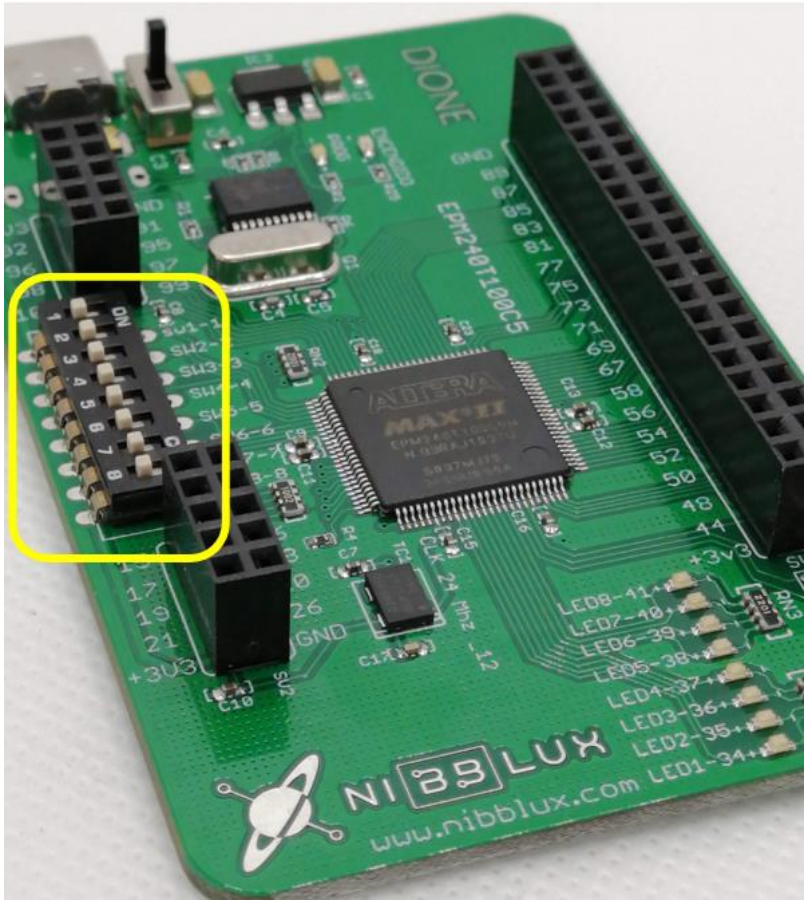


Figura 4. Interruptores de entrada.

DIONE cuenta con 8 leds color azul conectados a los pines 34, 35, 36, 37, 38, 39, 40, 41, del dispositivo EPM240T100C5, estos leds pueden ser usados como indicadores de estados lógicos. Estos pines no pueden ser usados como entradas.

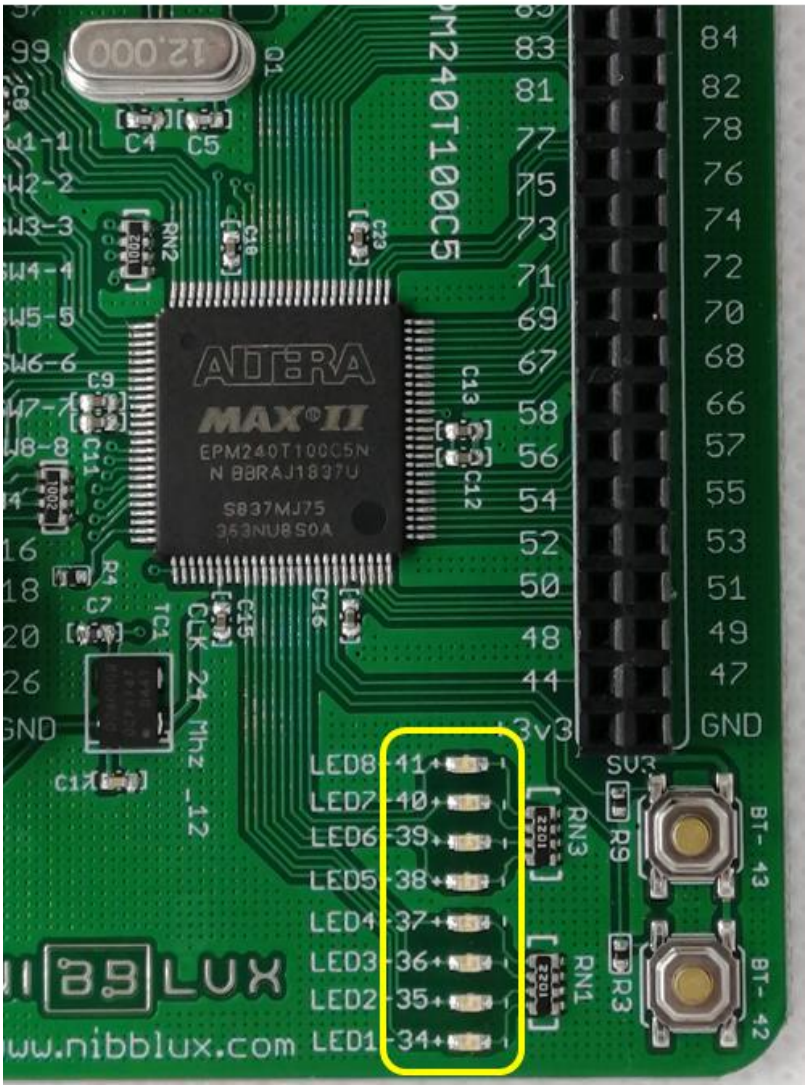


Figura 5. Salidas de led.

Oscilador 24Mh

Oscilador de frecuencia fija, tecnología MEMS, 3.3V frecuencia de oscilación 24Mhz, forma de onda cuadrada. Este oscilador puede ser usado como señal de reloj, está conectado al pin 12 del dispositivo EPM240T100C5, Para habilitarlo declare el pin 12 como variable de entrada en software.

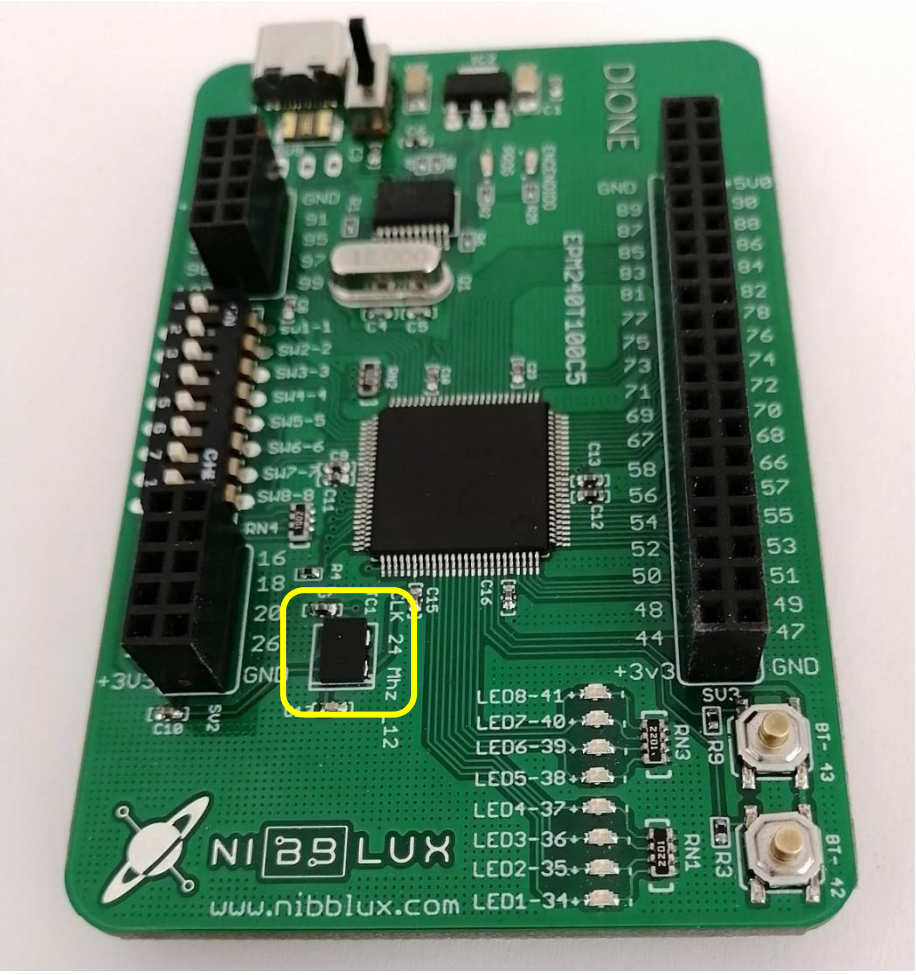


Figura 6. Oscilador 24MHz.

Dione cuenta con dos botones (push bottton) conectados a los pines 42 y 43, cada uno de ellos está conectado mediante un resistor de pull down. Estos pines no pueden ser usados como salidas. Se debe considerar que este tipo de botones generan lo que se conoce como rebotes, que es un cambio de estado lógico a alta frecuencia que se debe al funcionamiento interno del botón, este efecto dura unos cuantos milisegundos y después el estado lógico se estabiliza. Esta característica de los botones se puede eliminar mediante software usando retardos.

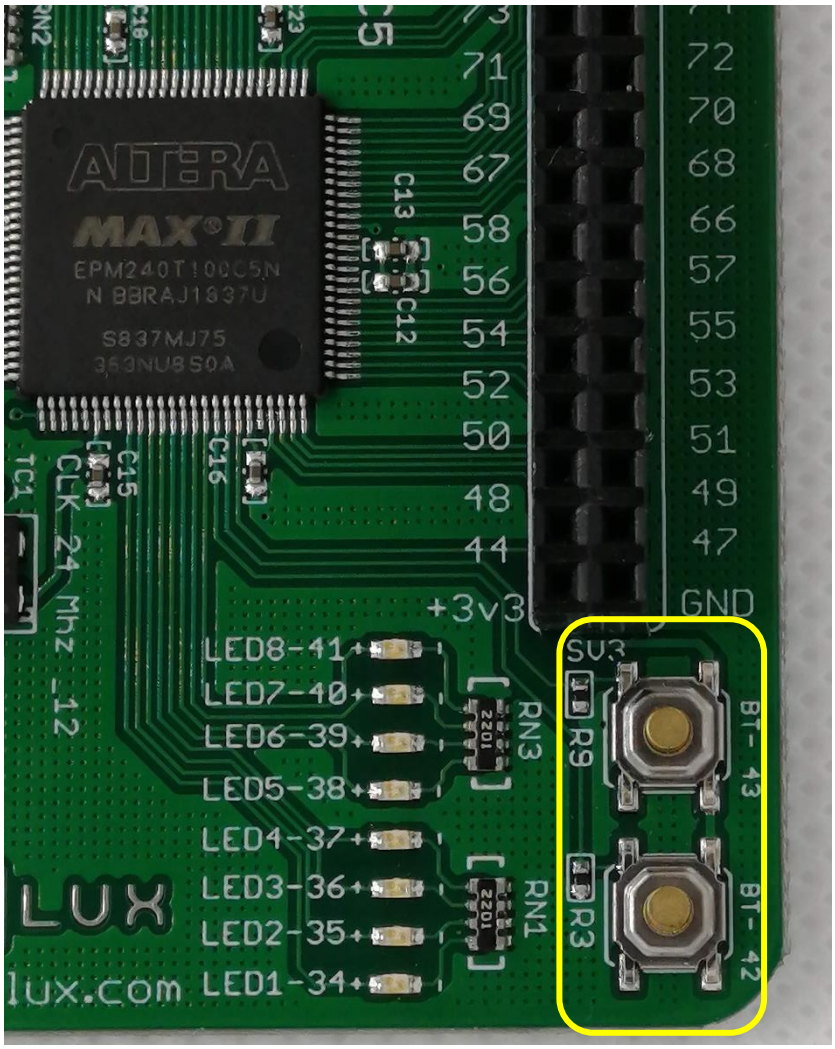


Figura 7. Botones de entrada.

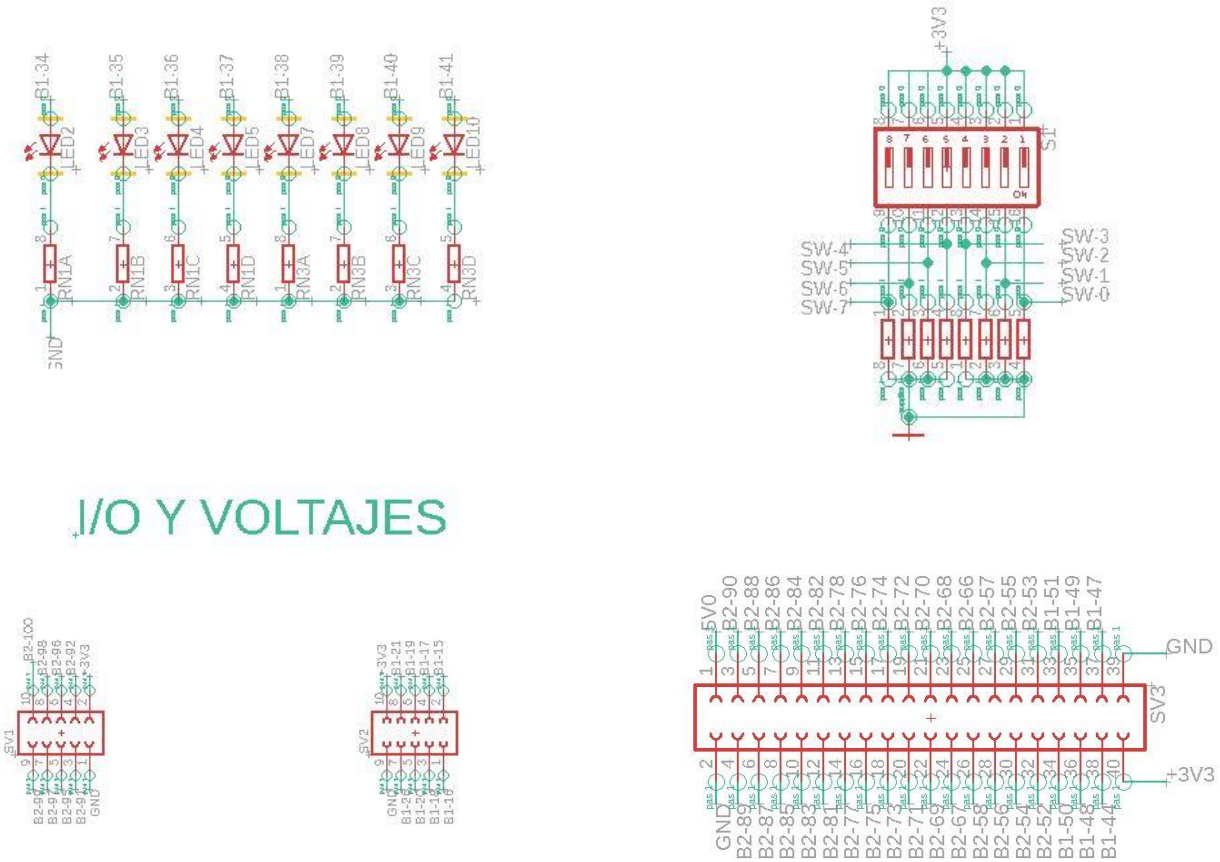


Figura 8. Diagramas esquemáticos de periféricos de entrada y salida.

Dimensiones

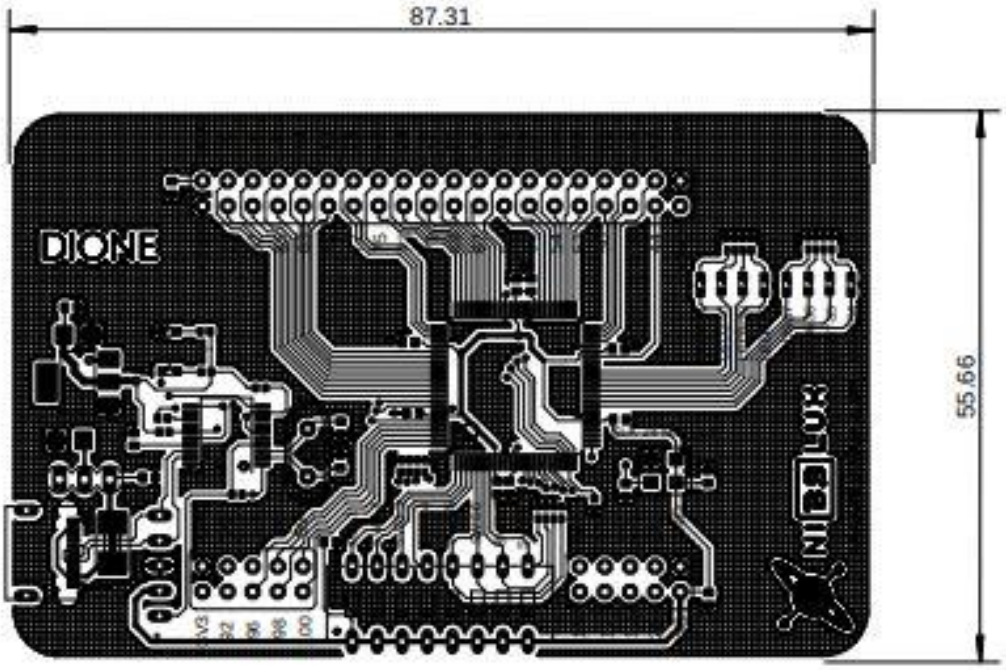


Figura 9. Dimensiones de la tarjeta
DIONE expresadas en milímetros [mm].

Referencias

- MAX II Device Handbook, ALTERA

Historial de Revisión

Fecha y revisión	Revisó	Cambios hechos
22/08/2019	Ing. David Alonso Tapia Abrego.	Primera publicación
04/09/2019	Ing. Ricardo Alberto Villegas Pantoja. Ing. David Alonso Tapia Abrego.	Correcciones en redacción.
23/01/2020	Ing. David Alonso Tapia Abrego.	Se añadió la parte de botones de entrada y se actualizaron las imágenes.
10/09/2020	Ing. David Alonso Tapia Abrego.	Funcionamiento con fuente de alimentación externa.

Elaborado por:

Ing. José Luis Molina Olivera.

Ing. David alonso Tapia Abrego.

Ing. Carlos Edoardo Manzano Fragoso.

NIBBLUX

DIONE CPLD MANUAL DE PRÁCTICAS

INSTALACIÓN Y USO DEL ENTORNO QUARTUS 18.1

VHDL

DIONE CPLD

CONTENIDO

Objetivo

Proporcionar a los lectores una guía práctica de la implementación del lenguaje de descripción de hardware VHDL en dispositivos programables de tipo CPLD y FPGA.

Introducción

La tarjeta DIONE fue diseñada como una herramienta práctica para el aprendizaje e implementación de proyectos con dispositivos CPLD. El circuito integrado es un EPM240T100C5 de la familia MAX II de Altera. DIONE cuenta con los dispositivos necesarios para ejecutar diferentes tipos de ejercicios dentro de la plataforma Quartus 18.1.

Dione dispone de:

- CPLD MAX II EPM240T100C5, programable en lenguaje VHDL, entorno Quartus 18.1.
- Interruptor de encendido - apagado.
- Fusible térmico, protege a la fuente de voltaje del USB ante un cortocircuito.
- Programador integrado en la tarjeta mediante conector USB tipo C.
- Salida de voltaje de +3.3V y +5V para alimentación de módulos o circuitos externos.
- 52 pines de entrada/salida libres.
- Oscilador MEMS 24MHz para señales de reloj.
- 8 interruptores conectados como entradas.
- 8 leds color azul conectados como salidas.

Para más información revisar el documento “DIONE CPLD Especificaciones Técnicas¨, que se encuentra en la página www.nibblux.com.

El software a usar es Quartus 18.1 de Intel, este software es el recomendado por el fabricante. La interfaz de programación lleva por nombre USB-Blaster y está integrada en la tarjeta DIONE, esta interfaz hace uso del protocolo JTAG para crear la comunicación entre la CPLD y Quartus 18.1. cuando esta es programada.

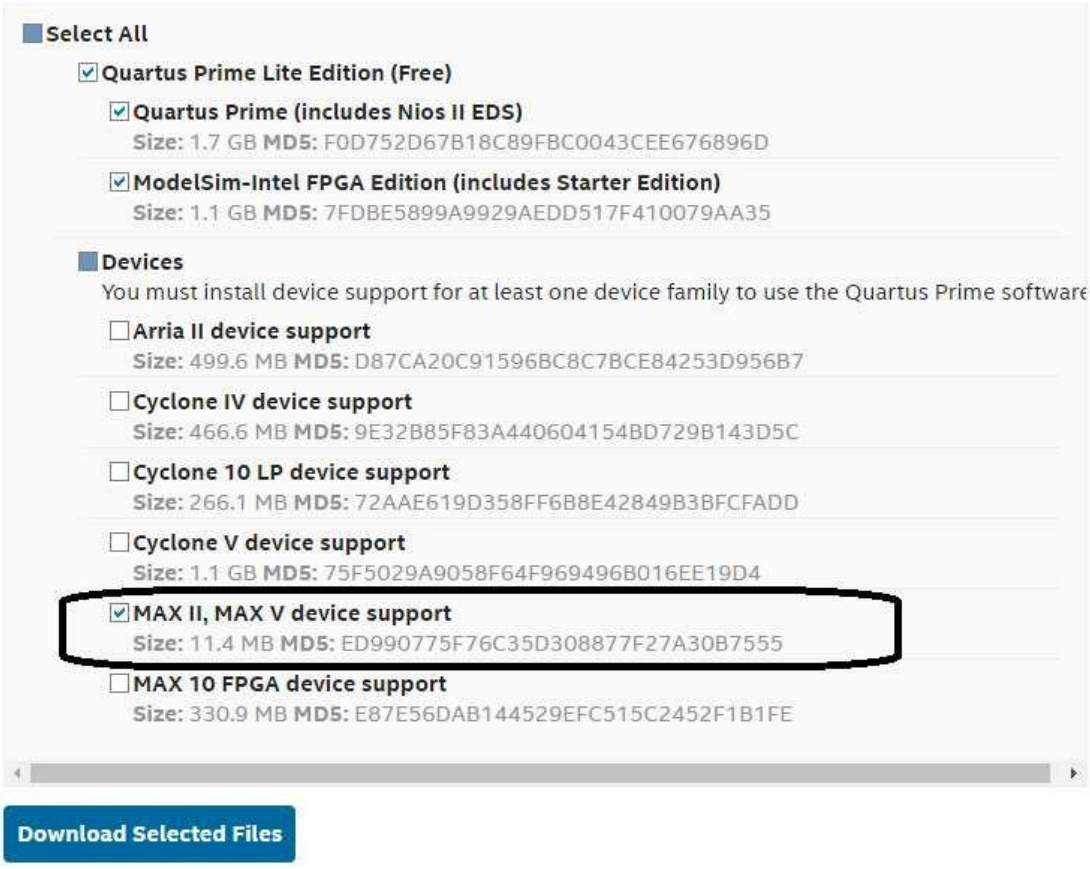
Instalación de QUARTUS 18.1.

