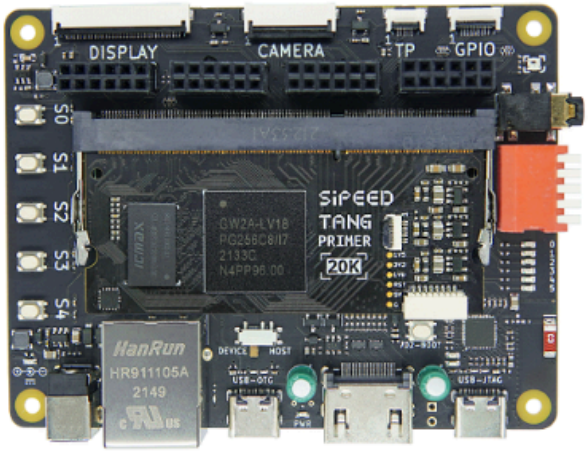
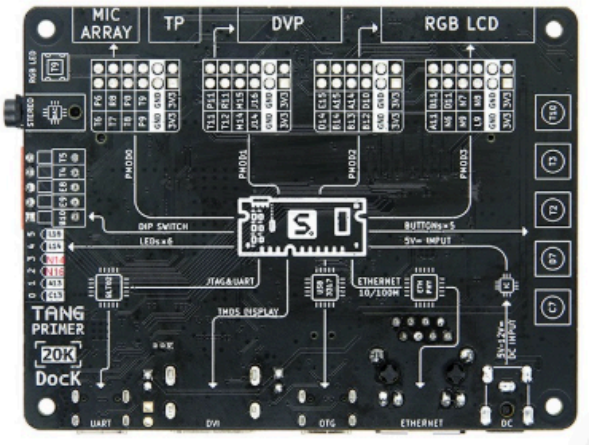
Documentación: Instalación y Programación de la FPGA **Tang Primer 20K**

Tabla de contenido

1. Materiales necesarios

- Placa Tang Primer 20K Core Board
- Dock / Ext board (opcional)
- Cable USB-JTAG para programación
- PC con Windows o Linux
- Gowin IDE (herramienta oficial)
- (Opcional) OpenFPGALoader en Linux para programar



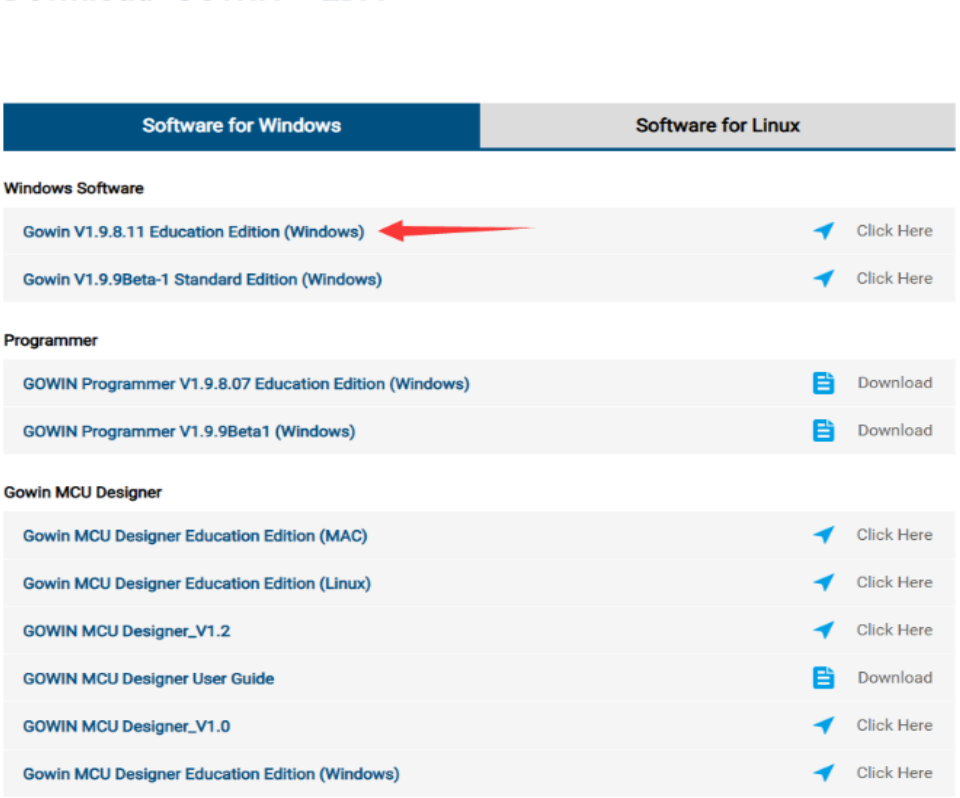


2. Instalación del entorno de desarrollo (Gowin IDE + Programmer)

El fabricante recomienda usar la Gowin IDE, porque la FPGA de Tang Primer 20K es un chip de Gowin.

Windows (recomendado para empezar)

1. Ve a la web oficial *GOWIN EDA* → *Download* y descarga la versión para Windows (Education o Standard según necesidad).



2. Ejecuta el instalador **`.exe`** y sigue los pasos (suele instalar también el *GOWIN Programmer* o pedir instalación separada del programmer). Si la instalación solicita servidor de licencias, la *Education* suele funcionar sin licencia.
3. Conecta la placa vía USB y comprueba que Windows la reconoce (puerto COM para UART y/o adaptador FTDI si corresponde).

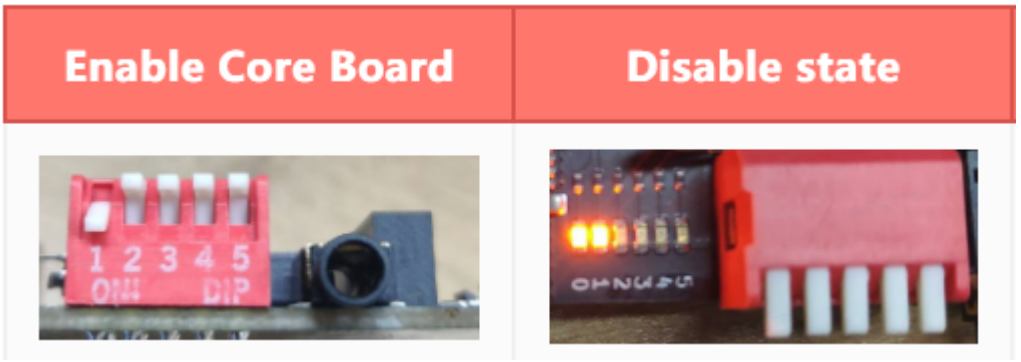
Linux (Ubuntu / Debian)

1. Descarga la versión Linux de Gowin o usa los paquetes publicados. Si el *Programmer* oficial no detecta la placa, se recomienda usar **openFPGALoader** (instalable con **`apt`** en algunas distros o compilando desde fuente).
2. Para instalar openFPGALoader (ejemplo en Ubuntu):

```
sudo apt update
sudo apt install openfpgaloader
```