Como se mencionó anteriormente, el software a emplear será el Quartus 18.1, (se recomienda la versión más actualizada para evitar problemas de compatibilidad), este software se puede encontrar en el siguiente link: http://fpgasoftware.intel.com/?edition=lite&_ga=2.164226303.1749834903.1556980290-1378022179.1556980290 (es necesario registrarse en la página de intel para poder descargar el software).

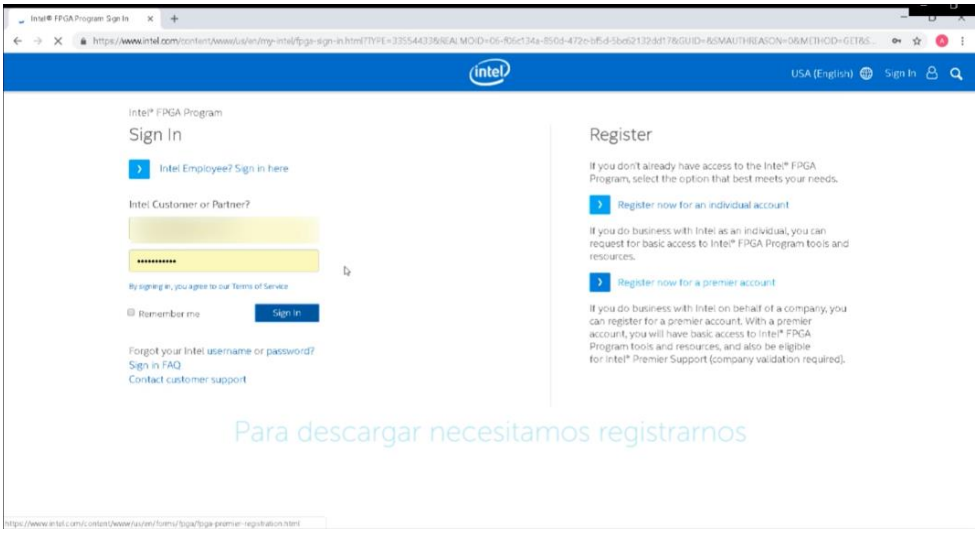
Seleccionar la versión 18.1 y descargar la versión "lite edition", ya que solamente se necesita soporte para la "MAX II".
En la página siguiente aparecen ciertos parámetros para la descarga, ahí se selecciona el sistema operativo del usuario, el software y el método de descarga.

Al desplazarse a la parte inferior se encuentran los siguientes campos, solo se modifica la parte de ¨Devices¨, es necesario marcar la opción "MAX II, MAX V DEVICE SUPPORT", ya que es el paquete de soporte para el integrado que se está usando.

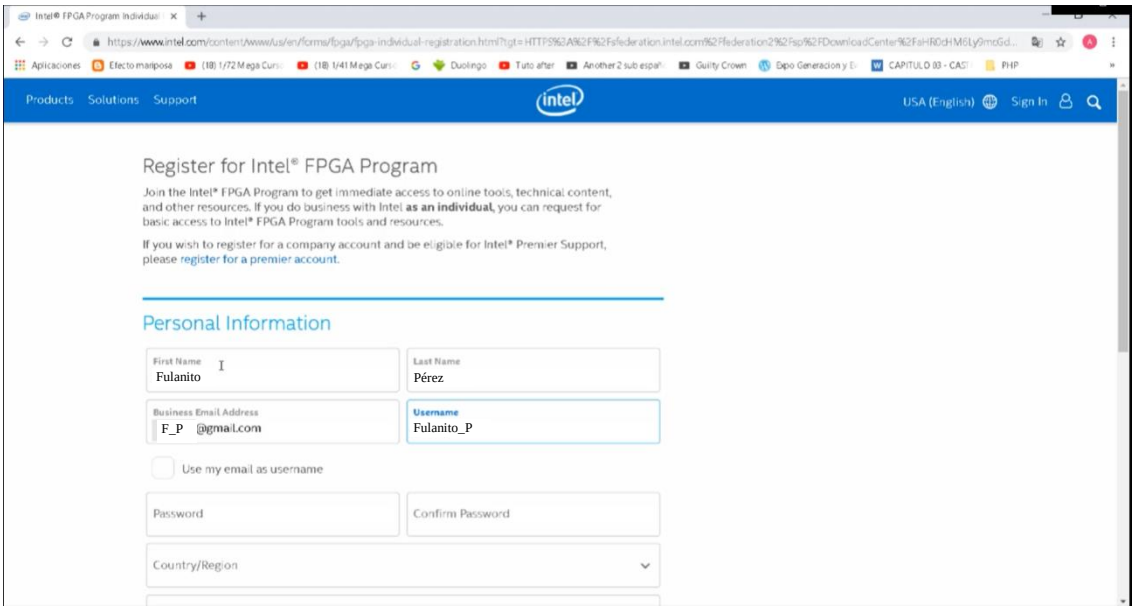
:



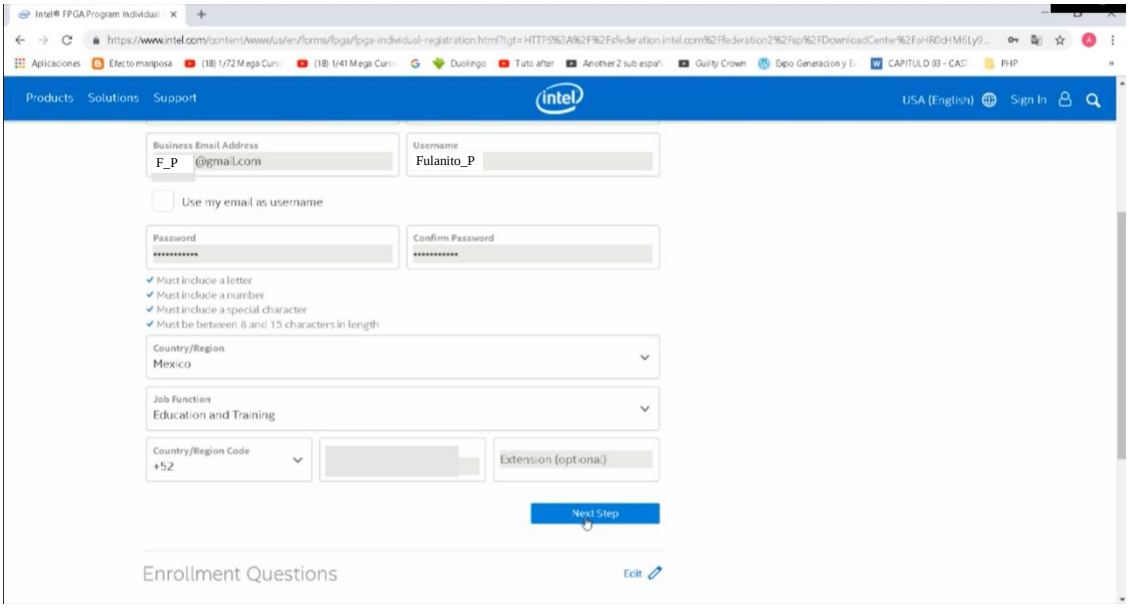
Aparecerá una ventana como la siguiente, en este punto es necesario registrarse para poder descargar el software.

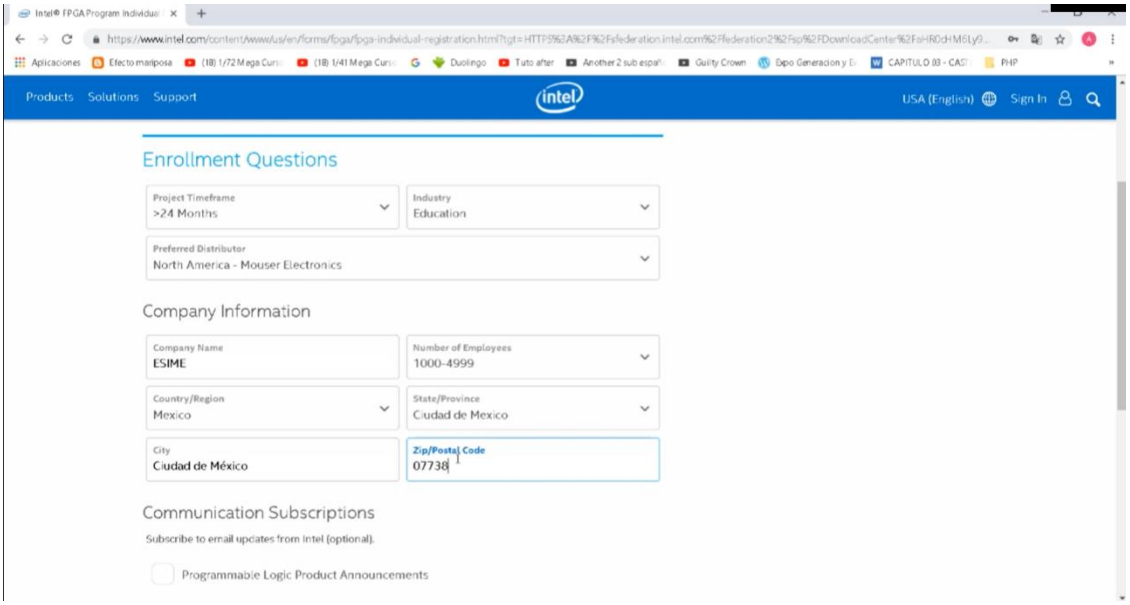


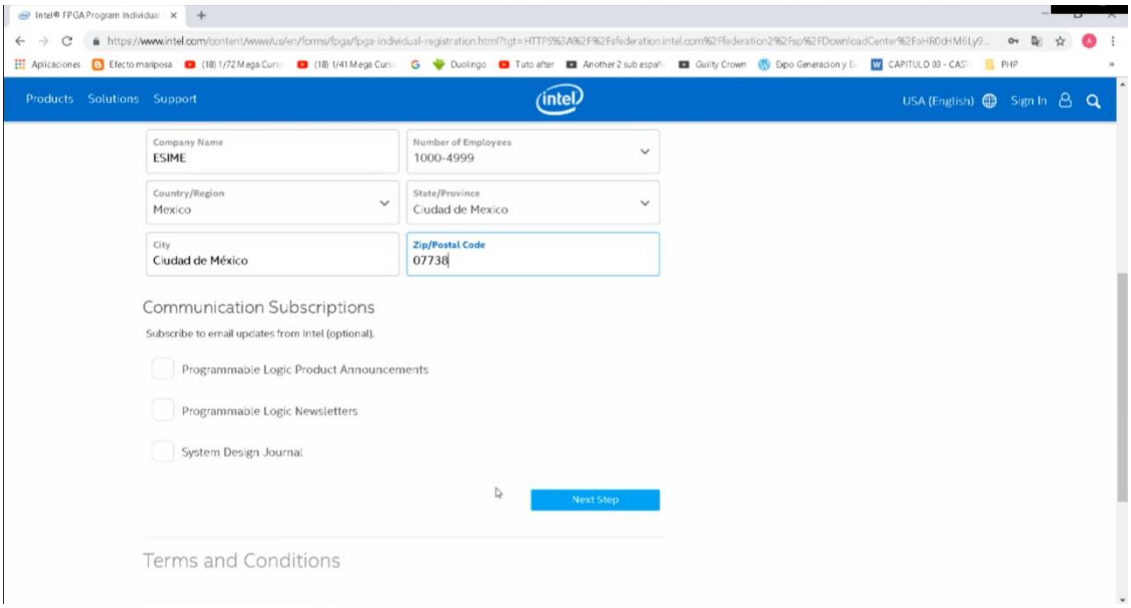
A continuación se muestra un ejemplo de cómo registrarse.



Dar clic en Next Step.

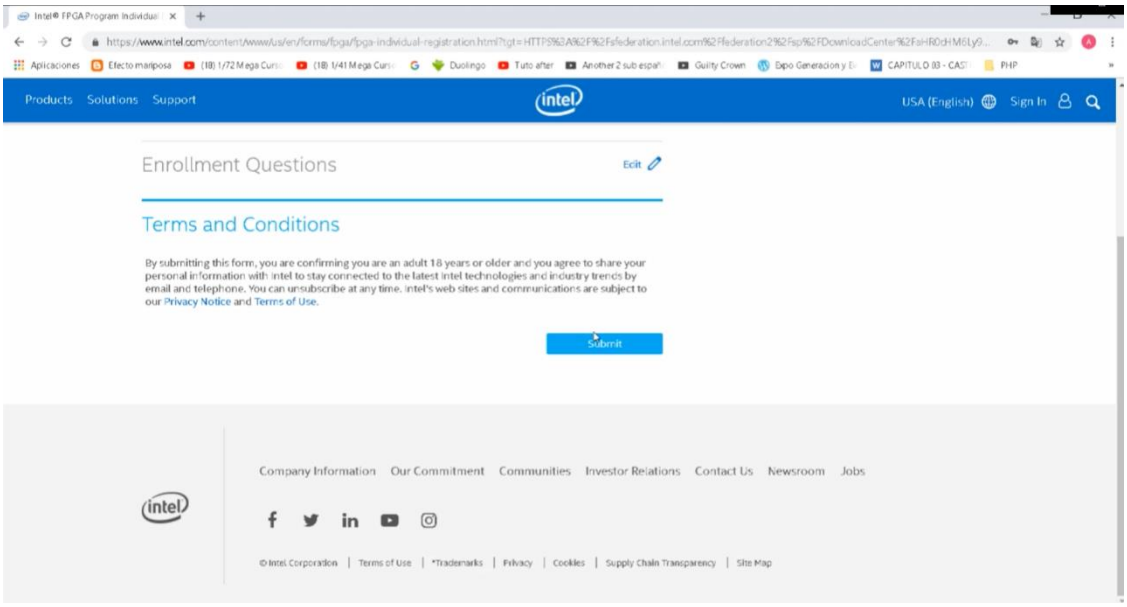




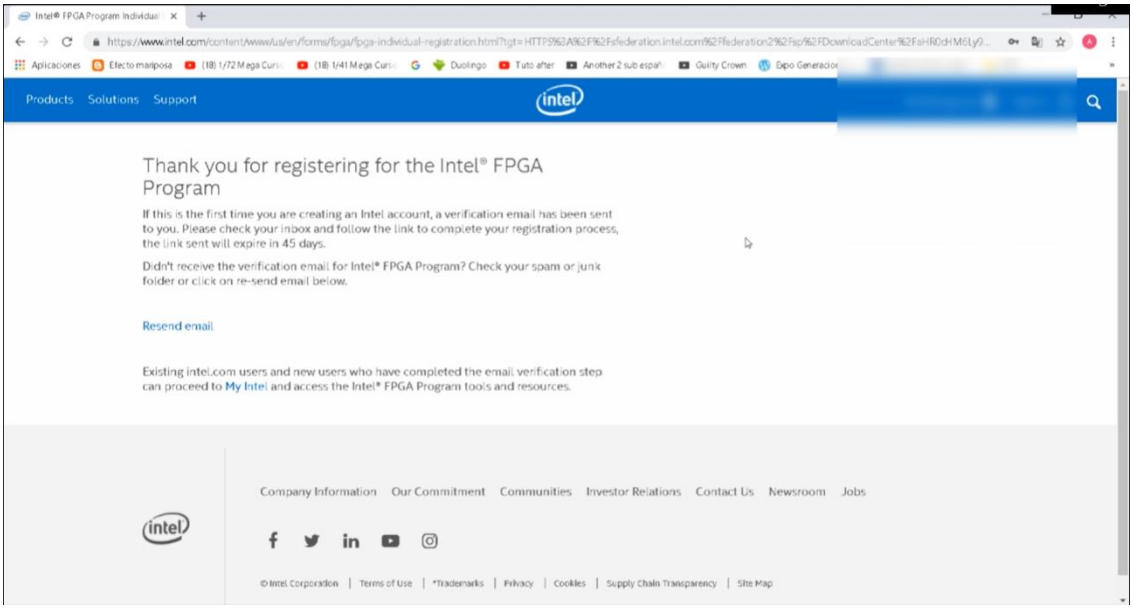


Dar click en Next Step.

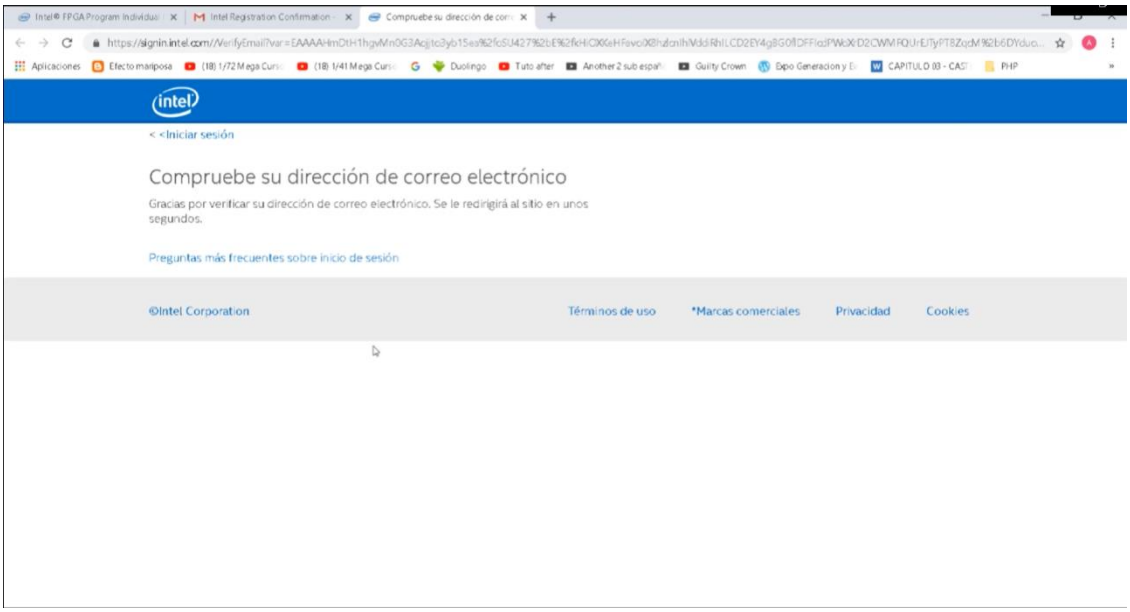
Ahora dar clic en Submit.



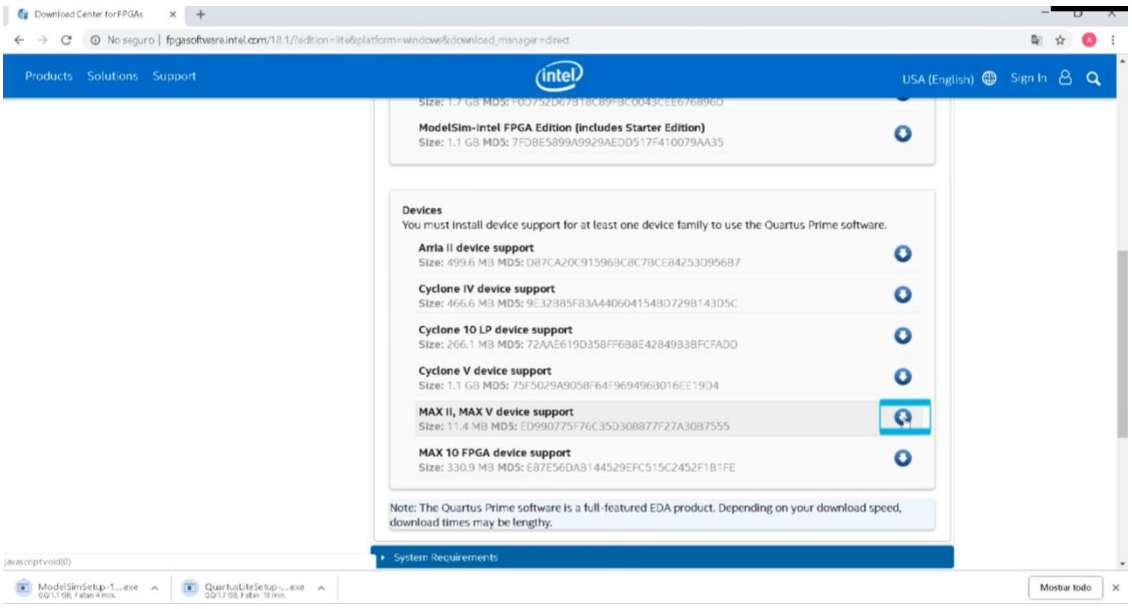
Aparecerá la siguiente ventana que indica que se envió un correo electrónico a la cuenta ingresada anteriormente, es necesario abrir ese correo para verificar la cuenta.



Después de verificar el correo electrónico aparecerá una ventana como esta:



Después de este paso es necesario iniciar sesión en la plataforma de Intel con tu usuario y contraseña, una vez iniciada la sesión es posible descargar Quartus 18.1 lite edition.

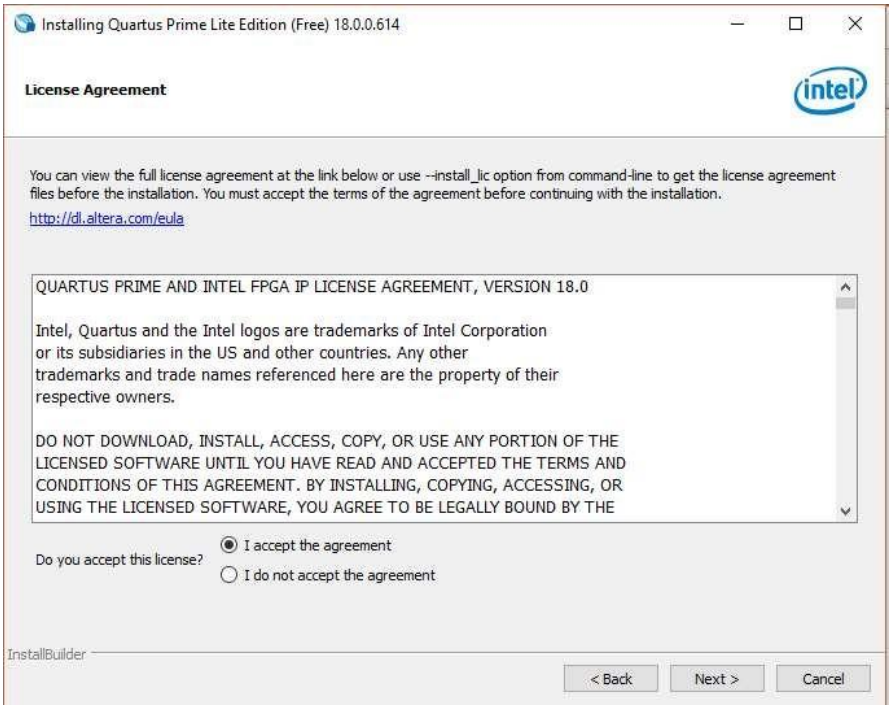


Para finalizar dar clic en "DOWNLOAD SELECTED FILES" y comenzará la descarga.

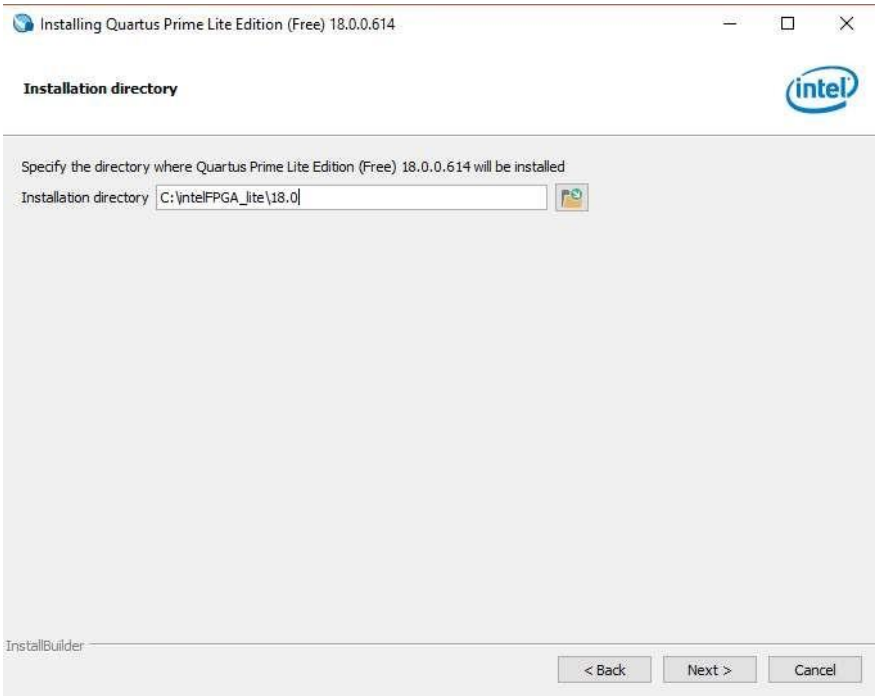
Una vez que se tienen los archivos descargados, aparece un paquete como este:

Dar clic a "quartuslitesetup-18.0.0.614" para comenzar la instalación (es necesario ejecutarlo como administrador para evitar conflictos en la instalación). Aparecerá una ventana como esta:

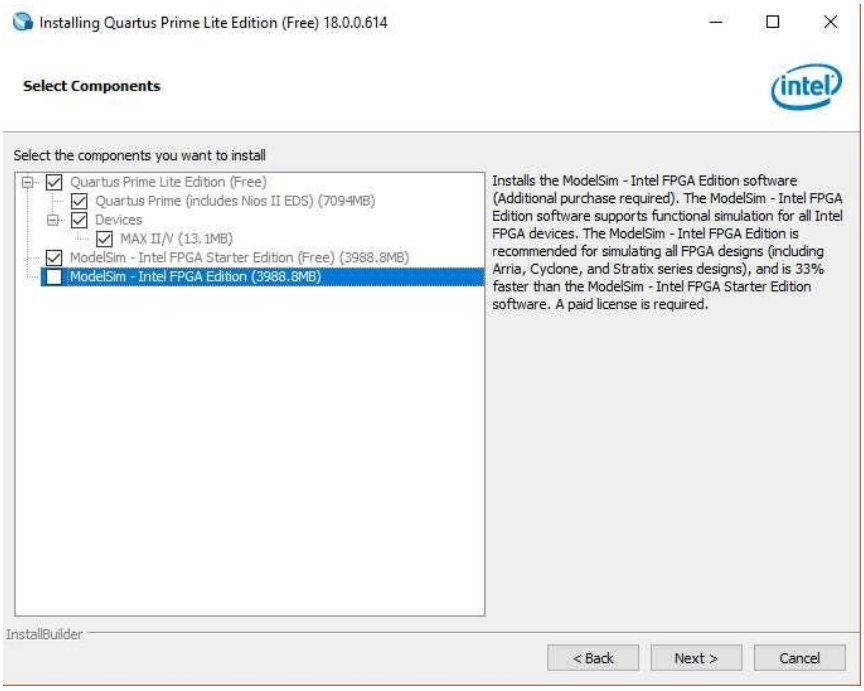
Dar clic en “Next”.



Seleccionar el campo “I accept the agreement” para aceptar las condiciones.

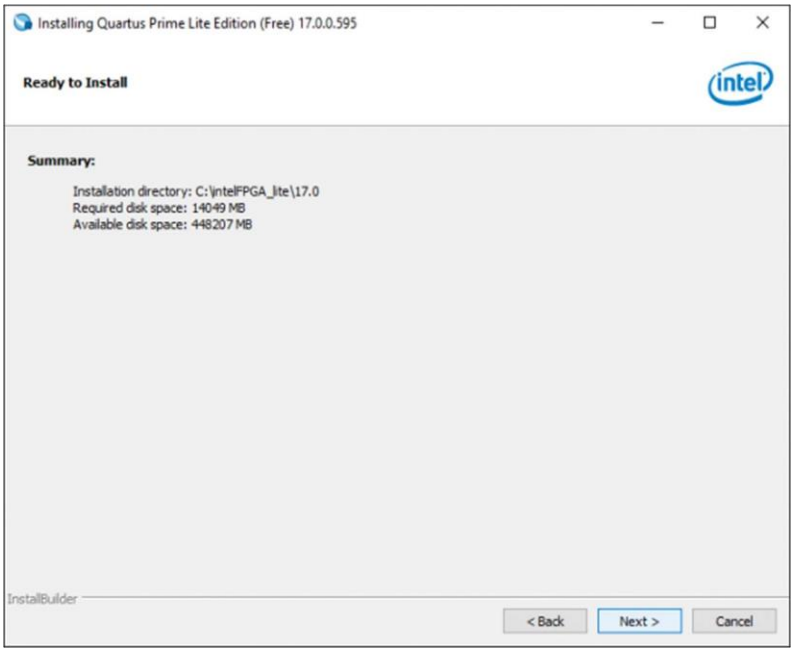


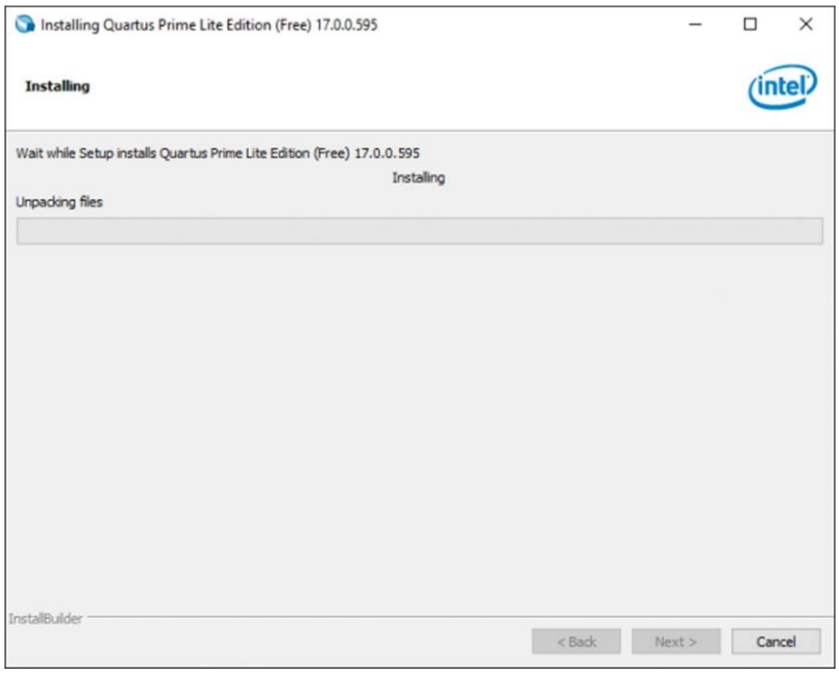
Se selecciona un directorio para la instalación y después dar clic “Next”.



Saldrá una ventana de selección de componentes, se marcan los elementos que son necesarios, en este caso son los que están en escala de grises, se deben marcar todos los elementos excepto el último, como se muestra en la imagen.

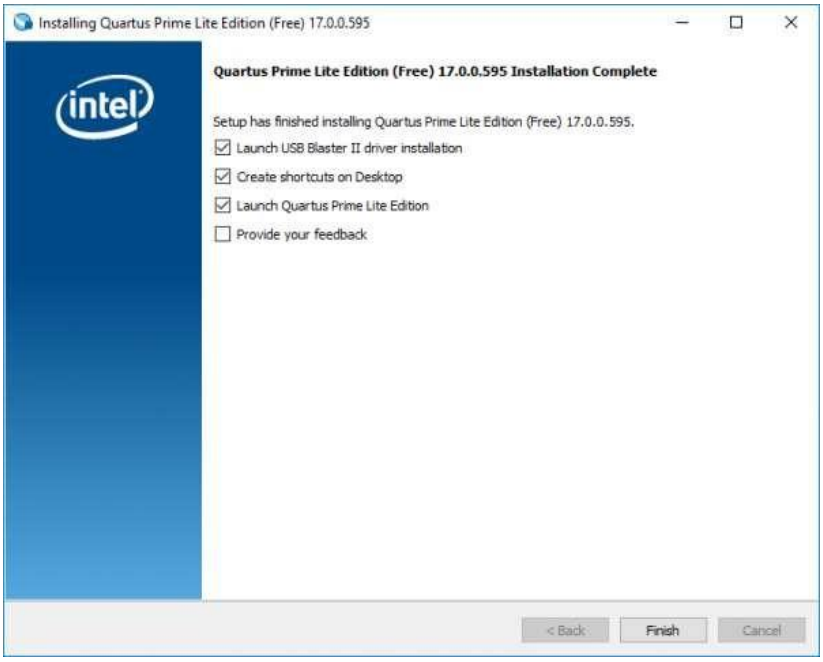
Luego aparece una ventana con información del espacio requerido para la instalación, dar “Next” (en caso de no contar con espacio suficiente en el disco no podrá instalar el software).





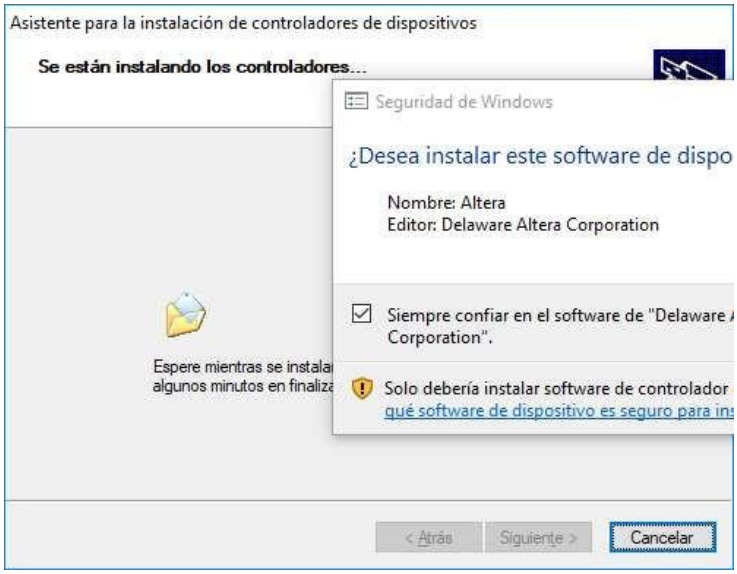
El proceso de instalación comienza, este proceso puede llevar de 30 minutos a 1 hora y media, dependiendo del tipo de procesador y disco duro con el que se cuente.

Terminada la instalación nos aparecen las siguientes opciones:

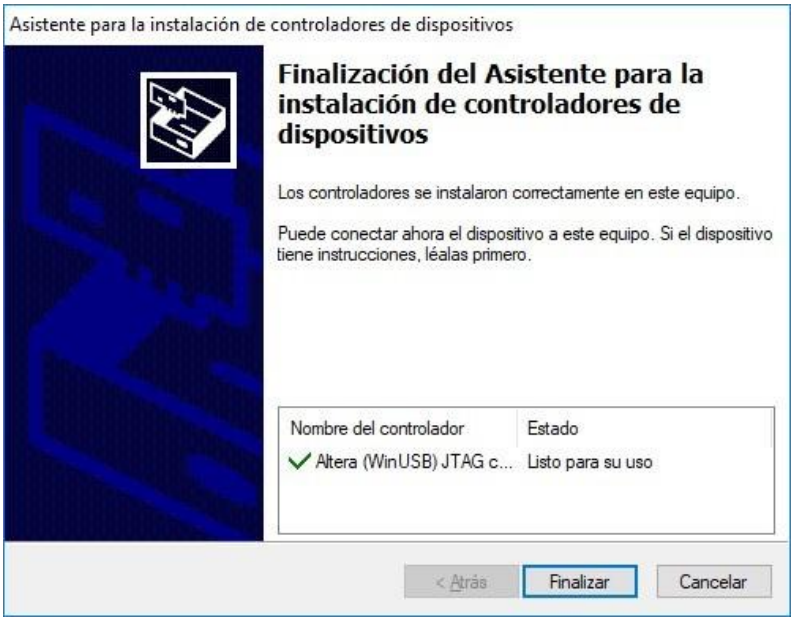


Todas son acciones opcionales a gusto del usuario, la opción ¨Launch USB-Blaster II driver installation¨, se refiere al driver para el programador que está incluido en la tarjeta es necesario dejar esta opción marcada.

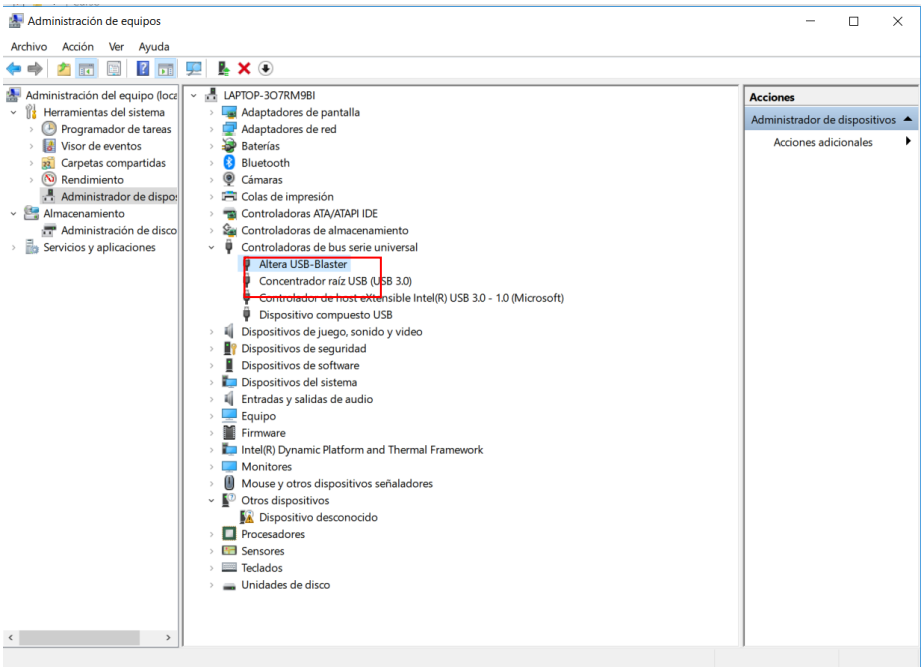
Aparecerá la siguiente ventana, se debe aceptar la instalación del driver.



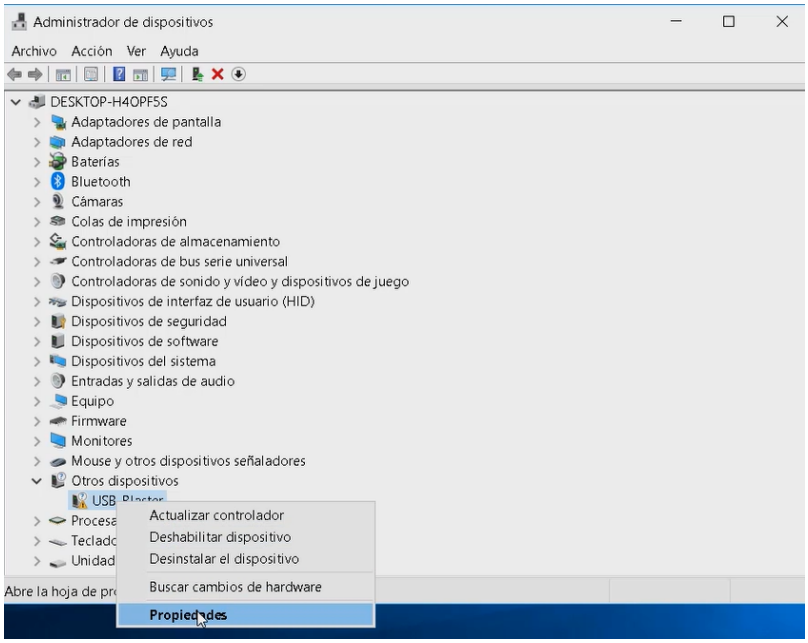
La siguiente ventana indica que todo se instaló de forma correcta.



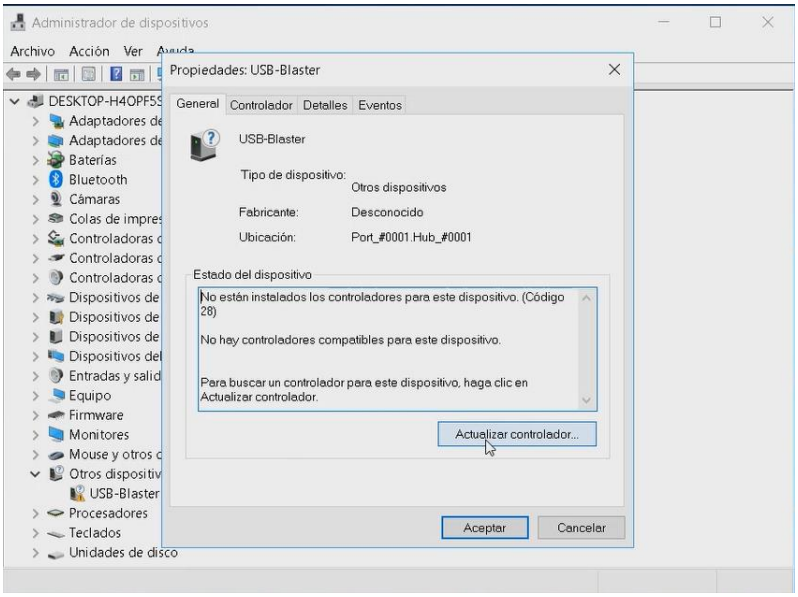
Para comprobar que el controlador está instalado es necesario conectar la tarjeta DIONE a la computadora mediante el cable USB y encenderla. Después de esto abrir el administrador de dispositivos y deberá aparecer una ventana como la siguiente.

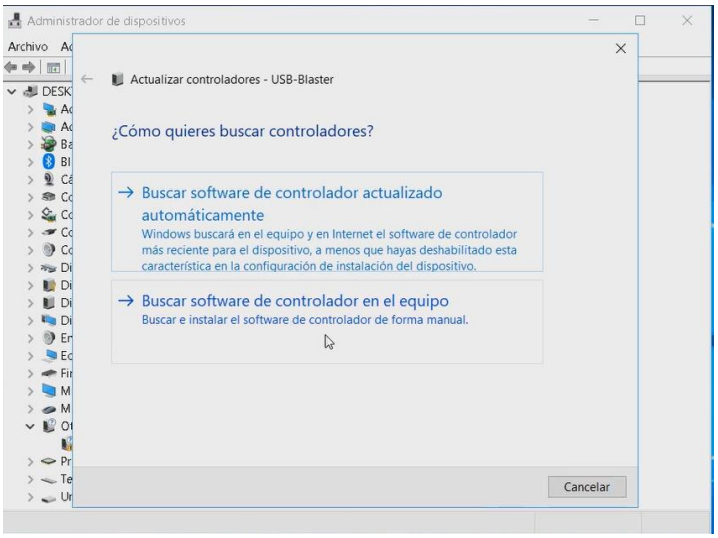


Si por algún motivo el controlador del USB Blaster tuvo un error en la instalación y no aparece de este modo se debe instalar el controlador desde la carpeta raíz donde se instaló Quartus, para esto abrir el administrador de dispositivos, seleccionar el dispositivo ¨USB-Blaster", dar clic derecho "propiedades".