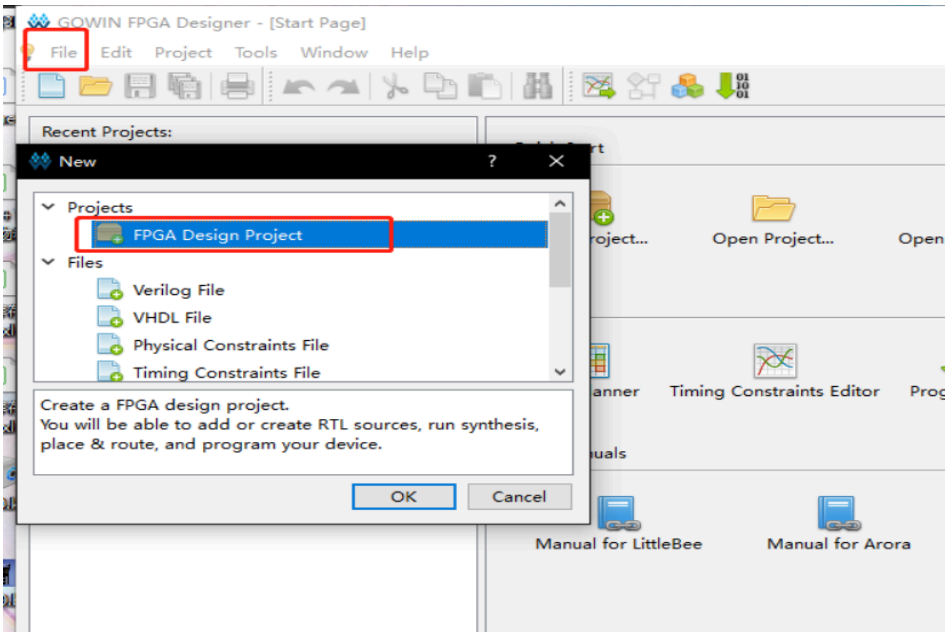
   Usa **`openFPGALoader --detect`** para comprobar detección.

3. **Preparar la placa (hardware)**
    - Asegúrate de que el DIP-switch/enable de la Dock esté en la posición correcta para permitir la programación (algunas versiones requieren mover un switch durante el proceso). Si la placa arranca con una firmware de test (DDR test) puede calentarse: puedes sobrescribir/erasear ese firmware.
    - Es necesario habilitar la placa base antes de usar el depurador para depurar el chip, simplemente coloque el interruptor 1 en el interruptor DIP hacia abajo, de lo contrario, el LED O y el LED 1 están encendidos y la placa base no funciona.

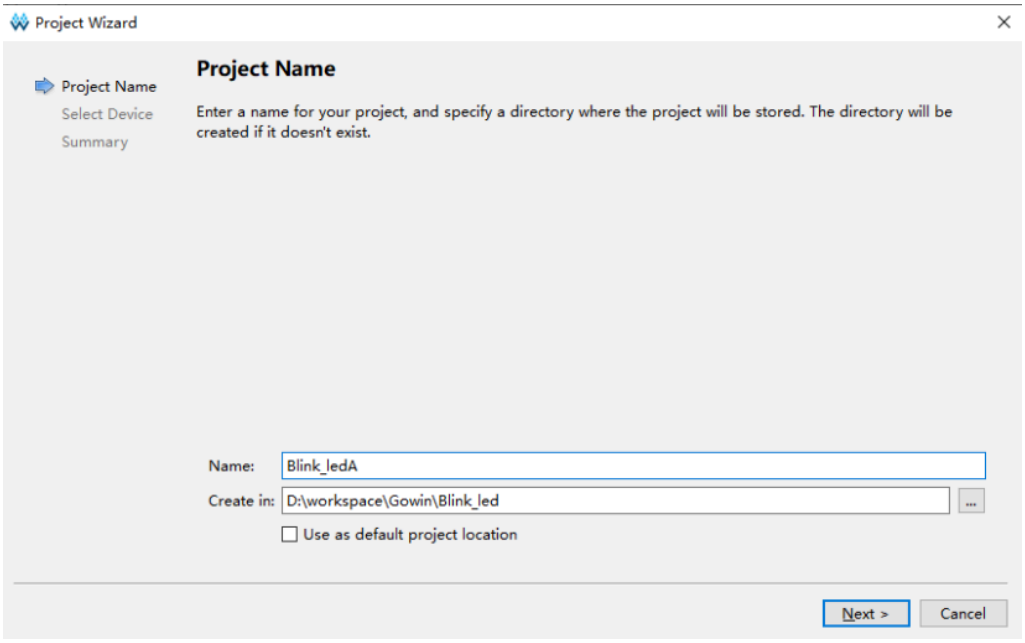


    - Evita hubs USB al depurar JTAG; conecta directo al puerto USB del PC. (OpenFPGALoader y guías recomiendan evitar hubs).

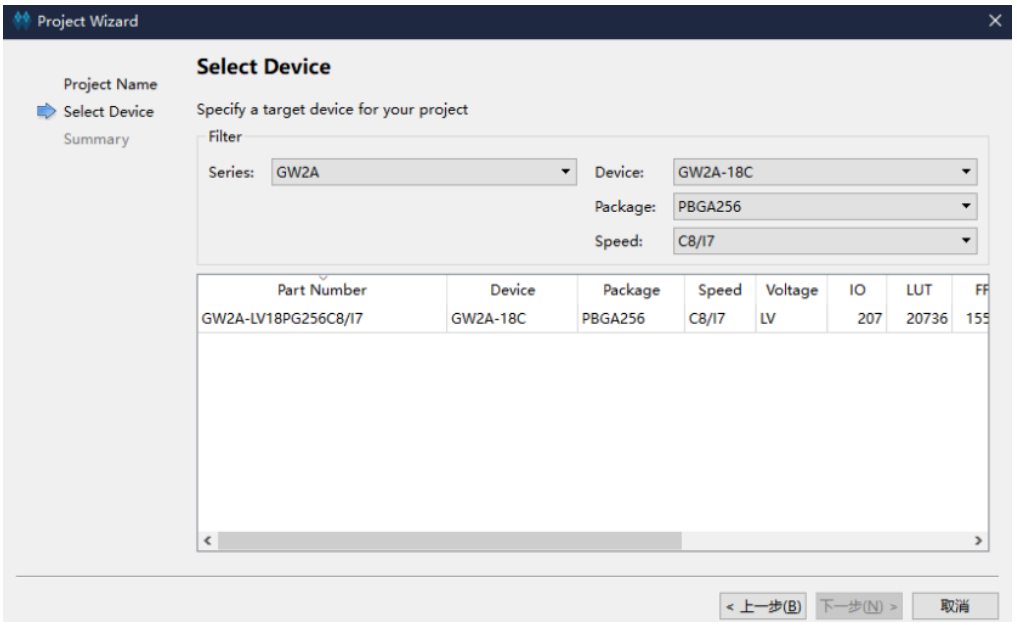
4. **Crear un proyecto nuevo en Gowin IDE**
    1. Abrir Gowin IDE → **`File`** → **`New`** → **`FPGA Design Project`**.



2. Rellenar nombre y ruta del proyecto (usa rutas sin espacios ni caracteres especiales).



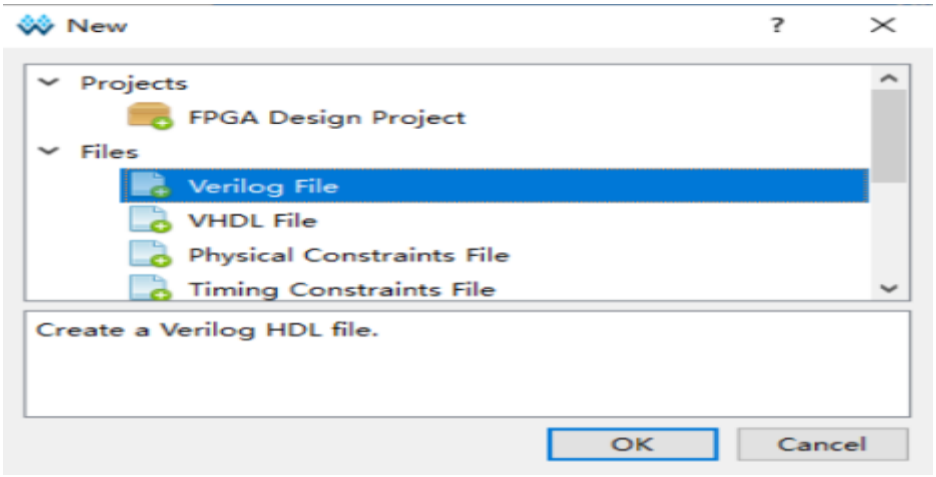
3. Seleccionar el **device** correcto: **GW2A-LV18PG256C8/I7** (usa el filtro de dispositivos para encontrarlo). Esto es importante porque las síntesis/PNR dependen del paquete.



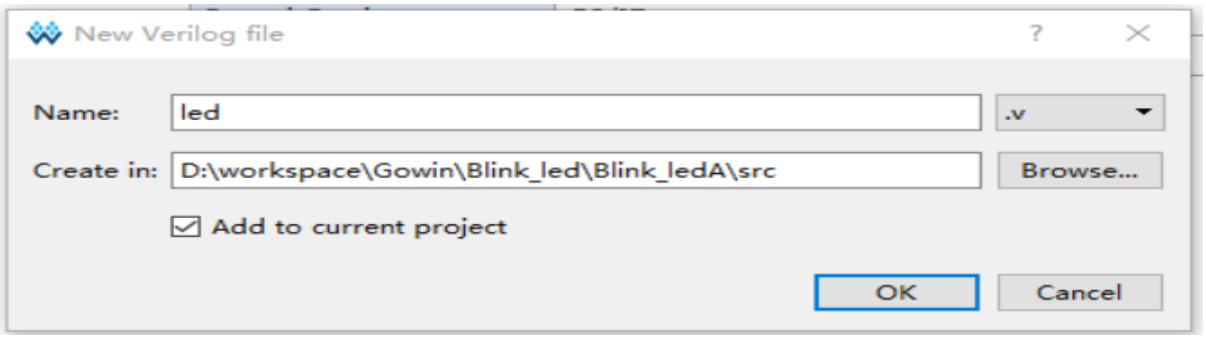
4. Confirmar y crear proyecto.

4. Escribir el código Verilog (Blink LED)

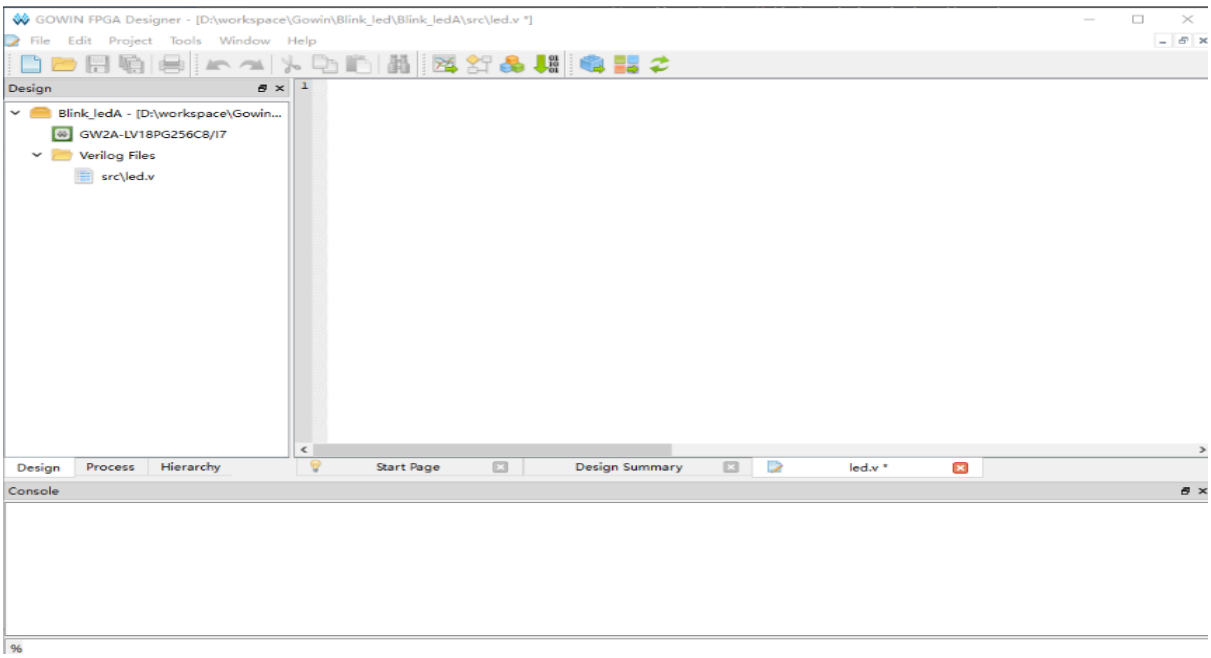
Gowin IDE contiene 3 formas de crear un archivo. Aquí utilizamos teclas de acceso directo **Ctrl + N** para crear un nuevo archivo. Las otras 2 formas de crear un nuevo archivo no se mencionan aquí. En las ventanas emergentes, elegimos **Verilog File**, también puedes elegir **VHDL File** si eres bueno en eso. Aquí simplemente usamos Verilog como ejemplo.



Luego haga clic en Aceptar para configurar el nombre del archivo, aquí lo tomamos **led** como el nombre del archivo verilog como ejemplo.



Luego necesitamos preparar nuestro código.