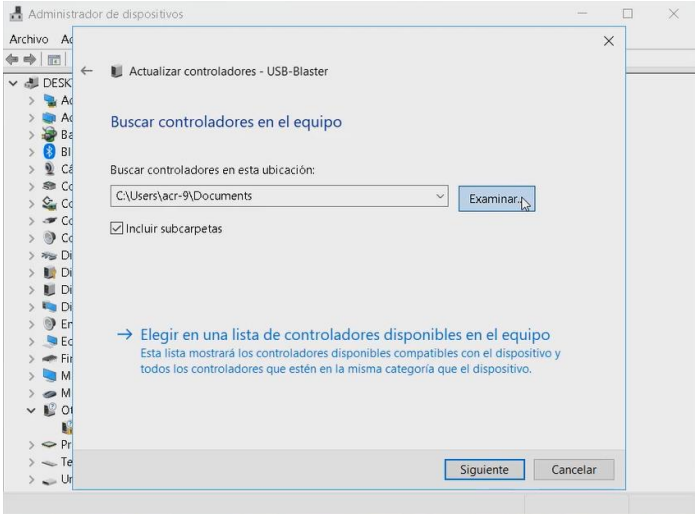
Aparecerá una ventana como la siguiente, dar clic en “Actualizar controlador”,

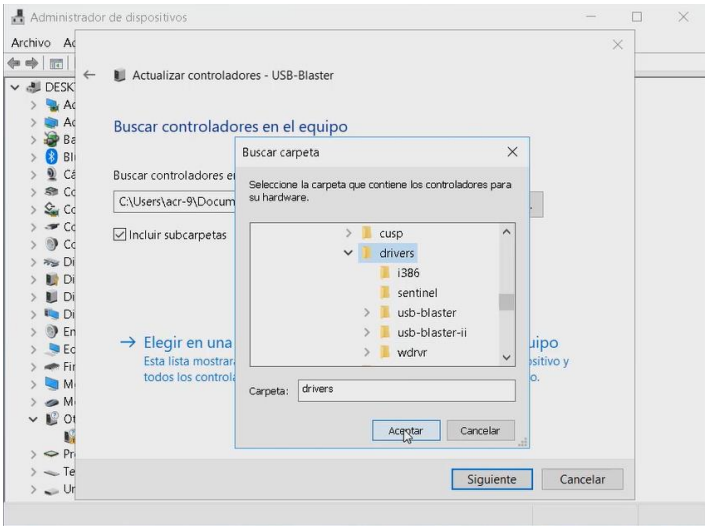




después, aparece otra ventana, damos clic en la opción “buscar software del controlador en el equipo”.

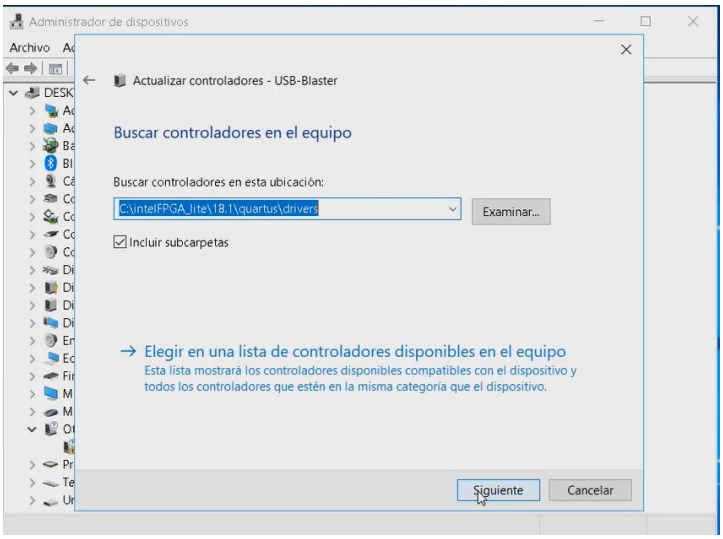
Aparece la siguiente ventana, en este paso debemos seleccionar la carpeta donde se encuentra el driver, que se encuentra en c:\intelfpga_lite\18.0\quartus\drivers.

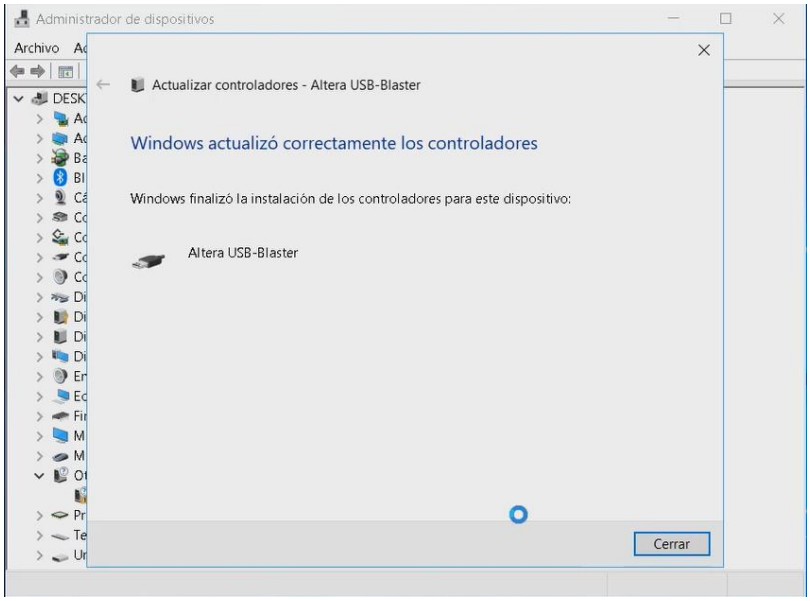




Una vez encontrada la carpeta "drivers" se selecciona y se da clic en "aceptar".

Luego dar clic en "siguiente" y comenzará la instalación.





Una vez concluida la instalación cerrar las ventanas del administrador de dispositivos. Ahora está todo listo para trabajar en la tarjeta DIONE.

Hasta este punto se tiene todo lo necesario para crear proyectos en Quartus. A continuación, se muestran una serie de ejemplos de cómo usar la tarjeta DIONE CPLD y sus periféricos.

Ejemplos

A continuación, se presentan una serie de ejemplos que le ayudarán a familiarizarse con el lenguaje de programación de hardware VHDL, usando el entorno Quartus 18.1 y la tarjeta DIONE CPLD de NIBBLUX. Para más información sobre la tarjeta DIONE consultar el documento "DIONE CPLD Especificaciones Técnicas" que se encuentra en www.nibblux.com o en Facebook /Nibblux.

Ejemplo 1 Compuertas Lógicas

Las compuertas lógicas son la base de la electrónica digital, con ellas se pueden crear diferentes tipos de circuitos digitales con diferentes funciones, en la actualidad las compuertas lógicas de tipo TTL están siendo remplazadas por dispositivos lógicos programables como CLPDs y FPGAs. Estos dispositivos están construidos por múltiples elementos lógicos que se interconectan entre sí por medio de software para crear circuitos como compuertas, decodificadores, multiplexores, etc., todo esto a través de un leguaje de descripción de hardware (HDL). Estos leguajes de descripción de hardware realizan un modelo del dispositivo, para ello realizan una abstracción que se le conoce como Register Transfer Logic (RTL), que es una forma de transferir la sintaxis escrita en VHDL a un diagrama lógico.

Después de realizar la descripción se realiza la síntesis. En la síntesis se realiza la conexión de elementos lógicos según la descripción hecha en código y se crea un archivo de tipo "pof" que posteriormente se carga al dispositivo.

Teniendo en cuenta todo lo anterior se realizarán algunas descripciones de compuertas lógicas usando un leguaje de descripción de hardware. Se usará el entorno de Quartus 18.1 y el lenguaje VHDL.

FUNCIONES LÓGICAS BÁSICAS

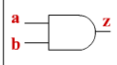
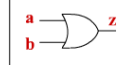
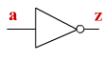
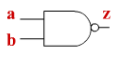
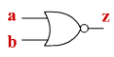
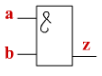
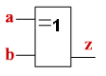
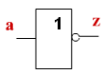
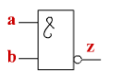
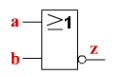
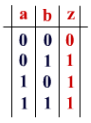
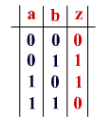
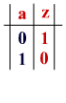
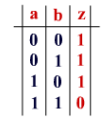
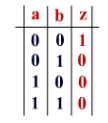
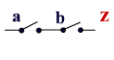
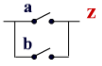
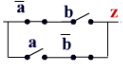
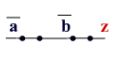
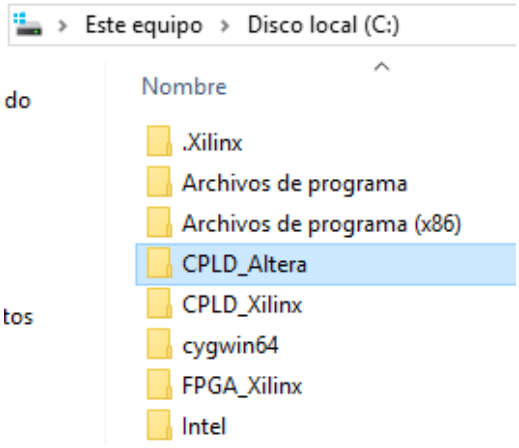
NOMRE	AND - Y	OR - O	XOR O-exclusiva	NOT Inversor	NAND	NOR
SÍMBOLO						
SÍMBOLO						
TABLA DE VERDAD						
EQUIVALENTE EN CONTACTOS						
AXIOMA	$z = a \cdot b$	$z = a + b$	$z = \bar{a} \cdot b + a \cdot \bar{b}$	$z = \bar{a}$	$z = \overline{a \cdot b}$	$z = \overline{a + b}$

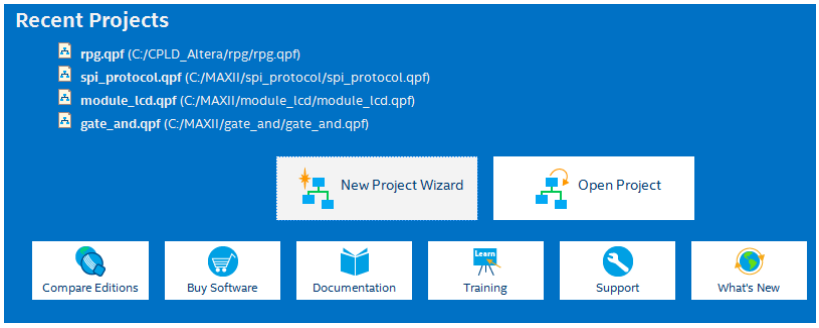
Tabla 1. Tablas de verdad de compuertas lógicas. En la tabla se muestran los símbolos lógicos y su representación física como interruptores.

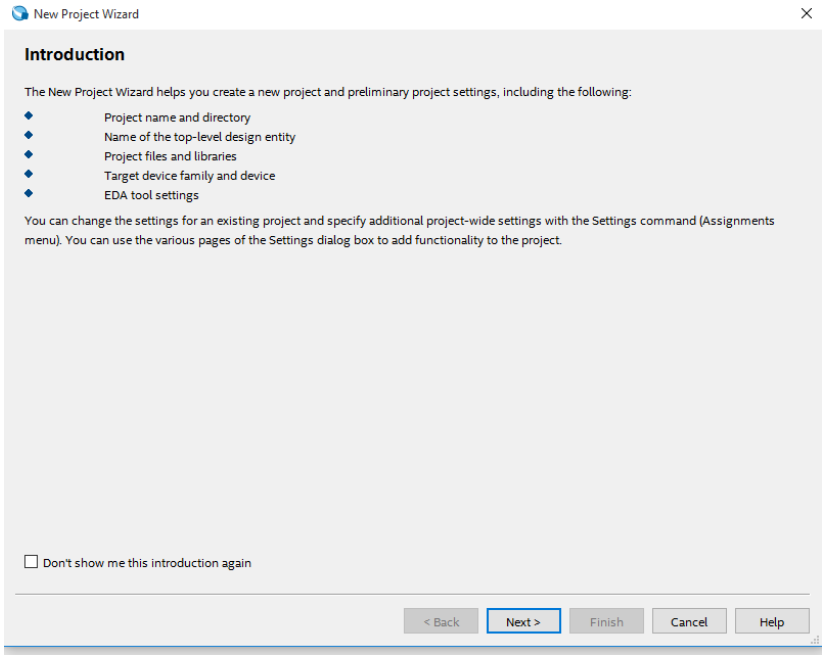
A continuación, se realizan los pasos para la creación del proyecto.

1.- En el disco local C crear una carpeta, puede tener cualquier nombre, en este caso es CPLD_Altera. En esta carpeta se guardarán todos los proyectos que se realicen.



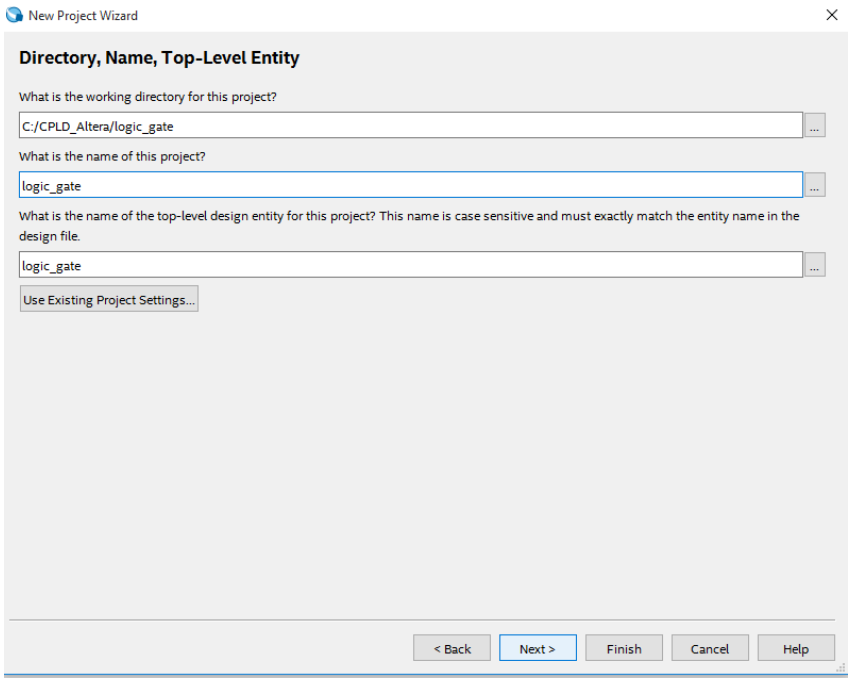
2. Abrir el entorno de Quartus 18.1 y crear un nuevo proyecto wizard (seleccionar New Project wizard).

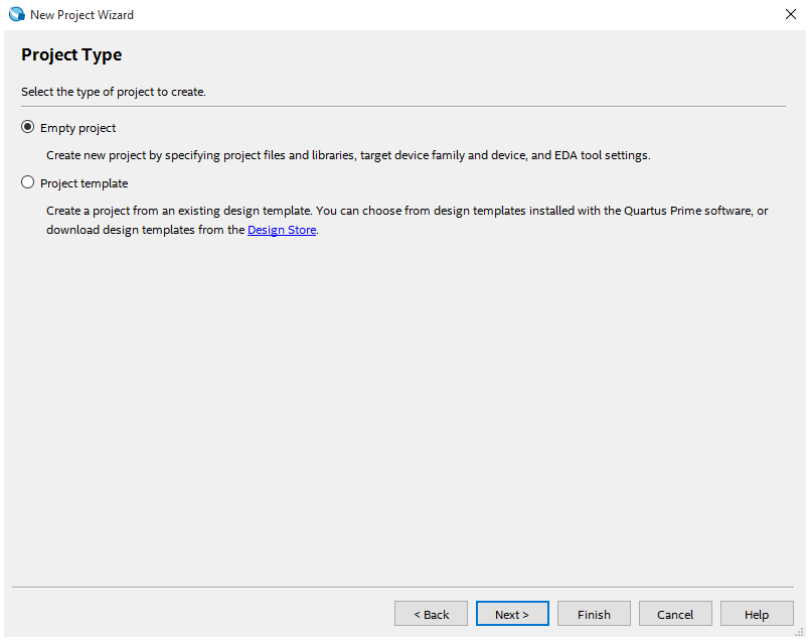




3. Al seleccionar la opción de New Project wizard se abrirá una ventana en donde se especifica el nombre del proyecto y la ruta. Clic Next.

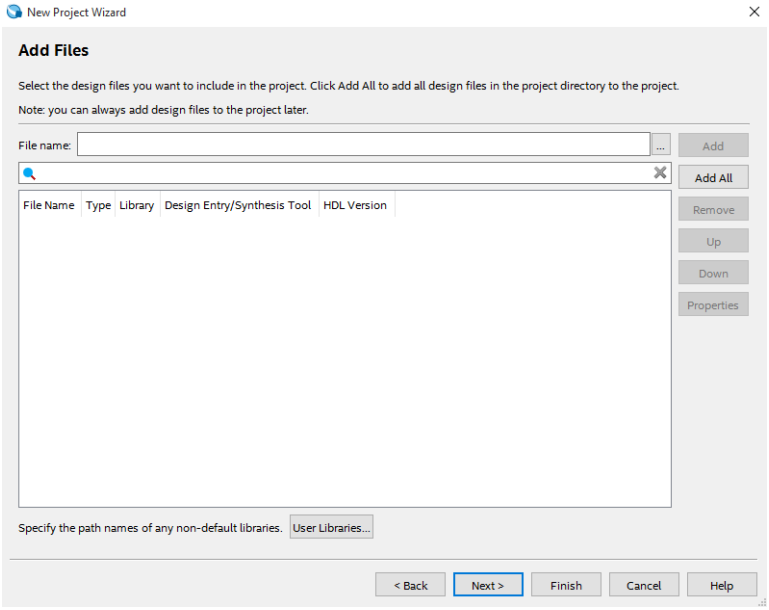
4.- Dentro de la carpeta CPLD_Altera se debe crear otra con el nombre del proyecto, para este caso "logic_gate", se selecciona la carpeta logic_gate como directorio del proyecto. En el siguiente renglón se escribe el nombre del proyecto, este debe ser igual al nombre de la carpeta donde se va a guardar, para este caso "logic_gte". Clic en next.





5. Después aparece la opción de usar una platilla o un proyecto vacío. Seleccionar "Empty proyect" para tener un proyecto vacío. Clic en next.

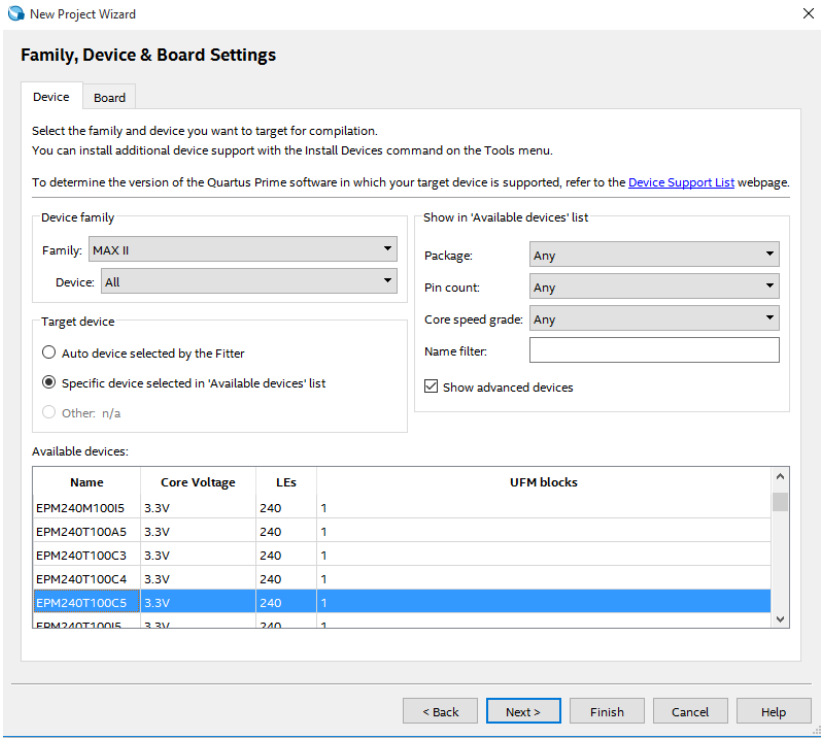
6. Ahora se tiene que especificar si se utilizarán archivos existentes. Esta opción es útil Cuando se tienen dispositivos descritos en leguaje VHDL y se desea usarlos. En este caso no se tiene ningún dispositivo, por lo que se omite este paso. Clic en next.



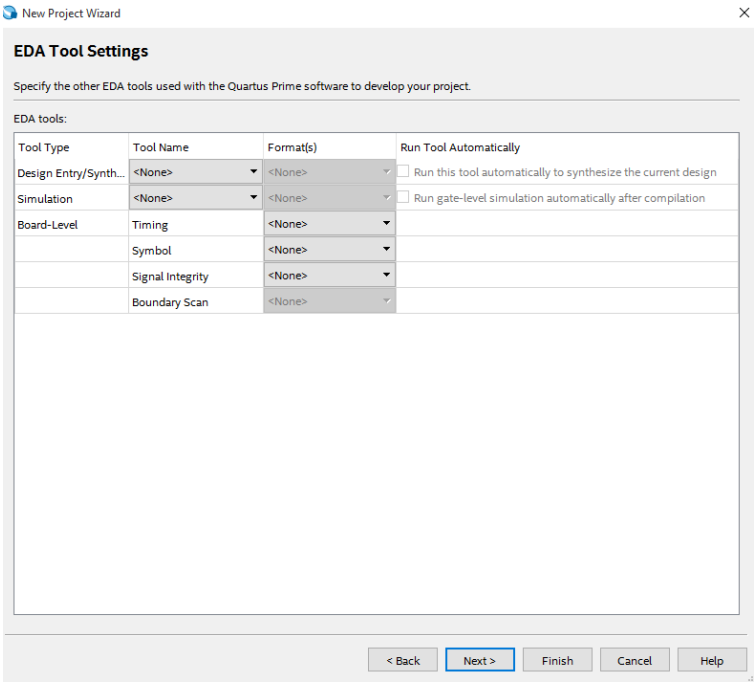
7. Después se debe especificar el chip a utilizar.

Familia: MAXII
Dispositivo: todos
Chip: EPM240T100C5

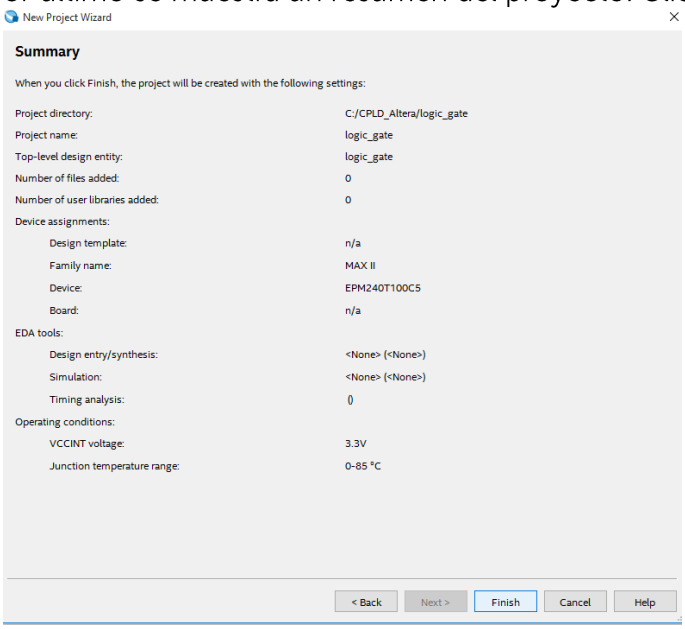
Clic en next.