Introducción a Verilog

Verilog es un tipo de lenguaje de descripción de hardware (HDL), se utiliza para describir circuitos digitales. La unidad básica de Verilog es el módulo. Un módulo se compone de dos partes: una describe la interfaz y la otra describe la función lógica interna, es decir, define cómo la entrada afecta a la salida. Un módulo es así:

```
1  module module_name
2  #(parameter)
3  (port) ;
4      function
5  endmodule
```

El módulo comienza desde el módulo y finaliza con el módulo final. Al módulo le sigue el nombre del módulo (module_name), los parámetros de la variable transitable (parámetro), la declaración de puerto y dirección (puerto), seguido de la descripción de la función lógica interna (función) y, finalmente, se utiliza el módulo final para representar este módulo.La función lógica interna generalmente está compuesta por los bloques asignar y siempre; la declaración asignar describe el circuito lógico y el bloque siempre se utiliza para describir el circuito de temporización.

Descripción del código

Podemos ver que el módulo que crearemos contiene 2 puertos:

```
1  module led(
2      input  Clock,
3      output IO_voltage
4  );
5
6  endmodule
```

Para el contador de tiempo dentro del módulo, el oscilador de cristal en la placa central Primer 20K es de 27 MHZ, por lo que tenemos 27 millones de veces los bordes ascendentes por segundo y solo necesitamos contar 13500000 veces los bordes ascendentes para obtener 0,5 segundos. El contador comienza desde **0**, y para contar 13500000 veces los bordes ascendentes, contamos hasta 13499999. Cuando se cuenta a 0,5 S, configuramos un indicador para informar al LED IO que invierta su voltaje. El código de recuento general es el siguiente:

```
//parameter Clock_frequency = 27_000_000; // Crystal oscillator frequency i
parameter count_value       = 13_499_999; // The number of times needed to

reg [23:0]  count_value_reg ; // counter_value
reg         count_value_flag; // IO change flag

always @(posedge Clock) begin
    if ( count_value_reg <= count_value ) begin //not count to 0.5S
        count_value_reg  <= count_value_reg + 1'b1; // Continue counting
        count_value_flag <= 1'b0 ; // No flip flag
    end
    else begin //Count to 0.5S
        count_value_reg  <= 23'b0; // Clear counter,prepare for next time c
        count_value_flag <= 1'b1 ; // Flip flag
    end
end
```

El código para cambiar el voltaje IO es el siguiente:

```
reg IO_voltage_reg = 1'b0; // Initial state

always @(posedge Clock) begin
    if ( count_value_flag )  //  Flip flag
        IO_voltage_reg <= ~IO_voltage_reg; // IO voltage flip
    else //  No flip flag
        IO_voltage_reg <= IO_voltage_reg; // IO voltage constant
end
```

Combinado con los códigos

```
module led(
    input  Clock,
    output IO_voltage
);

/********** Counter **********/
//parameter Clock_frequency = 27_000_000; // Crystal oscillator frequency
parameter count_value       = 13_499_999; // The number of times needed t

reg [23:0]  count_value_reg ; // counter_value
reg         count_value_flag; // IO chaneg flag

always @(posedge Clock) begin
    if ( count_value_reg <= count_value ) begin //not count to 0.5S
        count_value_reg  <= count_value_reg + 1'b1; // Continue counting
        count_value_flag <= 1'b0 ; // No flip flag
    end
    else begin //Count to 0.5S
        count_value_reg  <= 23'b0; // Clear counter,prepare for next time
        count_value_flag <= 1'b1 ; // Flip flag
    end
end

/********** IO voltage flip **********/
reg IO_voltage_reg = 1'b0; // Initial state

always @(posedge Clock) begin
    if ( count_value_flag )  //  Flip flag
        IO_voltage_reg <= ~IO_voltage_reg; // IO voltage flip
    else //  No flip flag
        IO_voltage_reg <= IO_voltage_reg; // IO voltage constant
end

/***** Add an extra line of code *****/
assign IO_voltage = IO_voltage_reg;

endmodule
```

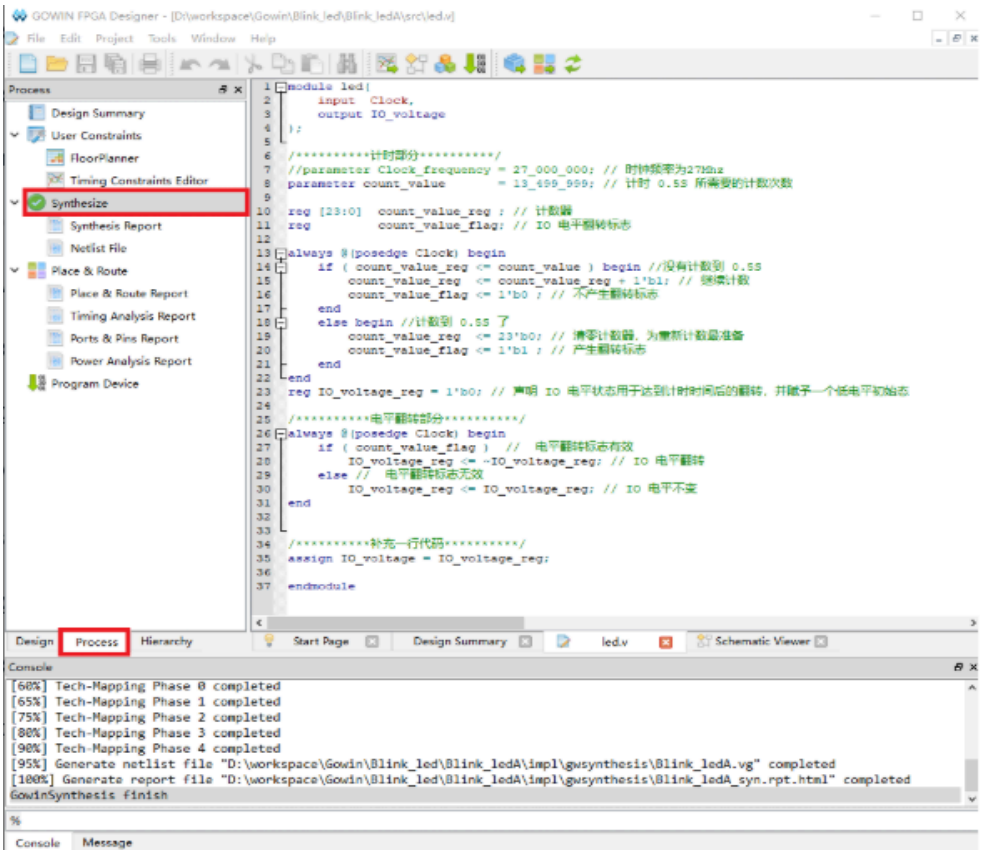
Porque el **IO_voltage** se declara en la posición del puerto, que es de tipo cable de forma predeterminada. Para conectarlo a la variable reg **IO_voltage_reg**, necesitamos usar asignar.

5. **Síntesis, restricciones, Place & Route**

Puedes asignar pines de 2 formas: GUI (FloorPlanner) o creando un archivo .cst (constraints). Las herramientas Gowin soportan la sintaxis CST / IO_LOC para fijar pines. Recomendación: usa FloorPlanner si eres nuevo, y exporta la CST luego.