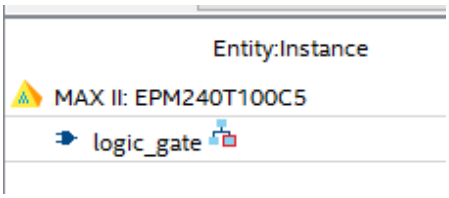
8. Una vez seleccionado el dispositivo se especifica si se va a utilizar una tarjeta o un simulador en específico, por este momento omitiremos estas opciones. En otras aplicaciones se utilizarán las simulaciones. Clic en next.



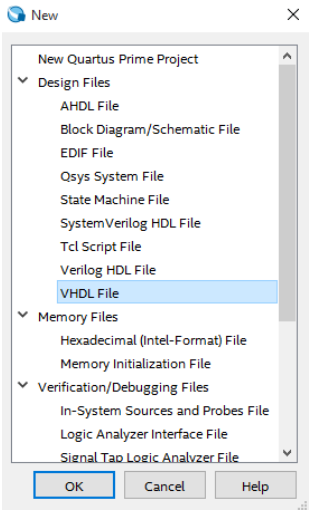
9. Por último se muestra un resumen del proyecto. Clic en



10. Una vez realizados los pasos anteriores, en la parte superior izquierda del entorno aparecerá el proyecto y el dispositivo utilizado.



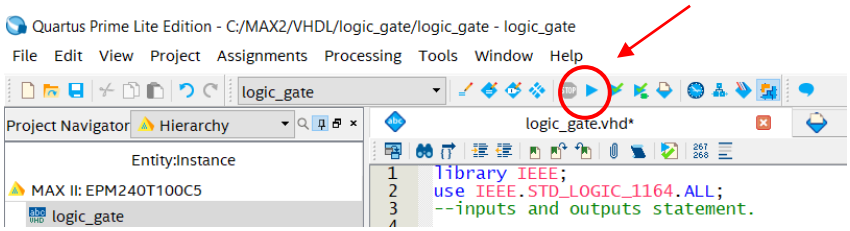
11. Ahora es momento de comenzar con el código. En la pestaña "New", se crea un archivo VHDL y se guarda con el mismo nombre del proyecto, en este caso "logic_gate.vhd". Otra forma de hacer esto es utilizar el comando ctrl + n y seleccionar VHDL file. Clic en ok.



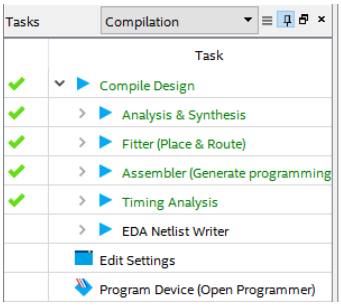
12. A continuación se muestra un ejemplo de cómo es la estructura del código para describir las compuertas lógicas. Escribimos este código en el archivo VHDL.

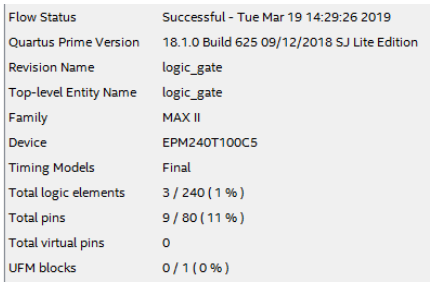
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY LOGIC_GATE IS

    PORT ( A1 : IN STD_LOGIC;    --and gate input.
           B1 : IN STD_LOGIC;    --and gate input.
           A2 : IN STD_LOGIC;    --or gate input.
           B2 : IN STD_LOGIC;    --or gate input.
           A3 : IN STD_LOGIC;    --xor gate input.
           B3 : IN STD_LOGIC;    --xor gate input.

           Y1 : OUT STD_LOGIC;   --and gate output.
           Y2 : OUT STD_LOGIC;   --or gate output.
           Y3 : OUT STD_LOGIC); --xor gate output.
END LOGIC_GATE;

ARCHITECTURE BEHAVIORAL OF LOGIC_GATE IS
BEGIN

    Y1 <= A1 AND B1;   --The sintax fir the and gate
    Y2 <= A2 OR  B2;   --The sintax fir the or gate
    Y3 <= A3 XOR B3;   --The sintax fir the xor gate

END BEHAVIORAL;
```

13. Una vez que se ha definido el código se procede a realizar la síntesis del programa, en este paso se hace una verificación del código para detectar los posibles errores de sintaxis, para esto se da clic en la opción "Start Compilation" que se encuentra en la parte superior del entorno de Quartus 18.1.



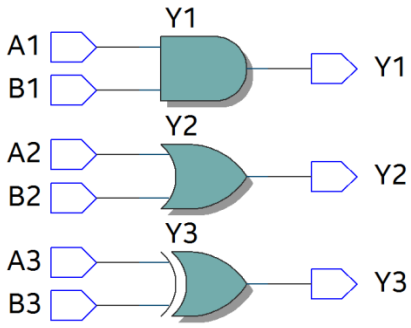
14. Si no hay ningún error de sintaxis Quartus dará un reporte del proyecto, en donde indica la cantidad de componentes además de sus entradas y salidas.



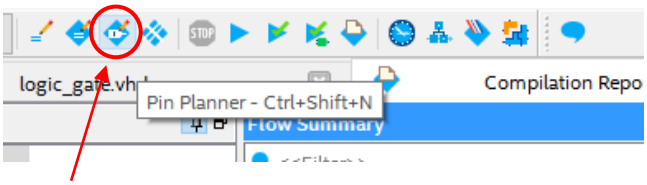


Flow Status	Successful - Tue Mar 19 14:29:26 2019
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	logic_gate
Top-level Entity Name	logic_gate
Family	MAX II
Device	EPM240T100C5
Timing Models	Final
Total logic elements	3 / 240 (1 %)
Total pins	9 / 80 (11 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

15. Después de realizar la síntesis se utilizará el análisis RTL, que es una herramienta que nos permite ver en forma de diagrama esquemático el código que realizamos anteriormente. Para ello se debe ir en la pestaña "tools", "Netlist viewer" y seleccionar "RTL viewer".



15. Al ver el análisis RTL se pueden observar que las entradas y salidas del circuito no están definidas, para definirlas nos dirigimos al menú superior central donde está la opción "pin planner". Cuando se selecciona esta opción se genera una ventana donde se muestra el dispositivo con sus puertos de entras/salida.



La tarjeta DIONE de NIBBLUX contiene etiquetas en la parte superior e inferior las cuales permiten ubicar el número de pin correspondiente a cada entrada y salida, este número de pin se escribe en la columna "Location" en la ventana del "pin planner" para asignar un pin en específico a cada variable de entrada y salida. Las entradas A1, A2, A3, B1, B2, y B3 se asignan a los pines PIN_1, PIN_2, PIN_3, PIN_4, PIN_5 Y PIN_6 respectivamente, estos pines están indicados en la tarjeta y corresponden a puertos asignados a interruptores que incluye la tarjeta.
Las salidas Y1, Y2 y Y3 se conectarán a los pines PIN_41, PIN_39 y PIN_37 respectivamente, estos pines corresponden a tres de los cuatro leds que se encuentran en la tarjeta DIONE, estos pines no pueden ser entradas ya que la conexión con los leds está predeterminada como salida.

Una vez asignadas las entradas y salidas se debe volver a usar la opción "Start compilation" para que se guarden los cambios.

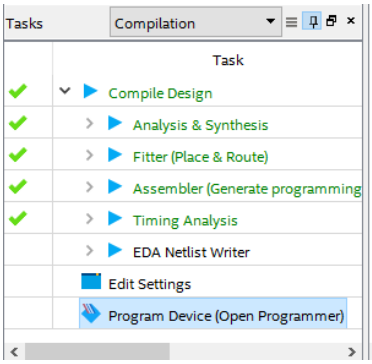
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate Preservati
A1	Input	PIN_1	2	PIN_1	3.3-V LVTTL		16mA (default)	
A2	Input	PIN_3	1	PIN_3	3.3-V LVTTL		16mA (default)	
A3	Input	PIN_5	1	PIN_5	3.3-V LVTTL		16mA (default)	
B1	Input	PIN_2	1	PIN_2	3.3-V LVTTL		16mA (default)	
B2	Input	PIN_4	1	PIN_4	3.3-V LVTTL		16mA (default)	
B3	Input	PIN_6	1	PIN_6	3.3-V LVTTL		16mA (default)	
Y1	Output	PIN_41	1	PIN_41	3.3-V LVTTL		16mA (default)	
Y2	Output	PIN_39	1	PIN_39	3.3-V LVTTL		16mA (default)	
Y3	Output	PIN_37	1	PIN_37	3.3-V LVTTL		16mA (default)	

16. La tarjeta DIONE cuenta con fuentes de voltaje de 3.3V, la función de estas fuentes es alimentar los circuitos de las entradas.

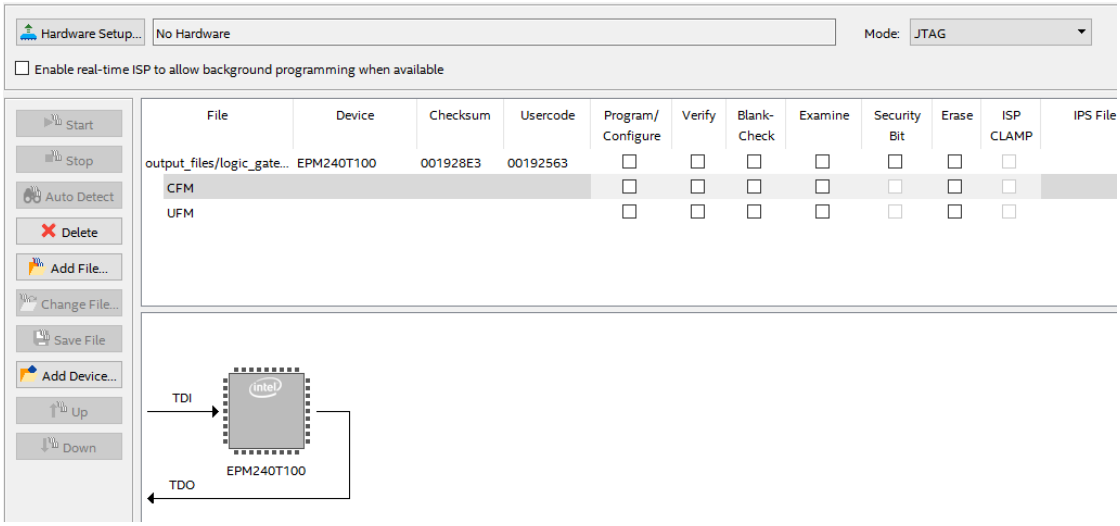
Para las entradas se debe usar un resistor de pull up o pull down, en el documento "DIONE CPLD Especificaciones Técnicas" se recomienda usar un resistor de 10K ohm.

Como las entradas presentan una alta impedancia el flujo de corriente en la entrada es mínimo y se puede despreciar. Esto no ocurre en las salidas, LOS PINES UTILIZADOS COMO SALIDAS SI PRESENTAN FLUJO DE CORRIENTE, por lo que es necesario que esta corriente sea controlada por un resistor según la aplicación. LA CORRIENTE MÁXIMA DE SALIDA DE CADA PIN DE LA TARJETA DIONE ES DE 16mA, si la corriente de salida en un pin excede este valor la CPLD se dañará. Recordemos que el voltaje de salida de cualquier pin es 3.3V por lo que la corriente de salida se calcula usando la ley de ohm. Se recomienda que el mínimo valor de resistor de salida sea de 330 ohm de este modo la corriente de salida máxima será de 10mA, esta es la corriente de salida máxima sugerida ya que si se elige 16mA se estará trabajando el pin en su límite máximo y eso acorta su vida útil.

17. Por último se procede a programar la tarjeta. Para esto, una vez sintetizado el código y con los pines de entrada y salida asignados, se selecciona la opción "program Device", que se encuentra en el panel izquierdo del entorno de Quartus, al hacer doble clic se abrirá una ventana.

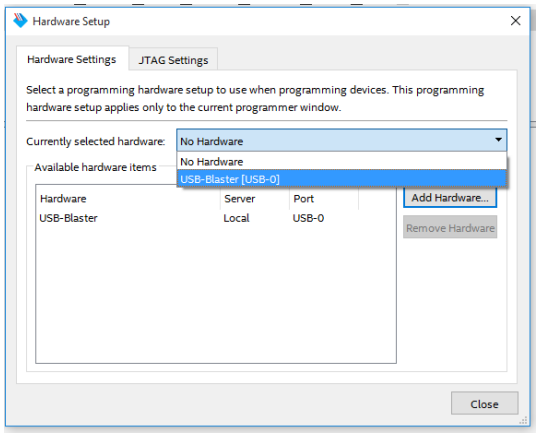


En este momento deberá conectar y encender la tarjeta Dione. Aparecerá la siguiente ventana.

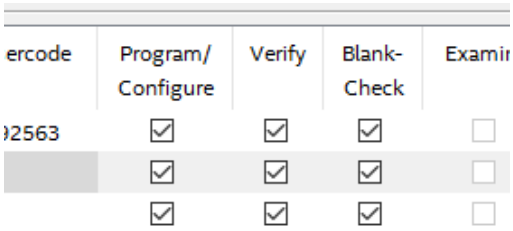


Seleccionar la opción de “Hardware Setup” para seleccionar el hardware a usar, en este caso la tarjeta DIONE, para ello dar doble clic en la opción “USB-Blaster [USB-0]” y después se da clic en “Close”.

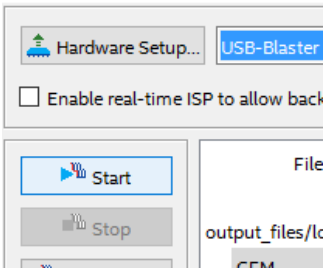
Nota: Si no aparece el USB-Blaster ir a la página 13 de este documento para la instalación del controlador.

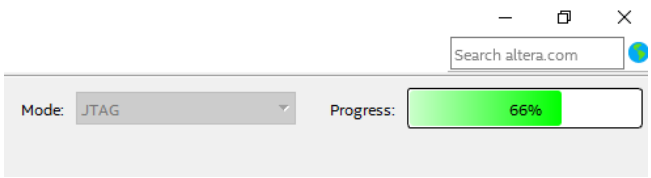


Después de seleccionar el hardware a programar deberá marcar las siguientes opciones



Por último, dar clic en el botón “Start” y comenzará la programación mediante el protocolo JTAG. Podemos ver el progreso de la programación de la tarjeta en la barra de estado que aparece en la parte superior derecha de la ventana de programación.





Ahora se procede a verificar el funcionamiento de las compuertas lógicas en la tarjeta DIONE.

Las figuras están ordenas por la posición de los interruptores: 00, 01, 10 y 11.

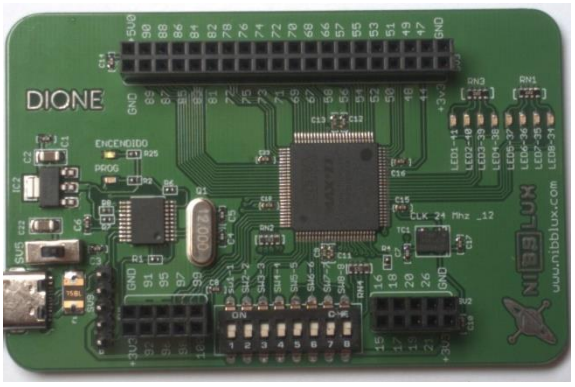


Figura 1.1. Interruptores en la posición 00, ninguna salida en alto.

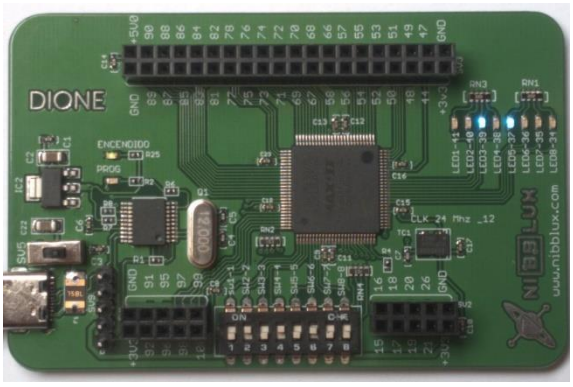


Figura 1.2. Interruptores en la posición 01, Y2 y Y3 en estado alto.

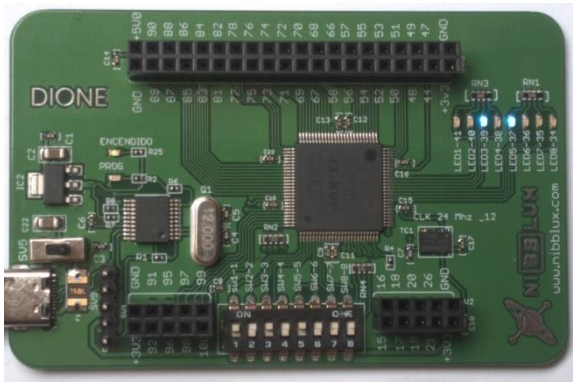


Figura 1.3. Interruptores en la posición 10, Y2 y Y3 en estado alto.

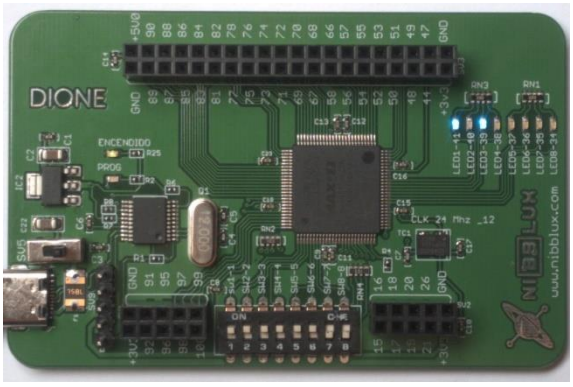


Figura 1. Interruptores en la posición 11, Y1 y Y2 en estado alto.

¡Felicidades! Usted logró implementar las compuertas AND, OR y XOR en la tarjeta DIONE. Con los mismos pasos usted puede realizar otros códigos y probarlos en DIONE de manera sencilla.

Ejemplo 2 Funciones Booleanas

El algebra de funciones booleanas difiere en gran medida del algebra lineal, ya que las constantes y variables booleanas solo permiten tener dos valores posibles: 0 y 1. Las variables booleanas se utilizan a menudo para representar un nivel de voltaje presente en un alambre o en las terminales de entrada/salida de un circuito.

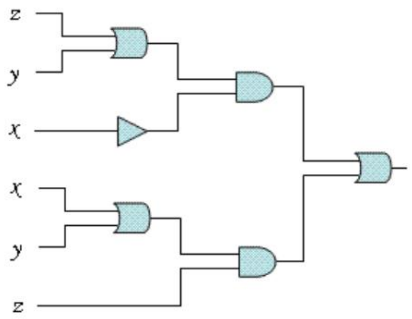
Las funciones booleanas son funciones que pueden depender de n entradas y son muy extensas. Cuando se emplean físicamente con compuertas digitales de la familia 74 es muy costoso además de que se pierde mucho tiempo y es muy fácil cometer errores al cablear, por esa razón hay teoremas de algebra de Bool, leyes de Morgan y mapas de Karnoaug para reducir estas funciones y con esto el tamaño de los circuitos. Cuando se realizan funciones Booleanas en VHDL los problemas mencionados desaparecen. Muchos entornos como Quartus, ISE Design Suite o Vivado Desing Suite se encargan de optimizar las funciones booleanas.

A continuación, se implementará una función booleana en la tarjeta DIONE de NIBBLUX.

Considere la siguiente función Booleana:

$$f(x,y,z) = (x+y) * z + \bar{x} * (y+z)$$

Su implementación en compuertas es la siguiente:



Y su tabla de verdad es:

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

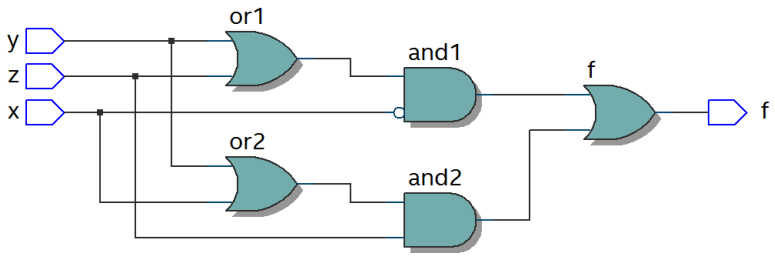
Tabla 2. Tabla de verdad de la función booleana f(x,y,z).

La función booleana se implementará a partir de su representación en compuertas lógicas. A continuación, se muestra la sintaxis o descripción de código para implementar la función booleana.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- inputs and outputs statement.
entity bolean_function is
    Port ( x : in  STD_LOGIC;    -- input.
           y : in  STD_LOGIC;    -- input.
           z : in  STD_LOGIC;    -- input.
           f : out  STD_LOGIC); -- output.
end bolean_function;

architecture Behavioral of bolean_function is
signal and1,and2,or1,or2,or3,not1 : std_logic := '0';
begin
-- The syntax for the not1 gate.
not1 <= not(x);
-- The syntax for the or1 gate.
or1 <= y or z;
-- The syntax for the or2 gate.
or2 <= y or x;
-- The syntax for the and1 gate.
and1 <= or1 and not1;
-- The syntax for the and2 gate.
and2 <= or2 and z;
-- The syntax for the or3 gate.
f <= and1 or and2;

end Behavioral;
```

En este código se hace uso de una nueva declaración, una señal. En VHDL una señal equivale a un alambre o cable que interconecta a las compuertas entre sí, la salida de una compuerta o un bloque de compuertas con la entrada de otra compuerta o bloque de compuertas. En el código anterior se declaran las señales and1, and2, or1, or2, not1 y or3. not1 es salida de la compuerta not y además es la entrada de la compuerta and. La señal not1 une ambas compuertas. or1 es la señal, de salida de la compuerta or y además es la entrada de la compuerta and. La señal or1 une ambas compuertas.
Y así para todas las señales, por esta razón son una especie de cable virtual. La señal or3 se puede incluir, de la siguiente manera:

or3 <= and1 or and2;
f<= or3;
Después de tener toda la descripción se realiza la síntesis y comprobación de sintaxis. Si no hay errores el análisis RTL queda de la siguiente manera:

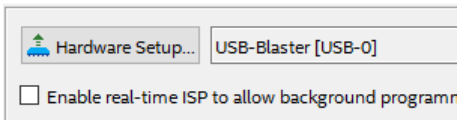


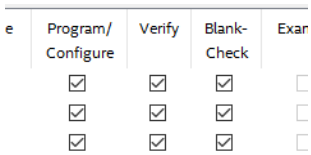
Ahora se seleccionan las entradas y salidas con la opción “pin planner”.

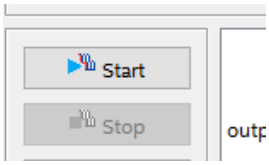
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Streng	ict Preservati
out F	Output	PIN_41	1	PIN_41	3.3-V LVTTL		16mA ...ault)	
in X	Input	PIN_1	2	PIN_1	3.3-V LVTTL		16mA ...ault)	
in Y	Input	PIN_2	1	PIN_2	3.3-V LVTTL		16mA ...ault)	
in Z	Input	PIN_3	1	PIN_3	3.3-V LVTTL		16mA ...ault)	
<<new node>>								

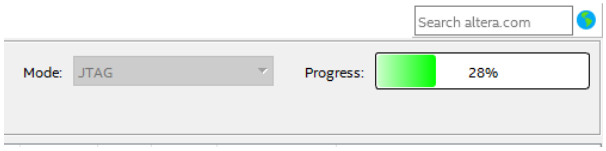
Al terminar de agregar las entradas y salidas es necesario realizar la síntesis y comprobar que no hay errores de sintaxis.

Se programa la tarjeta Dione. En “Hardware Setup” seleccionar USB-Blaster [USB-0], marcar las opciones “program”, “verify” y “bank check”. Después programar y que el proceso termine.









Después de programar la tarjeta DIONE se puede verificar la tabla de verdad de la función booleana. Las figuras están acomodadas en el orden: 000, 001, 010, 011, 100, 101, 110, 111

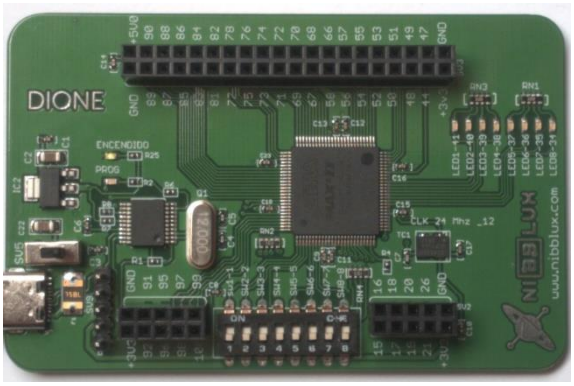


Figura 2.1. XYZ=000, F=0

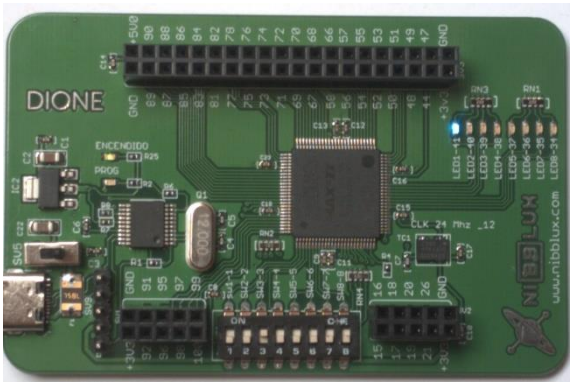


Figura 2.2 XYZ=001, F=1

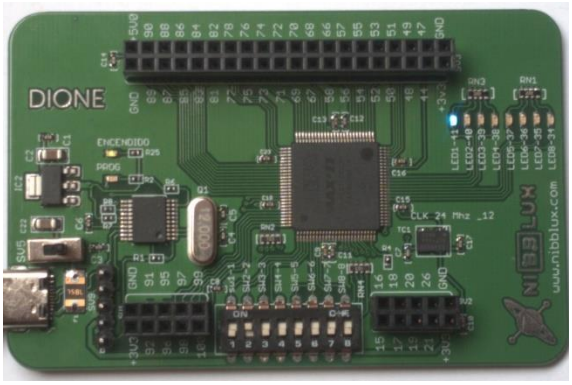


Figura 2.3. XYZ = 010, F = 1.

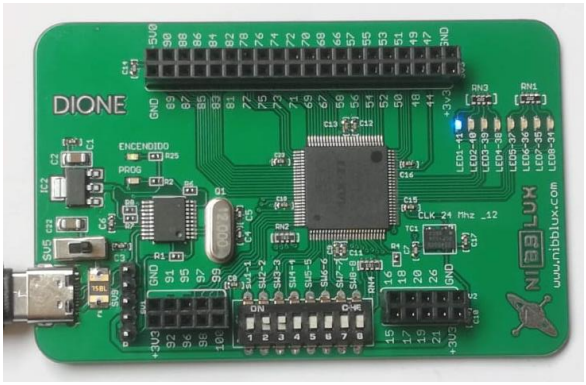


Figura 2.4. XYZ = 011, F = 1.

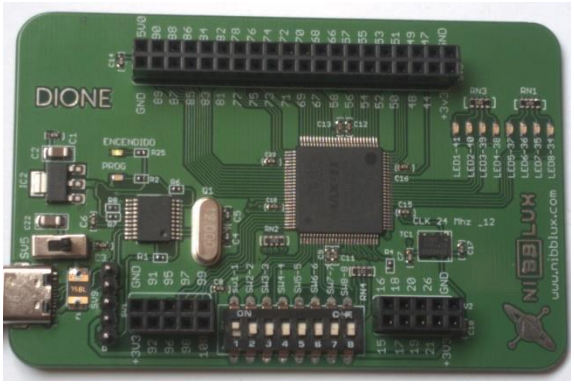


Figura 2.5. XYZ = 100, F = 0.

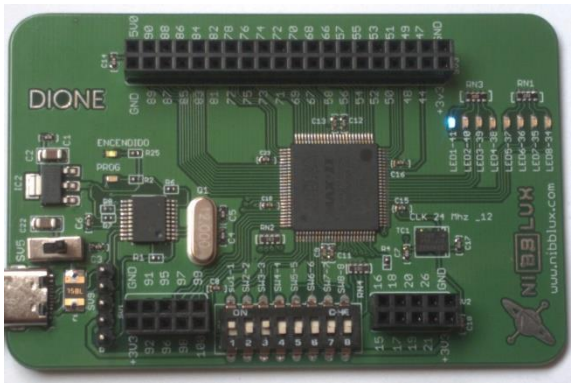


Figura 2.6. XYZ = 101, F = 1.

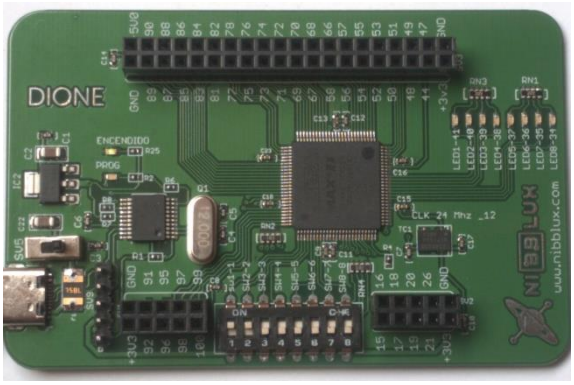


Figura 2.7. XYZ=110, F=0

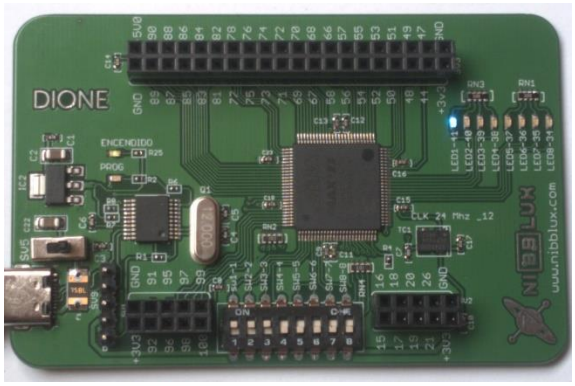


Figura 2.8. XYZ=111, F=1

Ejemplo 3 Decodificadores.

Un decodificador es un circuito combinacional que consta de n entradas y 2^n salidas. Cada salida es asignada y corresponde a cada una de las combinaciones de la entrada. A continuación, se presenta el diagrama lógico de un decodificador de cuatro líneas y su tabla de verdad.

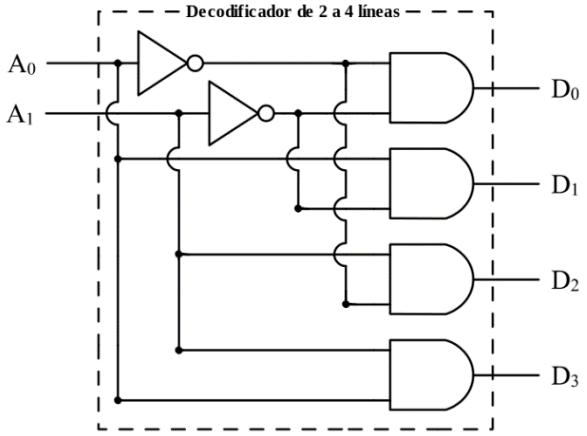


Tabla de verdad

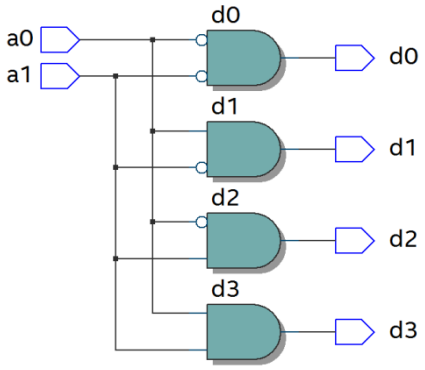
A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Tabla 3. Tabla de verdad de la función booleana f(x,y,z).

Para implementar el decodificador se tiene que realizar su diagrama lógico. Hay muchas formas en código de implementar los decodificadores, en el siguiente ejemplo se realizará un decodificar de 2 a 4 utilizando su diagrama lógico. La descripción del decodificador 2 a 4 es la siguiente:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- inputs and outputs statement.
entity decoder is
    Port ( a0 : in   STD_LOGIC;    -- input.
           a1 : in   STD_LOGIC;    -- input.
           d0 : out  STD_LOGIC;    -- output.
           d1 : out  STD_LOGIC;    -- output.
           d2 : out  STD_LOGIC;    -- output.
           d3 : out  STD_LOGIC);   -- output.
end decoder;

architecture Behavioral of decoder is
begin
-- The syntax for the do output.
d0 <= (not(a0) and not(a1));
-- The syntax for the d1 output.
d1 <=  (a0 and not(a1));
-- The syntax for the d2 output.
d2 <= (not(a0) and a1);
-- The syntax for the d3 output.
d3 <= a0 and a1;

end Behavioral;
```

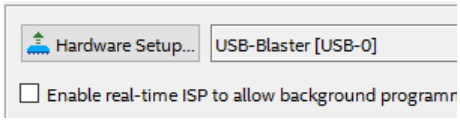
El análisis RTL se muestra a continuación:

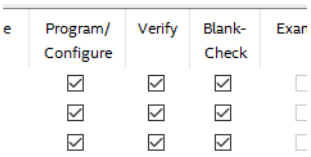


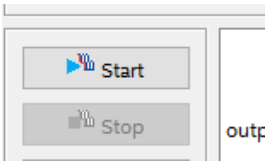
Después, se seleccionan las entradas en pin planner y luego se da clic en Start Compilation.

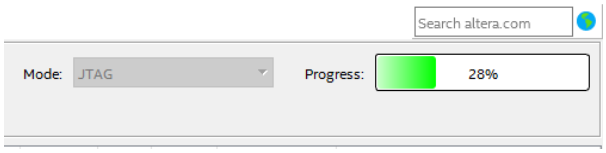
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Streng	ict Preservati
A0	Input	PIN_2	1	PIN_2	3.3-V LVTTL		16mA ...ault)	
A1	Input	PIN_1	2	PIN_1	3.3-V LVTTL		16mA ...ault)	
D0	Output	PIN_41	1	PIN_41	3.3-V LVTTL		16mA ...ault)	
D1	Output	PIN_39	1	PIN_39	3.3-V LVTTL		16mA ...ault)	
D2	Output	PIN_37	1	PIN_37	3.3-V LVTTL		16mA ...ault)	
D3	Output	PIN_35	1	PIN_35	3.3-V LVTTL		16mA ...ault)	

Por último, programar la tarjeta Dione.









Ahora comprobar la tabla de verdad del decodificador

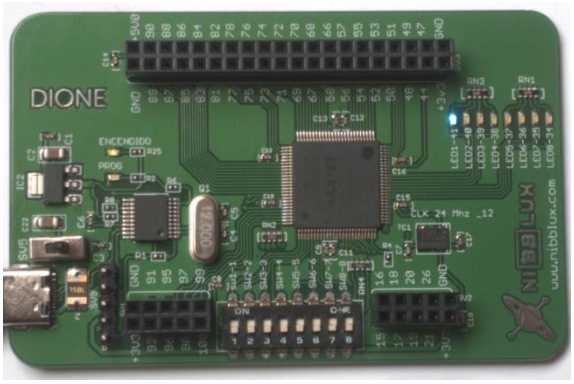


Figura 3.4. A1A0 = 11, D0D1D2D3 = 1000

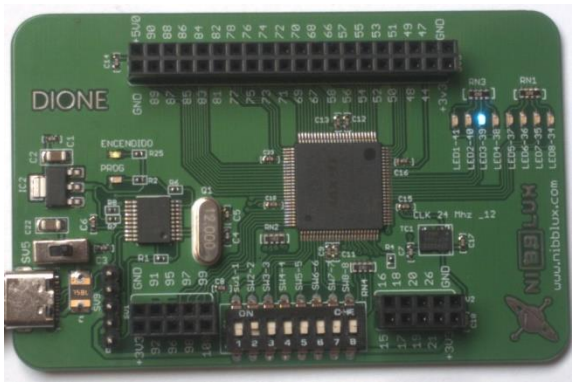


Figura 3.3. A1A0 = 10, D0D1D2D3 = 0100

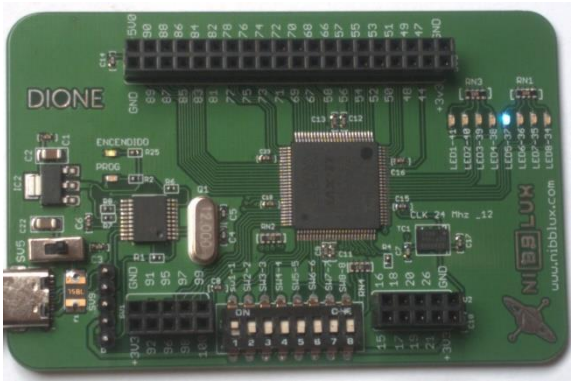


Figura 3.2. A1A0 = 01, D0D1D2D3 = 0010

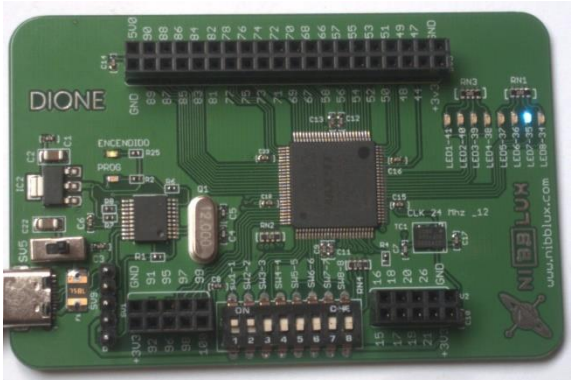


Figura 3.1. A1A0 = 00, D0D1D2D3 = 0001

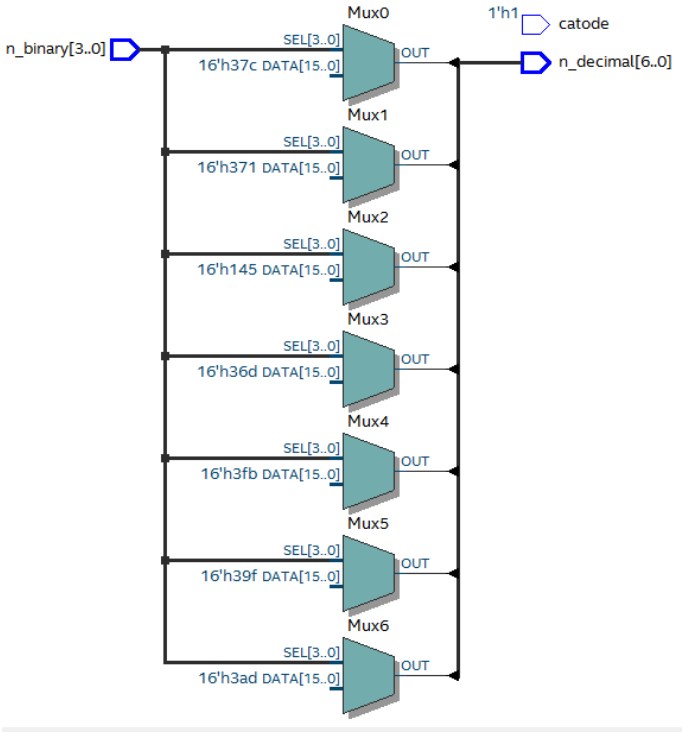
Ejemplo 4 Decodificador binario BCD.

Un decodificador binario a BCD es un circuito que se emplea para la visualización de números en un display de 7 segmentos, consta de cuatro bits en la entrada los cuales generan 16 combinaciones posibles, a cada combinación se le asigna un valor específico de 8 bits que corresponde a los leds que se encienden en el display. La implementación de un decodificador binario a BCD utilizando compuertas de tipo circuito integrado TTL es complicada y además tardada, la función booleana es extensa y se usan mapas de Karnahug para optimizarla. Es necesario caracterizar el display de siete segmentos (distribución de segmentos) con el fin de saber si es ánodo común (enciende con ceros) o cátodo común (enciende con unos). Existen varias formas de implementar este circuito en VHDL, en este ejemplo la implementación se hará usando la sentencia case-when.

La descripción del decodificador binario a decimal es la siguiente:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- inputs and outputs statement.
entity decoder_bin_bcd is
port( n_binary: in std_logic_vector(3 downto 0);
      catode : out std_logic;
      n_decimal:out std_logic_vector(6 downto 0));
end decoder_bin_bcd;
architecture Behavioral of decoder_bin_bcd is
begin
catode <= '1';
process(n_binary) begin
case n_binary is
when "0000" => n_decimal <= "0111111"; -- 0
when "0001" => n_decimal <= "0000110"; -- 1
when "0010" => n_decimal <= "1011011"; -- 2
when "0011" => n_decimal <= "1001111"; -- 3
when "0100" => n_decimal <= "1100110"; -- 4
when "0101" => n_decimal <= "1101101"; -- 5
when "0110" => n_decimal <= "1111100"; -- 6
when "0111" => n_decimal <= "0000111"; -- 7
when "1000" => n_decimal <= "1111111"; -- 8
when "1001" => n_decimal <= "1101111"; -- 9
when others => n_decimal <= "0000000";
end case;
end process;
end Behavioral;
```

Después de realizar la síntesis y la corrección de errores de sintaxis, se tiene el análisis RTL

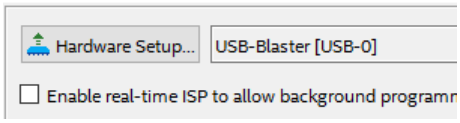


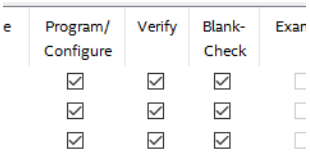
Nótese que el análisis RTL está hecho con multiplexores, el tema de multiplexores se explica más adelante. Esta es una de las maneras de implementar un decodificador binario a bcd en lenguaje VHDL.

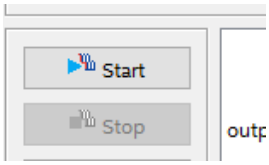
Agregando los puertos de entrada y salida

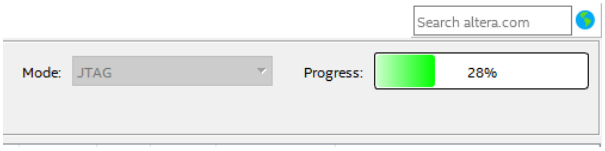
Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate Preservation
CATODE	Output			PIN_4	3.3-V ...fault)		16mA ...ault)	
N_BINARY[3]	Input	PIN_5	1	PIN_5	3.3-V LVTTL		16mA ...ault)	
N_BINARY[2]	Input	PIN_6	1	PIN_6	3.3-V LVTTL		16mA ...ault)	
N_BINARY[1]	Input	PIN_7	1	PIN_7	3.3-V LVTTL		16mA ...ault)	
N_BINARY[0]	Input	PIN_8	1	PIN_8	3.3-V LVTTL		16mA ...ault)	
N_DECIMAL[6]	Output	PIN_58	2	PIN_58	3.3-V LVTTL		16mA ...ault)	
N_DECIMAL[5]	Output	PIN_56	2	PIN_56	3.3-V LVTTL		16mA ...ault)	
N_DECIMAL[4]	Output	PIN_54	2	PIN_54	3.3-V LVTTL		16mA ...ault)	
N_DECIMAL[3]	Output	PIN_52	2	PIN_52	3.3-V LVTTL		16mA ...ault)	
N_DECIMAL[2]	Output	PIN_50	1	PIN_50	3.3-V LVTTL		16mA ...ault)	
N_DECIMAL[1]	Output	PIN_48	1	PIN_48	3.3-V LVTTL		16mA ...ault)	
N_DECIMAL[0]	Output	PIN_44	1	PIN_44	3.3-V LVTTL		16mA ...ault)	

Por último, se programa la tarjeta.

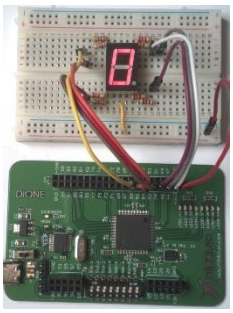


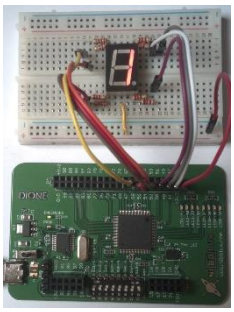


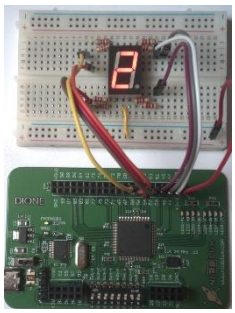


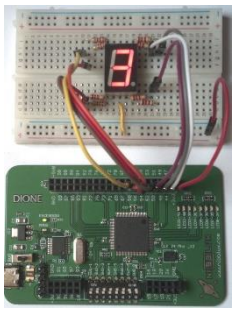


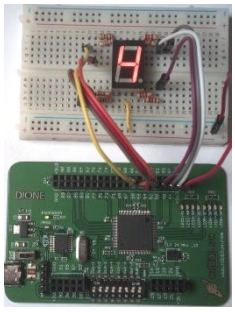
Ahora se verifica el correcto funcionamiento del decodificador.