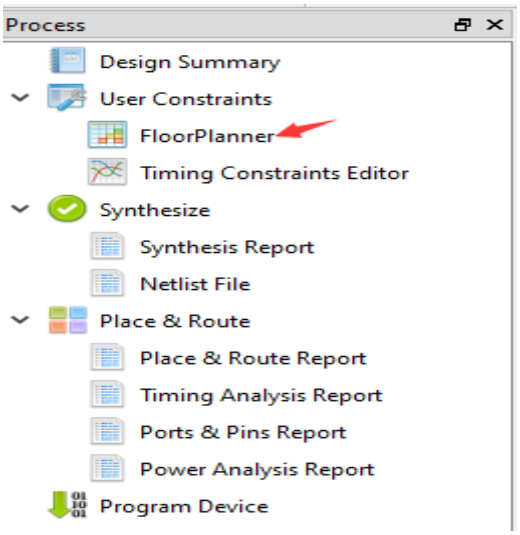
Sintetizar

Después de terminar el código, vaya a la interfaz "Proceso" y haga doble clic en "Sintetizar" para sintetizar el código y convertir el contenido del código verilog a netlist.

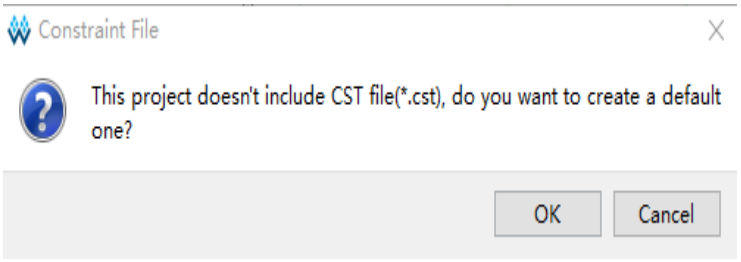


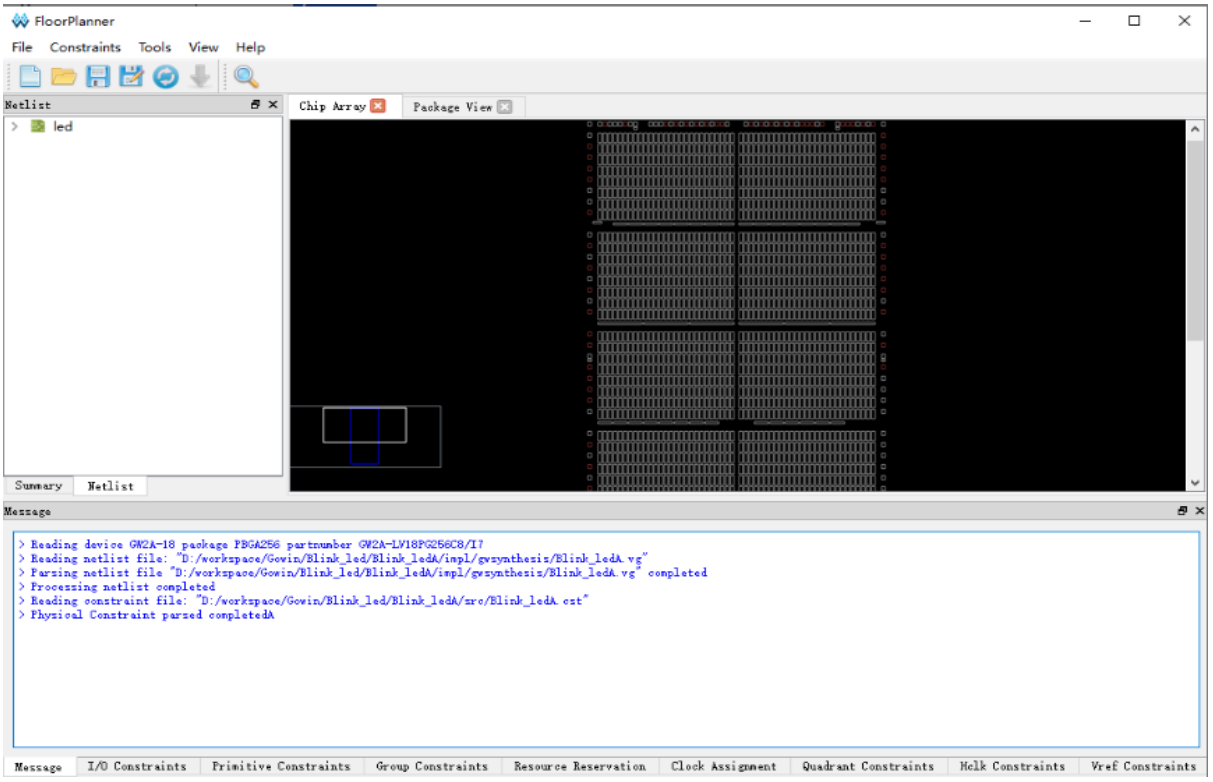
Restricción

Después de sintetizar nuestro código, necesitamos establecer restricciones para vincular los puertos definidos en nuestro código a los pines fpga, mediante los cuales podemos realizar la función de nuestro módulo en fpga. Haga clic en FloorPlanner en la parte superior de Sintetizar para establecer restricciones.

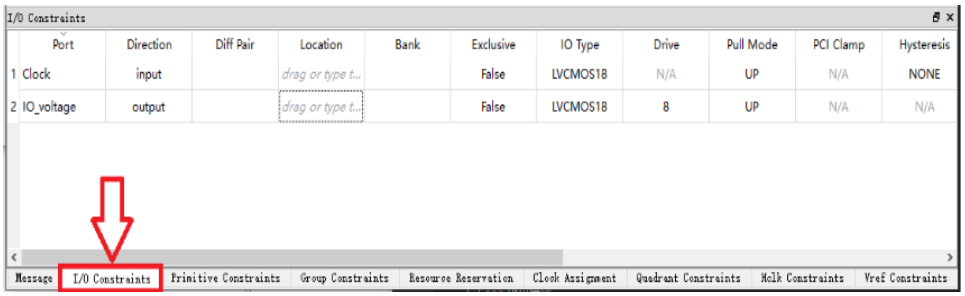


Dado que es la primera vez que lo creamos, aparecerá el siguiente cuadro de diálogo. Haga clic en Aceptar y aparecerá la interfaz de restricción gráfica.