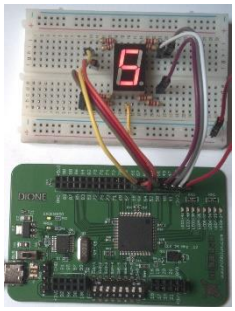


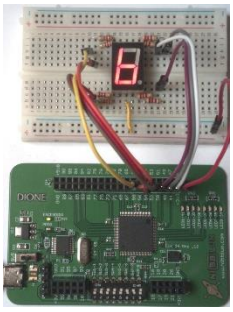


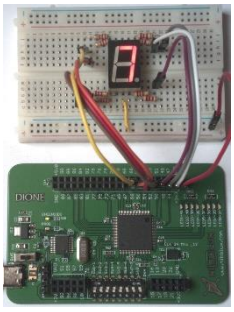


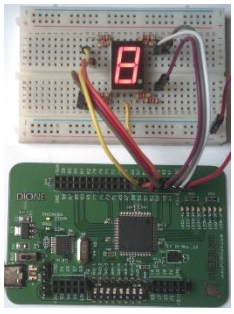


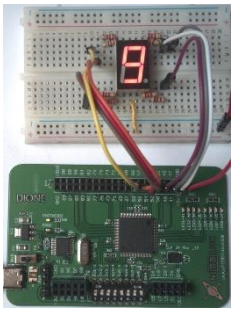










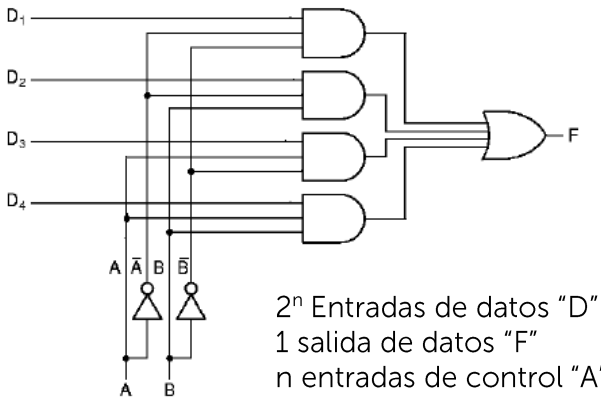


El decodificador funciona, muestra el valor de 0 a 9 en el display de 7 segmentos. Si se tiene el valor de 10 a 15 el decodificador no muestra ningún número.

Ejemplo 5 Multiplexor

Es un circuito lógico que acepta varias entradas de datos digitales y selecciona una de ellas en un momento dado para pasarla a la salida.
El enrutamiento de la entrada de datos deseada hacia la salida se controla mediante una entrada de selección también conocido como entradas de dirección.

La siguiente es una de las distintas formas de implementar un multiplexor, en el siguiente ejemplo se realizará un multiplexor 4 a 1 (4 entradas 1 salida). El diagrama lógico del multiplexor se muestra a continuación:



2^n Entradas de datos "D"
1 salida de datos "F"
n entradas de control "A" y "B"

Tabla de Verdad de Multiplexor

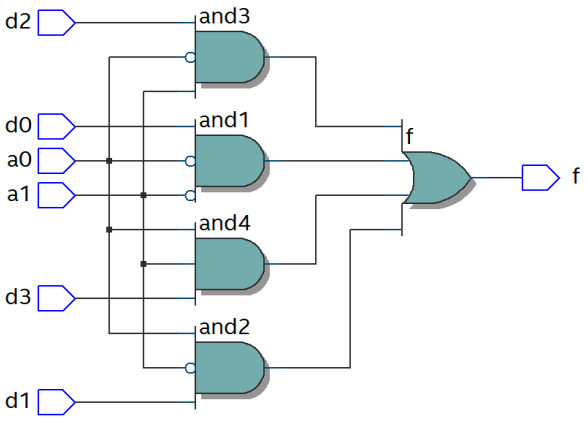
A1	A0	O0	O1	O2	O3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Tabla 5. Tabla de verdad del multiplexor

A continuación, se muestra la descripción del multiplexor de acuerdo con el diagrama lógico.

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   -- inputs and outputs statement.
4   entity mux4_1 is
5       Port ( a0 : in   STD_LOGIC;   -- input.
6              a1 : in   STD_LOGIC;   -- input.
7              d0 : in   STD_LOGIC;   -- input.
8              d1 : in   STD_LOGIC;   -- input.
9              d2 : in   STD_LOGIC;   -- input.
10             d3 : in   STD_LOGIC;   -- input.
11             f  : out  STD_LOGIC);  -- output.
12  end mux4_1;
13  architecture Behavioral of mux4_1 is
14  signal a0n,a1n,and1,and2,and3,and4 : std_logic := '0';
15  begin
16  a0n <= not(a0);                        -- The syntax for the a0n  output.
17  a1n <= not(a1);                        -- The syntax for the a1n  output.
18  and1 <= (a0n and a1n and d0);          -- The syntax for the and1 output.
19  and2 <= (a0 and a1n and d1);           -- The syntax for the and2 output.
20  and3 <=  (a0n and a1 and d2);          -- The syntax for the and3 output.
21  and4 <= (a0 and a1 and d3);            -- The syntax for the and4 output.
22  f <= (and1 or and2 or and3 or and4);   -- The syntax for the f    output.
23  end Behavioral;
```

Después de tener la descripción se debe hacer la síntesis y verificar que no hay problemas de síntesis. Hecho esto se realiza el análisis RTL para ver el diagrama lógico del multiplexor.



El diagrama RTL es el mismo que el diagrama lógico del multiplexor mostrado anteriormente. Ahora se tienen que agregar las entradas y salidas correspondientes:

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Streng	ict Preservati
in A0	Input	PIN_2	1	PIN_2	3.3-V LVTTL		16mA ...ault)	
in A1	Input	PIN_1	2	PIN_1	3.3-V LVTTL		16mA ...ault)	
in D0	Input	PIN_8	1	PIN_8	3.3-V LVTTL		16mA ...ault)	
in D1	Input	PIN_7	1	PIN_7	3.3-V LVTTL		16mA ...ault)	
in D2	Input	PIN_6	1	PIN_6	3.3-V LVTTL		16mA ...ault)	
in D3	Input	PIN_5	1	PIN_5	3.3-V LVTTL		16mA ...ault)	
out F	Output	PIN_41	1	PIN_41	3.3-V LVTTL		16mA ...ault)	

Para finalizar se programa la tarjeta y se verifica el funcionamiento del multiplexor, para esto, a las entradas d0, d1, d2 y d3, se les asigna el valor fijo 1010 respectivamente, al cambiar los valores de a0 y a1, la salida f toma el valor de la entrada d seleccionada.

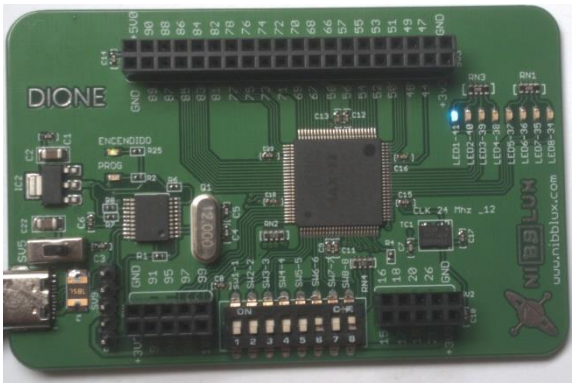


Figura 5.1. D0D1D2D3=1010, A0A1=00, f=1.

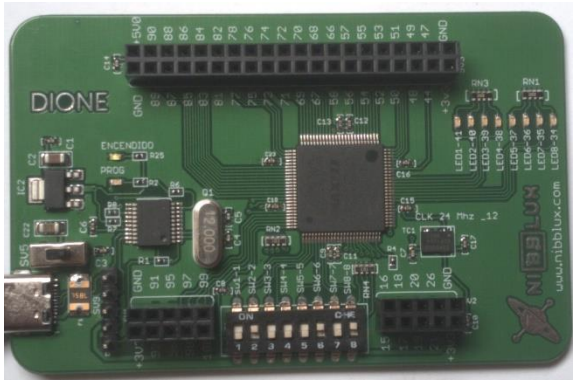


Figura 5.2. D0D1D2D3=1010, A0A1=10, f=0.

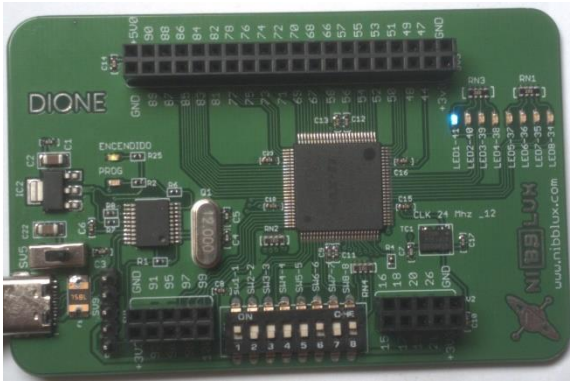


Figura 5.3. D0D1D2D3=1010, A0A1=01, f=1.

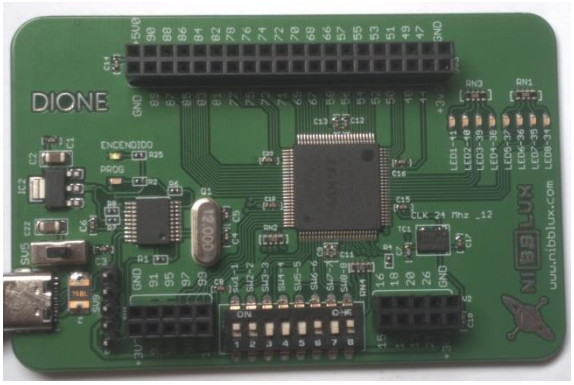


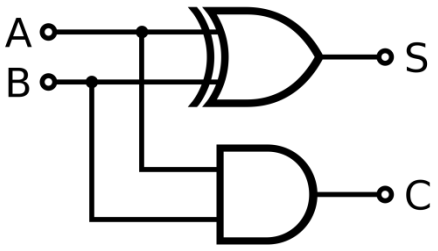
Figura 5.4. D0D1D2D3=1010, A0A1=11, f=0.

Ejemplo 6 Medio Sumador.

El medio sumador suma dos dígitos binarios simples A y B, denominados sumandos, y sus salidas son Suma (S) y Acarreo (C). La señal de acarreo representa un desbordamiento en el siguiente dígito en una adición de varios dígitos. El diseño más simple de medio sumador, representado a continuación, incorpora una puerta XOR para S y una puerta AND para C. Dos semisumadores pueden ser combinados para hacer un sumador completo, añadiendo una puerta OR para combinar sus salidas de acarreo. A Continuación se muestran el diagrama lógico y la tabla de verdad de este circuito.

Entradas		Salidas	
A	***B***	***C***	***S***
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

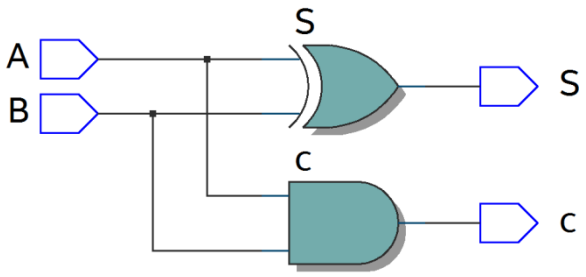
Tabla 6. Tabla de verdad para el medio sumador.



En medio sumador está construido por dos compuertas, una xor y una and.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- inputs and outputs statemend
entity half_adder is
    Port ( A : in  STD_LOGIC;    -- input
           B : in  STD_LOGIC;    -- input
           S : out std_logic;    -- Sum output
           c : out  STD_LOGIC); -- Carry output
end half_adder;

architecture Behavioral of half_adder is

begin

  C <= A and B; -- Carry
  S <= A xor B; -- Sum

end Behavioral;
```

Con base en el diagrama lógico anterior, se describe a continuación un medio sumador en lenguaje VHDL.



Al realizar la síntesis y verificar que no hay errores se genera el análisis RTL.

Se agregan los puertos de entrada y salida.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	urrent Streng	ict Preservati
A	Input	PIN_1	2	PIN_1	3.3-V LVTTL		16mA ...ault)	
B	Input	PIN_2	1	PIN_2	3.3-V LVTTL		16mA ...ault)	
C	Output	PIN_41	1	PIN_41	3.3-V LVTTL		16mA ...ault)	
S	Output	PIN_39	1	PIN_39	3.3-V LVTTL		16mA ...ault)	

Para terminar, se programa la tarjeta y se comprueba la tabla de verdad del medio sumador.

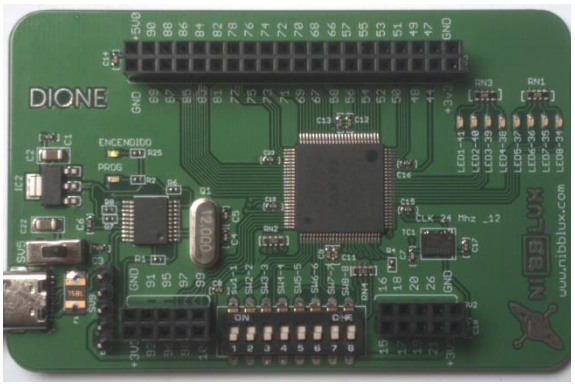


Figura 6.1. AB=00, CS=00.

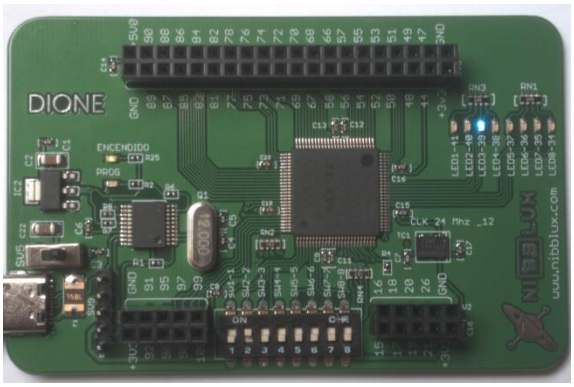


Figura 6.2. AB=10, CS=01.

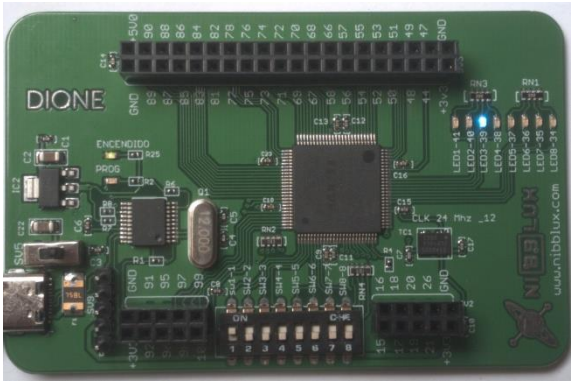


Figura 6.3. AB=01, CS=01.

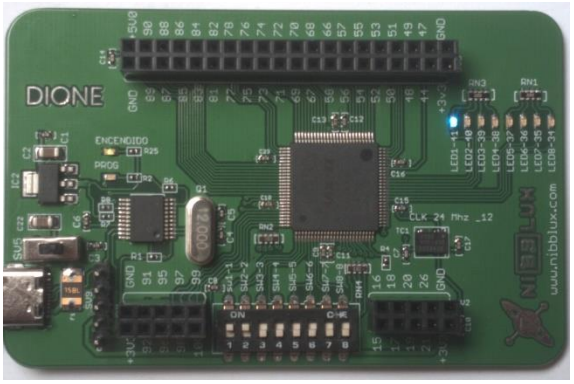


Figura 6.4. AB=11, CS=10

Ejemplo 7 Sumador Completo.

Un sumador completo suma números binarios junto con las cantidades de acarreo. Un sumador completo añade tres bits, a menudo escritos como A, B y Cin siendo A y B los sumandos y Cin el acarreo que proviene de la etapa anterior menos significativa. El sumador completo es un componente construido con la conexión en cascada de medios sumadores, que suma pares de números binarios de diferente cantidad de bits. El circuito produce una salida de dos bits, al igual que el medio sumador, denominadas acarreo de salida (Cout) y suma (S).

Un sumador completo se puede implementar de muchas maneras diferentes, tales como con un circuito a transistores o compuesto de compuertas. Un ejemplo de implementación es expresado con las siguientes ecuaciones:

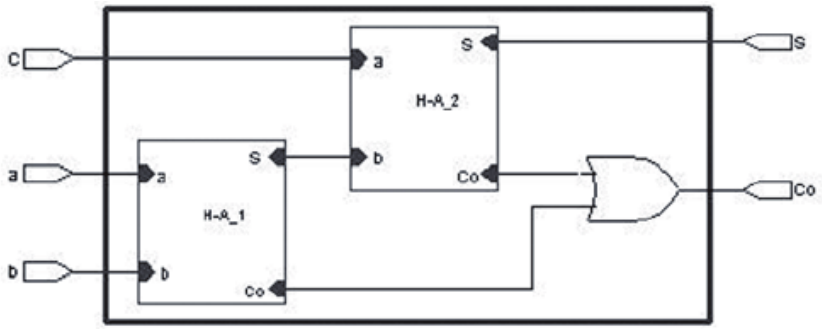
$$\begin{cases} S & = & A \oplus B \oplus C_{in} \\ C_{out} & = & (A \cdot B) + C_{in} \cdot (A + B) \end{cases}$$

La tabla de verdad del sumador completo es la siguiente:

Entradas			Salidas	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tabla 7.

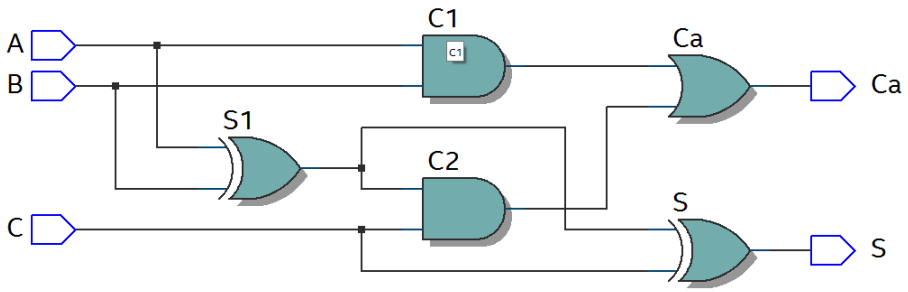
En esta descripción se utilizaron dos medios sumadores para crear un sumador completo como se muestra en la figura



La descripción de VHDL es la siguiente:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- inputs and outputs statemend
entity full_adder is
    Port ( A : in  STD_LOGIC;    -- input
           B : in  STD_LOGIC;    -- input
           C : in  STD_LOGIC;    -- input
           S : out std_logic;    -- Sum output
           Ca : out  STD_LOGIC); -- Carry output
end full_adder;

architecture Behavioral of full_adder is
signal C1,S1,C2 : std_logic := '0';
begin