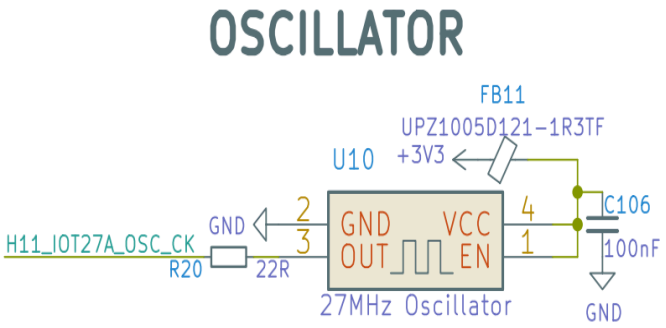




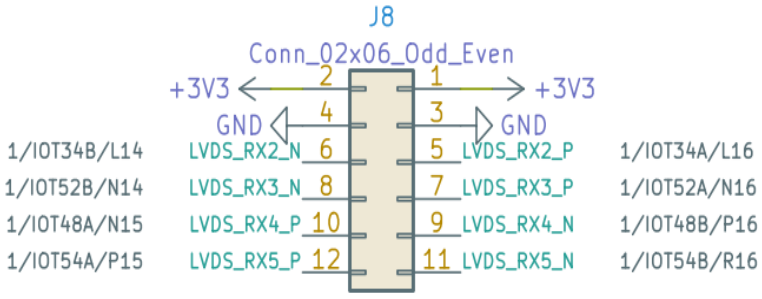
Utilizamos el método IO Constraints que se muestra a continuación para restringir los pines:



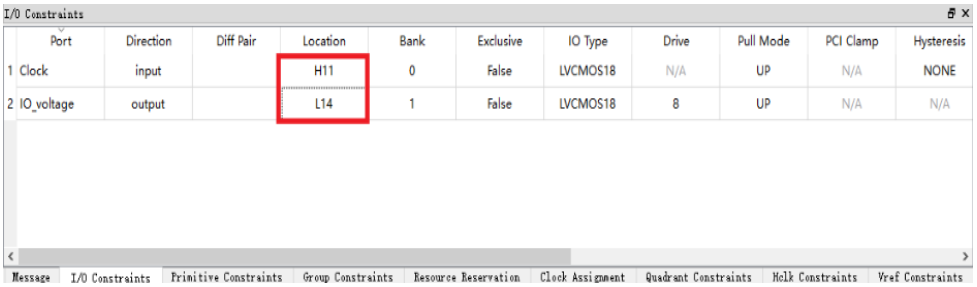
Según **Esquema del tablero central**, podemos saber que el pin de entrada del oscilador de cristal es H11



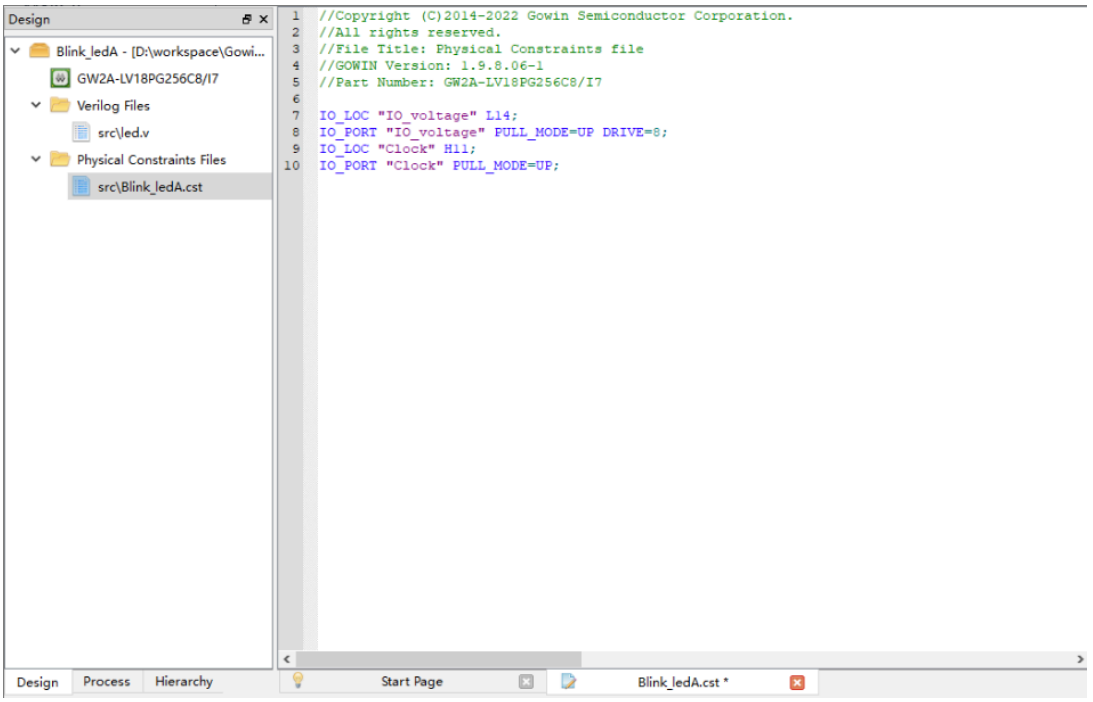
Teniendo en cuenta la serigrafía IO en ext_board, decidimos utilizar el pin **L14** en ext_board para flashear.



Entonces, para las restricciones de IO en la ventana interactiva de FloorPlanner, completamos los siguientes valores para PUERTO y Ubicación:

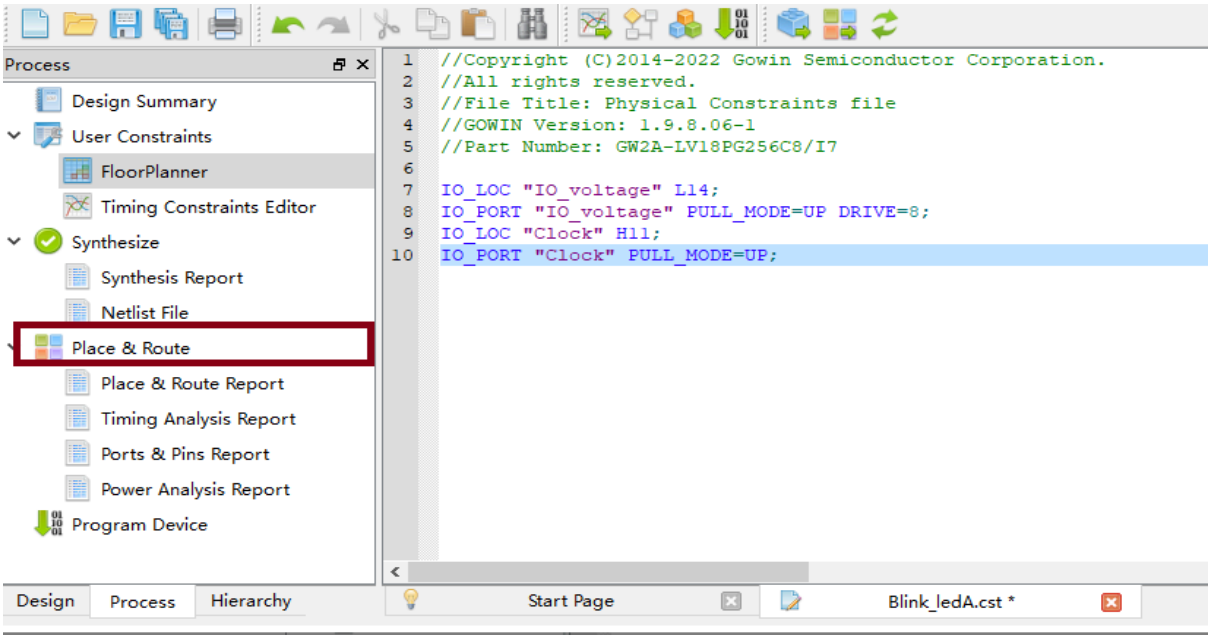


Usamos **Ctrl + S** Para guardar el archivo de restricciones, cierre la interfaz gráfica interactiva de FloorPlanner. Luego vemos que hay un archivo .cst en nuestro proyecto y su contenido es fácil de entender.



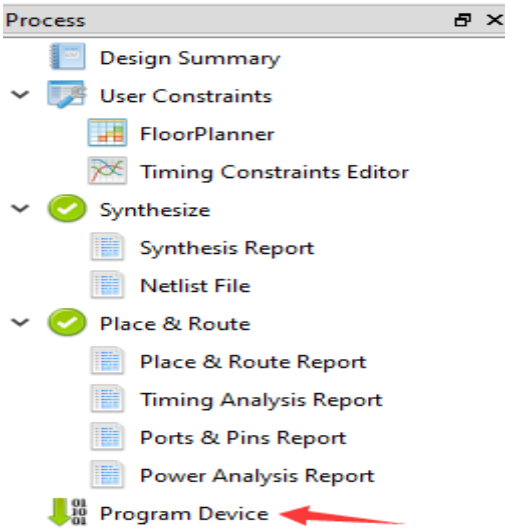
Place & Route

Después de terminar de restringir, ejecutamos Place & Route. El propósito es sintetizar la lista de redes generada y nuestras restricciones definidas para calcular la solución óptima a través de IDE y luego asignar recursos razonablemente en el chip FPGA. Haga doble clic en Lugar y ruta marcados con un cuadro rojo para ejecutar

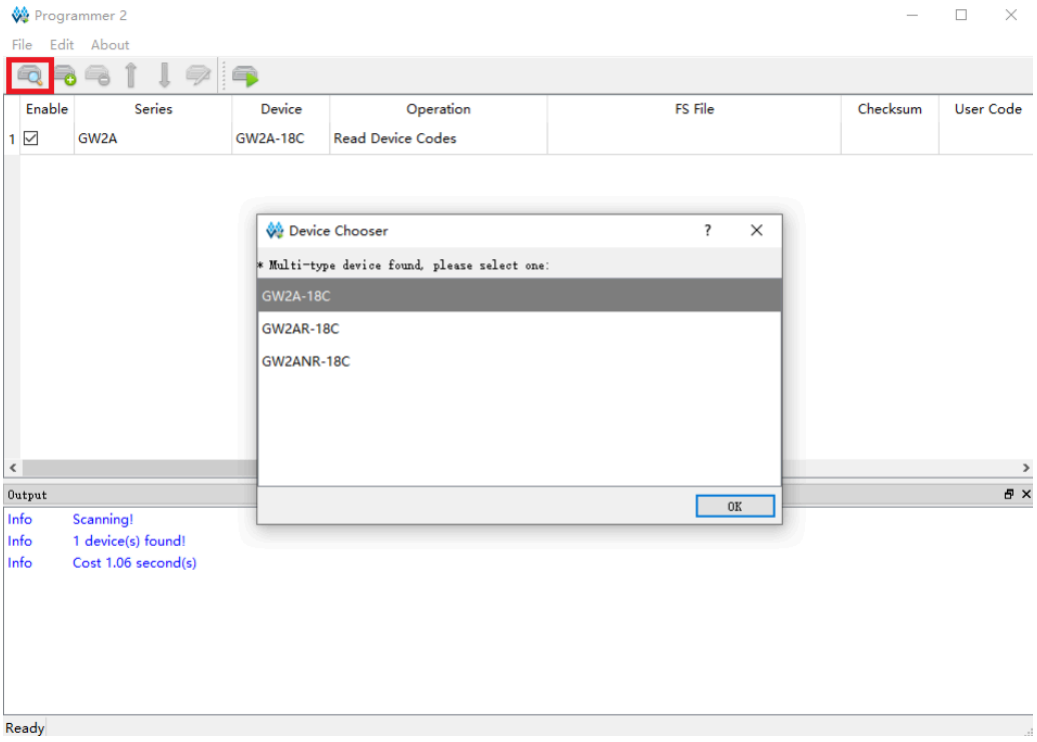


7. Programar la FPGA (cargar Bitstream)

Haga clic en **Program Device** dos veces para ejecutar la aplicación Programador.



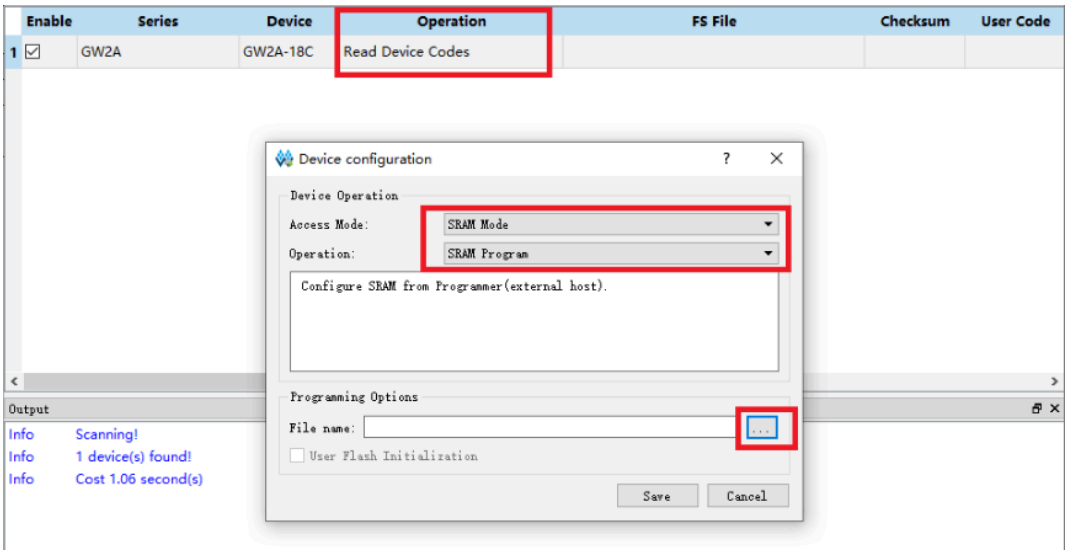
Haga clic en scan_device marcado con un cuadro rojo para escanear nuestro dispositivo.