  C1 <= A and B;    -- Carry of first half adder
  S1 <= A xor B;    -- Sum   f first half adder
  S  <=  C xor S1; -- Sum   of full adder
  C2 <= C and S1;   -- Carry of second half adder
  Ca  <= C1 or C2; -- Carry of full adder

end Behavioral;
```

Al realizar la síntesis y verificar que no hay errores se tiene el diagrama lógico que se obtiene del análisis RTL.



Se agregan los puertos de entrada y salida

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Streng	ict Preservati
in A	Input	PIN_1	2	PIN_1	3.3-V LVTTL		16mA ...ault)	
in B	Input	PIN_2	1	PIN_2	3.3-V LVTTL		16mA ...ault)	
in C	Input	PIN_3	1	PIN_3	3.3-V LVTTL		16mA ...ault)	
out CA	Output	PIN_41	1	PIN_41	3.3-V LVTTL		16mA ...ault)	
out S	Output	PIN_39	1	PIN_39	3.3-V LVTTL		16mA ...ault)	

Para finalizar se tiene la comprobación del sumador completo.

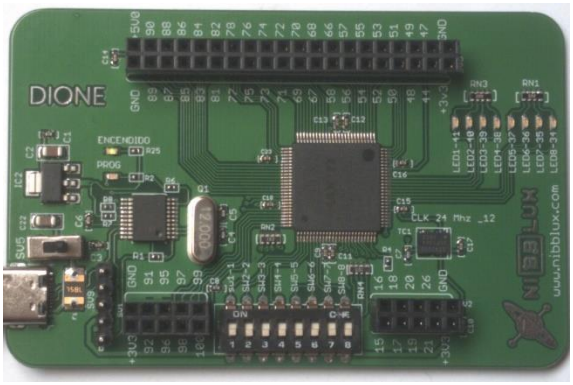


Figura 7.1. ABC=000, C_a S=00.

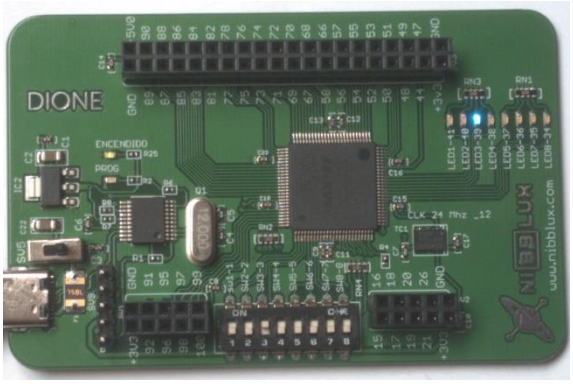


Figura 7.2. ABC=001, C_a S=01.

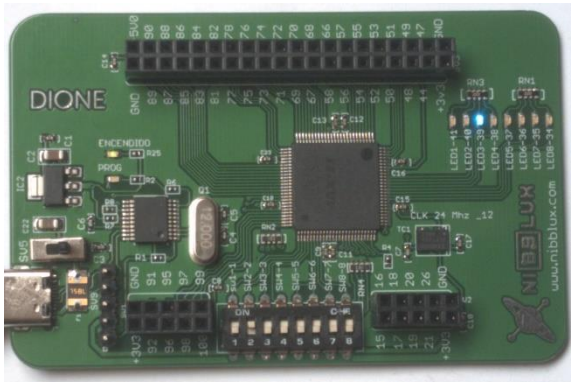


Figura 7.3. ABC=010, C_a S=01.

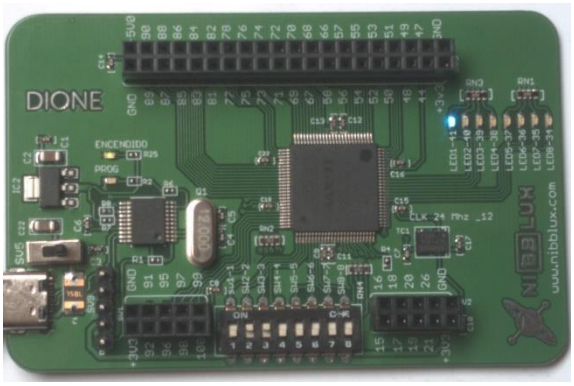


Figura 7.4. ABC=011, C_a S=10.

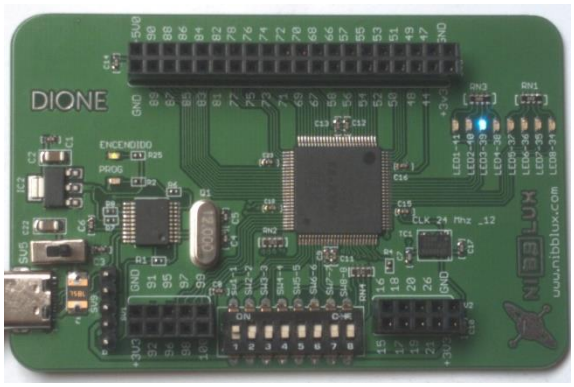


Figura 7.5. ABC=100, C_a S=01.

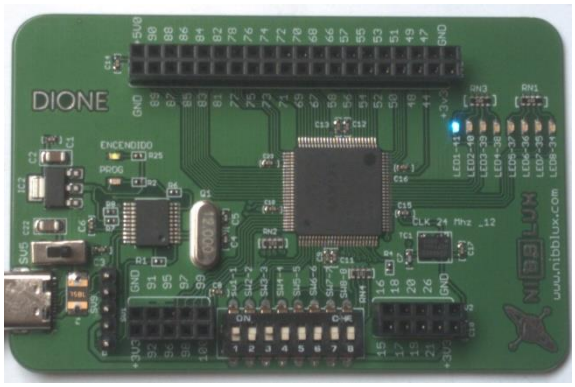


Figura 7.6. ABC=101, C_a S=10.

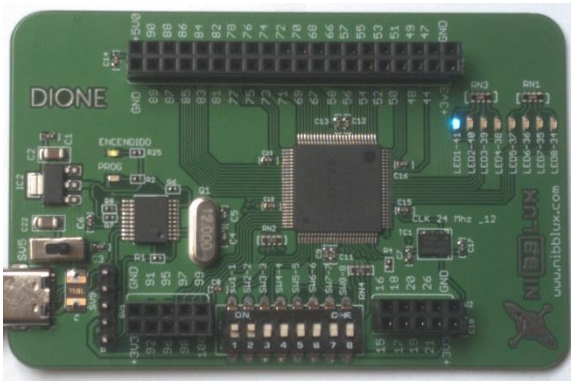


Figura 7.7. ABC=110, C_a S=10.

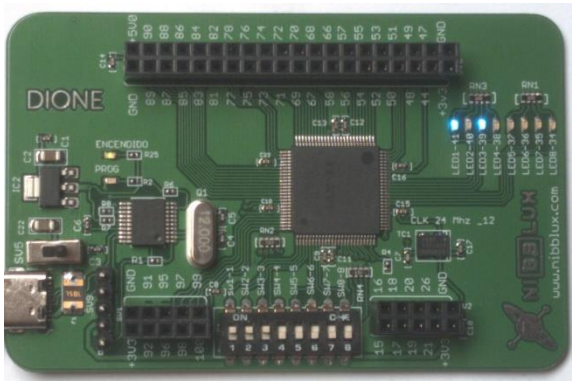
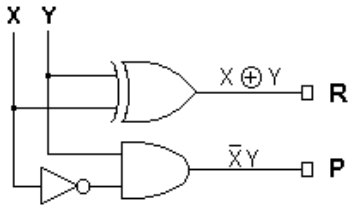


Figura 7.8. ABC=111, C_a S=11.

Ejemplo 8 Medio restador.

Un medio restador es un circuito combinacional que sustrae dos bits y produce su diferencia. También tiene una salida para especificar si se ha tomado un 1. Se designa el bit minuendo por x y el bit sustraendo mediante y. Para llevar a cabo x – y, tienen que verificarse las magnitudes relativas de x y y. Si x >= y, se tienen tres posibilidades; 0 - 0 = 0, 1 – 0 = 1 y, 1 - 1 = 0. El resultado se denomina bit de diferencia.
Si x < y, tenemos 0 – 1 y es necesario tomar un 1 de la siguiente etapa más alta. El 1 que se toma de la siguiente etapa más alta añade dos al bit minuendo, de la misma forma que en el sistema decimal lo que se toma añade 10 a un dígito minuendo. Con el minuendo igual a 2, la diferencia llega a ser 2 – 1 = 1. El medio restador requiere dos salidas. Una salida genera la diferencia y se denotará por el símbolo D. La segunda salida, denotada B es para la cantidad que se toma, genera la señal binaria que informa a la siguiente etapa que se ha tomado un 1. La tabla de verdad para las relaciones de entrada-salida de un medio restador ahora puede derivarse como sigue:

Su diagrama lógico y la tabla de verdad son las siguientes:



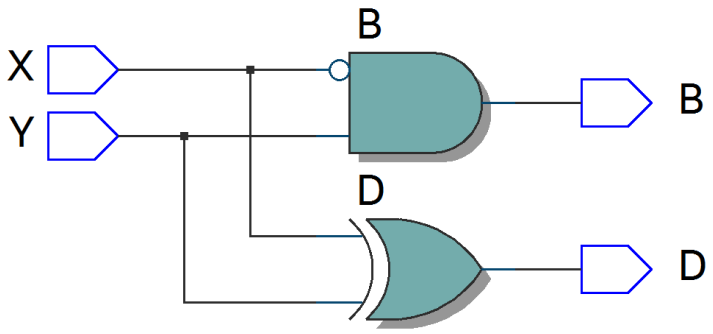
Truth Table			
Y	X	Diff.	Borrow
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	0

Tabla 8. Tabla de verdad del medio restador.

Con el diagrama lógico la descripción es la siguiente:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- inputs and outputs statemend
entity half_subtractor is
    Port ( X : in  STD_LOGIC;    -- input
           Y : in  STD_LOGIC;    -- input
           D : out std_logic;    -- difference output
           B : out  STD_LOGIC); -- Borrow     output
end half_subtractor;

architecture Behavioral of half_subtractor is
signal not1 : std_logic := '0';

begin

  not1 <= not(X);  -- not1 is a signal for xor gate
  B <= not1 and Y; -- Borrow
  D <= X xor Y;    -- Difference

end Behavioral;
```

El análisis RTL de la descripción anterior muestra lo siguiente:



Se agregan las entras y salidas.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	Current Streng	ict Preservati
B	Output	PIN_39	1	PIN_39	3.3-V LVTTL		16mA ...ault)	
D	Output	PIN_41	1	PIN_41	3.3-V LVTTL		16mA ...ault)	
X	Input	PIN_2	1	PIN_2	3.3-V LVTTL		16mA ...ault)	
Y	Input	PIN_1	2	PIN_1	3.3-V LVTTL		16mA ...ault)	

Se vuelve a compilar el programa y se le carga a la tarjeta DIONE. Después se verifica que funciona correctamente.

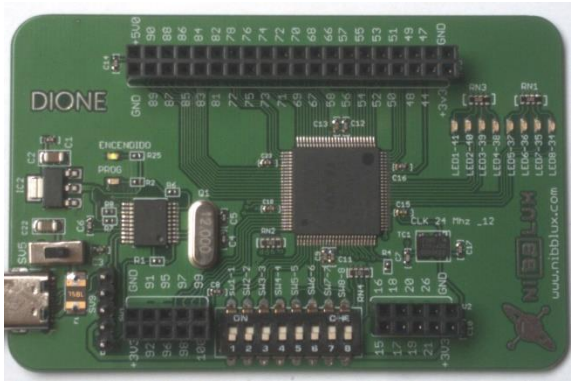


Figura 8.1. XY=00, BD=01.

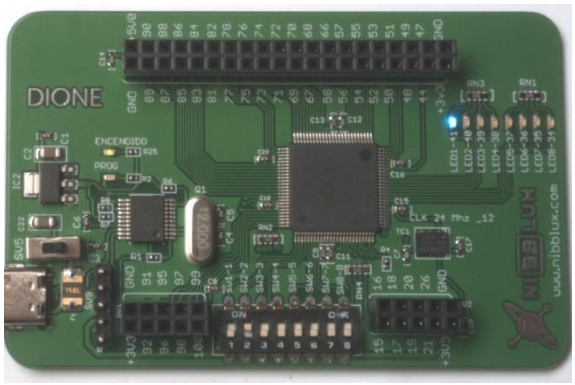


Figura 8.2. XY=10, BD=01.

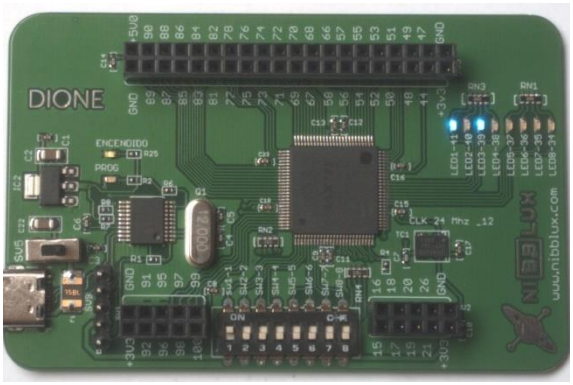


Figura 8.3. XY=01, BD=11.

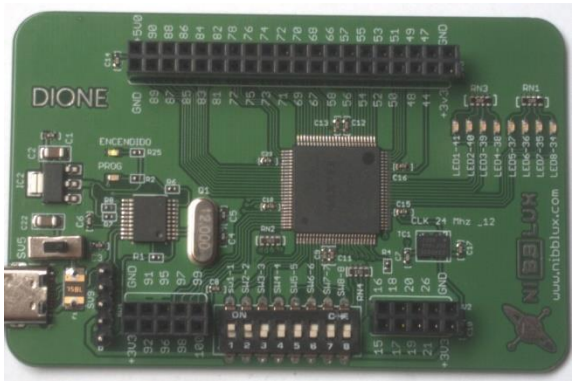


Figura 8.4. XY=11, BD=00.

Ejemplo 9 Restador completo.

Un restador completo es un circuito combinacional que lleva a cabo una sustracción entre dos bits, tomando en cuenta que uno de esos dos bits, ha tomado prestada una unidad del siguiente bit más significativo. Este circuito tiene tres entradas y dos salidas. Las tres entradas x, y, z, denotan al minuendo, sustraendo y a la toma previa, respectivamente. Las dos salidas, D y B, representan la diferencia y el bit que indica que se tomó prestada una unidad al siguiente bit más significativo, respectivamente.

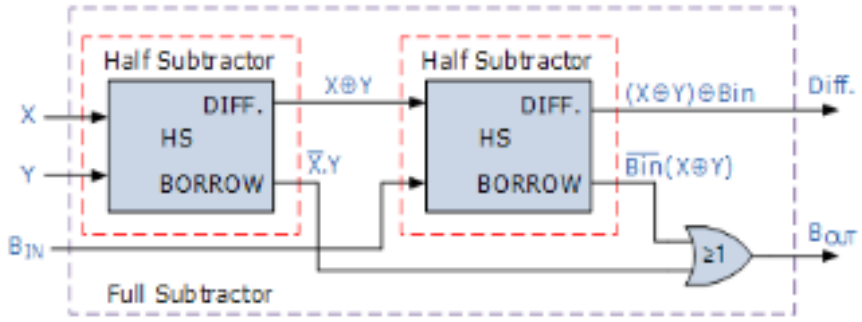
La tabla de verdad para el circuito es como sigue:

Truth Table				
B-in	Y	X	Diff.	B-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

Los ocho renglones bajo las variables de entrada designan todas las combinaciones posibles de 1 y 0 que pueden tomar las variables binarias. Los 1 y 0 para las variables de salida están determinados por la sustracción de x – y – z. Las combinaciones que tienen salida de toma z = 0 se reducen a las mismas cuatro condiciones del medio sumador.

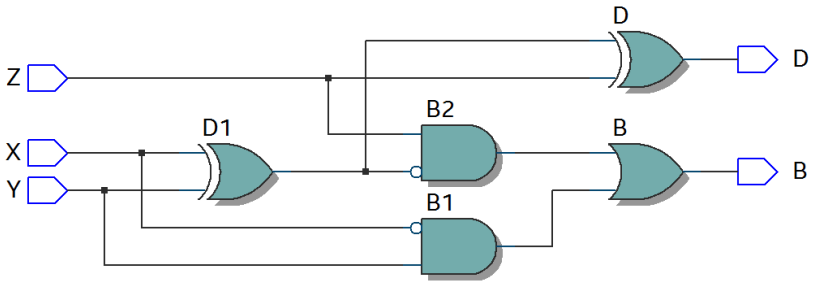
Para x = 0, y = 0 y z = 1, tiene que tomarse un 1 de la siguiente etapa, lo cual hace B = 1 y añade 2 a x. Ya que 2 – 0 – 1, D = 1. Para x = 0 y yz = 11, necesita tomarse otra vez, haciendo B = 1 y x = 2. Ya que 2 – 1 – 1 = 0, D = 0. Para x = 1 y yz = 01, se tiene x – y – z = 0, lo cual hace B = 0 y D = 0. Por último, para x = 1, y = 1, z = 1, tiene que tomarse 1, haciendo B = 1 y x = 3 y, 3 –1 – 1 = 1, haciendo D = 1.

En la siguiente figura se muestra el diagrama a bloques del restador completo.



La descripción del restador completa es la siguiente:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--inputs and outputs statment.
entity full_subtractor is
   port ( X : in STD_LOGIC;
          Y : in STD_LOGIC;
          Z : in STD_LOGIC;
          D : out STD_LOGIC;
          B : out STD_LOGIC);
end full_subtractor;

architecture behabioral of full_subtractor is
signal not1, not2, B1, B2, D1 : std_logic := '0';

begin

      not1  <= not(x);      --not1 is a signal for and gate one.
      B1    <= not1 and Y; --Borrow B1 is the output of first half subtractor.
      D1    <= X xor Y;    --Difference D1 is the output of first half subtractor.
      not2  <= not(D1);     --not2 is a signal for and gate two.
      B2    <= not2 and Z; --Borrow B2 is the output of second half subtractor.
      D     <= D1 xor Z;   --Difference D2 is the output of full subtractor.
      B     <= B1 or B2;   --Borrow B is the output of full subtractor.


end behabioral;
```



El análisis RTL es el siguiente, muestra el diagrama lógico del restador completo.

Se agregan los puertos de entras y salidas correspondientes.

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Reserved	urrent Streng	ict Preservati
out B	Output	PIN_39	1	PIN_39	3.3-V LVTTL		16mA ...ault)	
out D	Output	PIN_41	1	PIN_41	3.3-V LVTTL		16mA ...ault)	
in X	Input	PIN_3	1	PIN_3	3.3-V LVTTL		16mA ...ault)	
in Y	Input	PIN_2	1	PIN_2	3.3-V LVTTL		16mA ...ault)	
in Z	Input	PIN_1	2	PIN_1	3.3-V LVTTL		16mA ...ault)	

Ahora se carga el archivo a la tarjeta DIONE y se verifica el funcionamiento del código.

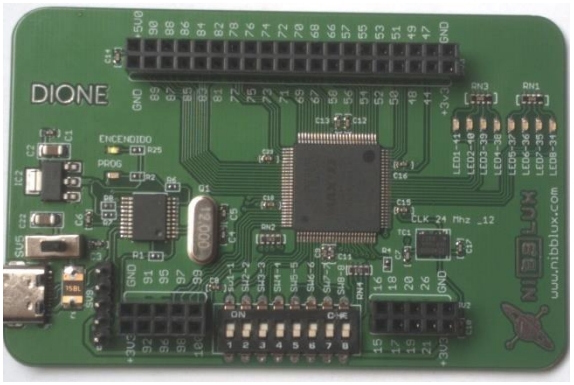


Figura 9.1. ZYX=000, DB=00.

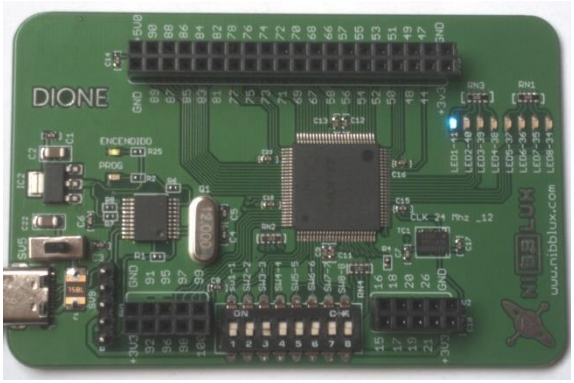


Figura 9.2. ZYX=001, DB=10.

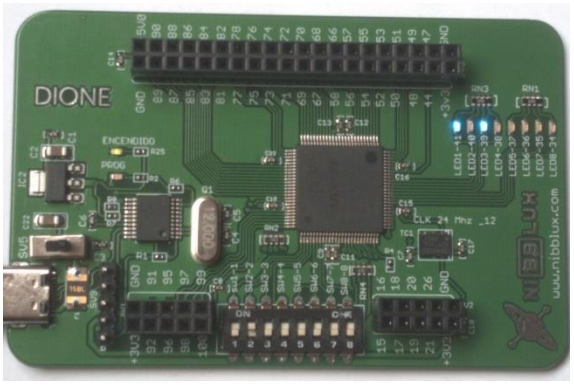


Figura 9.3. ZYX=010, DB=11.

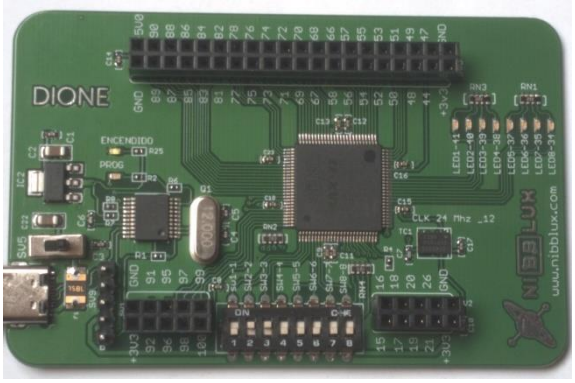


Figura 9.4. ZYX=011, DB=00.

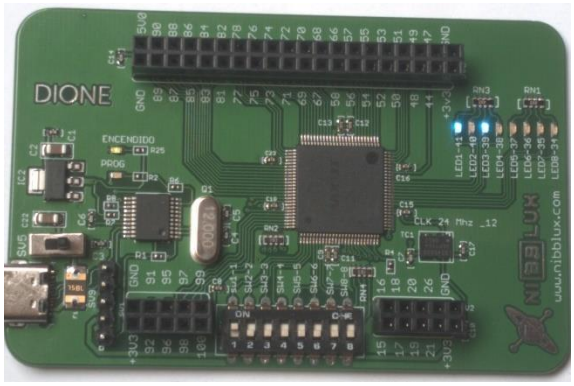


Figura 9.5. ZYX=100, BD=11.

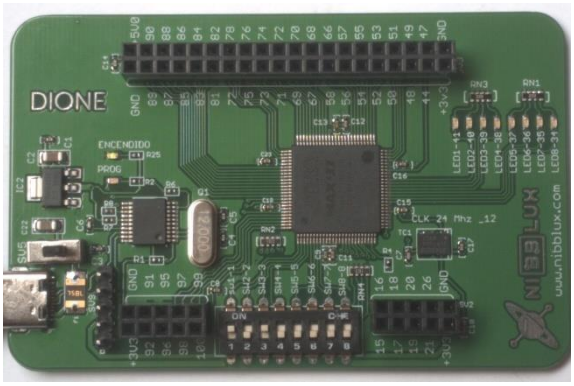


Figura 9.6. ZYX=101, DB=00.

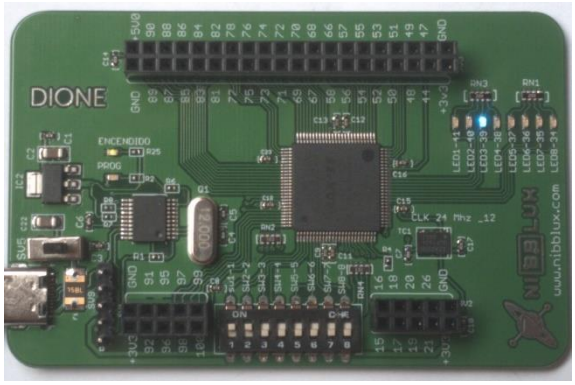


Figura 9.7. ZYX=110, DB=01.

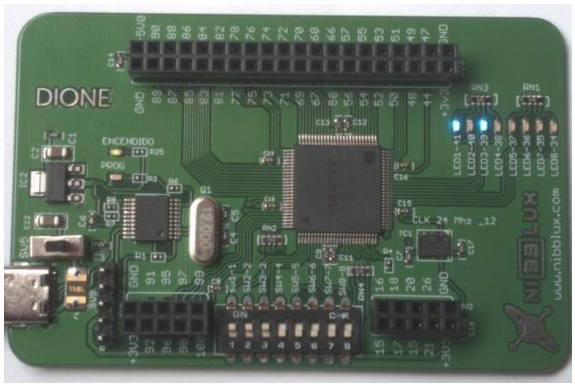


Figura 9.1. ZYX=111, DB=11.

Referencias

MAX II Device Hand Book, Altera.

Historial de revisión

Fecha y revisión	Revisó	Cambios realizados
24/03/2019	Ing. Ricardo Alberto Villegas Pantoja. Ing. David Alonso Tapia Abrego.	Primera publicación
03/10/2019	Ing. José Luis Molina Olivera.	Correcciones y redacción general.

Elaborado por:

Ing. Rodrigo Ismael Barrera Bernabel.
Ing. José Luis Molina Olivera.
Ing. Carlos Edoardo Manzano Fragoso.
Ing. David Alonso Tapia Abrego.