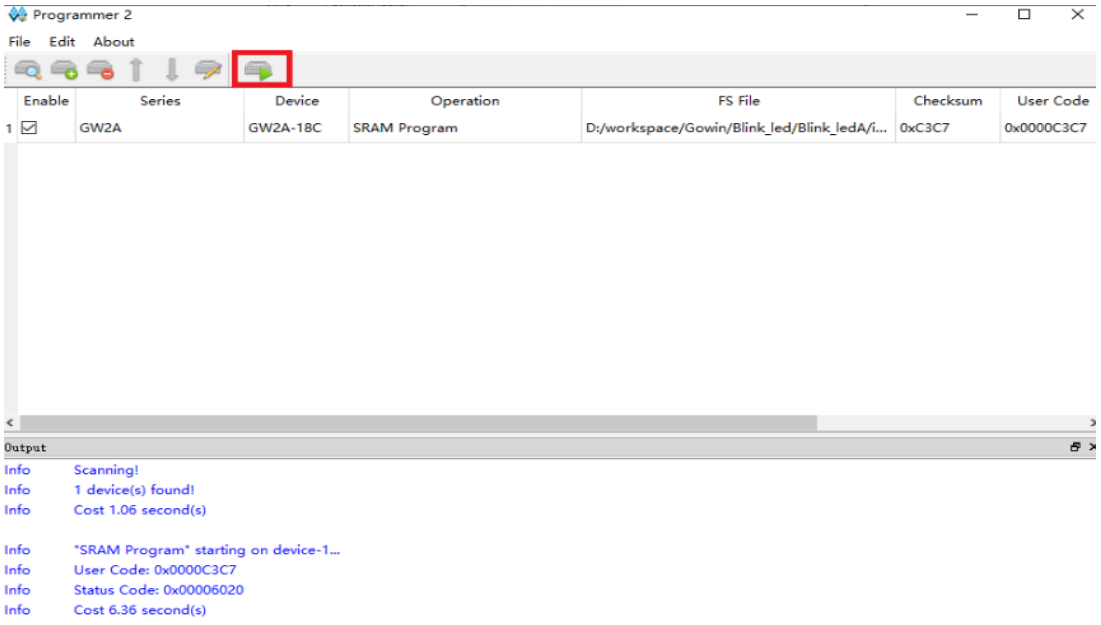
Haga clic en Aceptar para grabar fpga.

Grabar en SRAM

Normalmente, este modo se utiliza para verificar el flujo de bits. Debido a sus características de combustión rápida, se utiliza más, pero, por supuesto, la energía perderá datos, por lo que si desea encender el programa en ejecución, no puede elegir este modo. Haga clic en el cuadro de función debajo de Operación para abrir la interfaz de configuración del dispositivo, luego seleccione la opción Modo SRAM en Modo de acceso para configurar la descarga a SRAM y, finalmente, haga clic en el cuadro de tres puntos a continuación para seleccionar nuestra operación generada .fs archivo bitstream. En términos generales, el archivo de firmware bitstream se encuentra en el directorio impl -> pnr

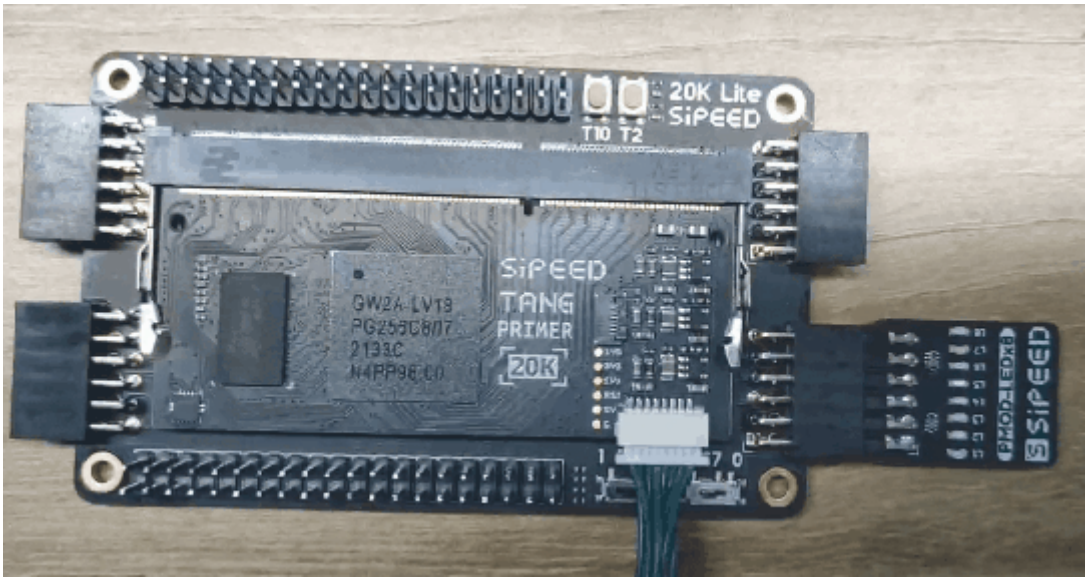


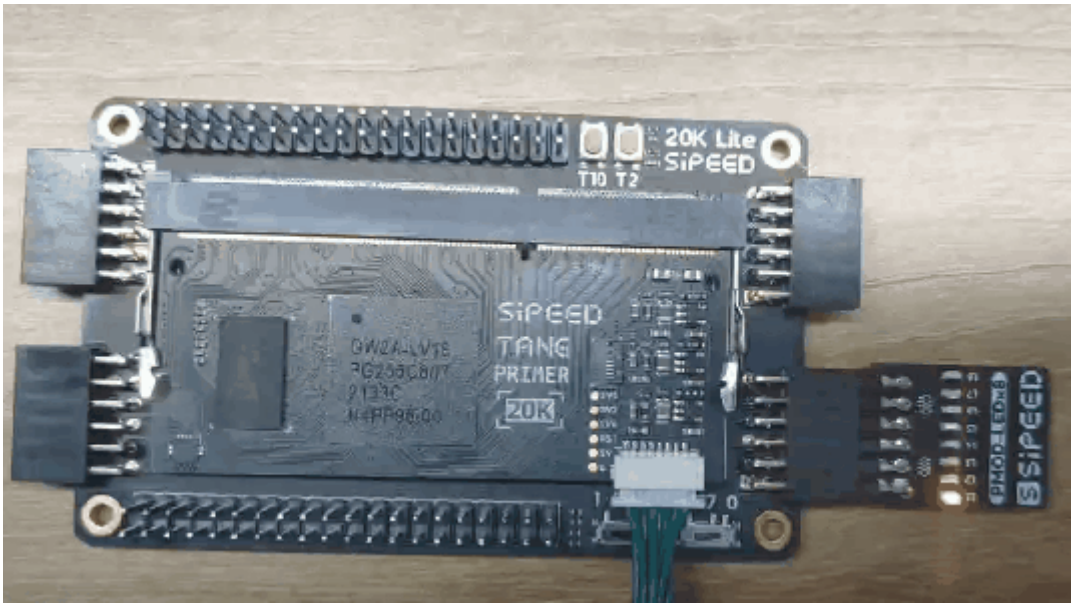
Haga clic donde está el cuadro rojo para grabar el firmware.



Aquí terminamos de descargar en SRAM.

Resultado


20K Lite
SiPEED
T10 T2
SiPEED
TANG
PRIMER
20K
SiPEED


20K Lite
SiPEED
T10 T2
SiPEED
TANG
PRIMER
20K
SiPEED