

Manual de Usuario

Herramienta de Alto Nivel para el Diseño de SoC basado
en RISC-V

César Solís Valverde
Greivin Fallas Sánchez

Contenido

Requisitos previos:	3
Ejecución.....	3
Agregar Componentes.....	4
Timer.....	4
Periféricos.....	4
UART	5
RAM	5
Configuración de CPU.....	6
Compilar el archivo.....	6
Interconexión de componentes	7
Generador de código.....	8
Componentes	8
CLK	8
CPU	9
PIO	9
TIMER.....	9
UART	10
Programa del SoC	11
Script.....	14

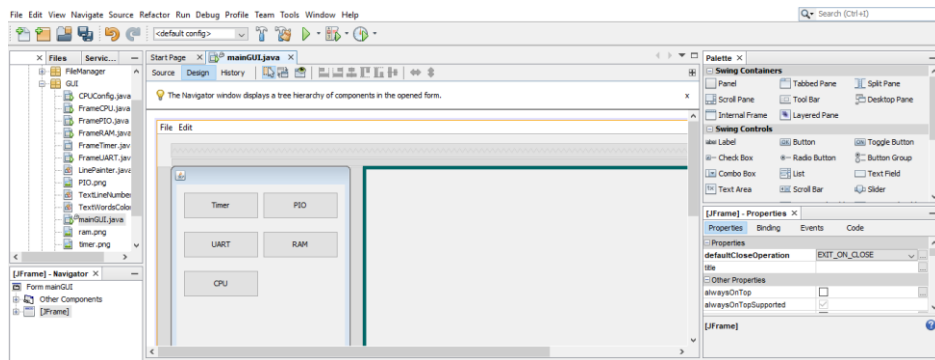
En el siguiente documento se detallarán paso a paso las acciones necesarias para poder generar sistemas a la medida utilizando la herramienta de alto nivel para el diseño de SoC basado en RISC-V.

Requisitos previos:

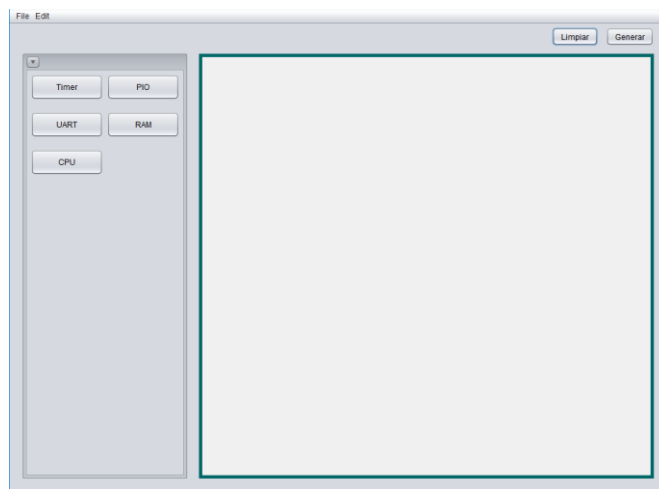
- Compilador RISC-V debidamente configurado (ver manual de instalación).
- NetBeans IDE 8.2 o posterior.
- JDK 8u11 o posterior.
- Máquina con Linux, preferiblemente Ubuntu 16.04

Ejecución

Para la ejecución del programa, se realiza desde NetBeans, ya que al tratarse del primer prototipo no se trabajó con un archivo .jar, en este caso. Por lo cual se ejecuta pulsando el botón *Run Project*.



Al momento de ejecutar el programa se despliega la siguiente pantalla:



Agregar Componentes

Es posible agregar distintos componentes, entre ellos un temporizador, periféricos, UART y memoria RAM, además de la posibilidad de configurar el procesador por medio de código en lenguaje C. Así mismo, se cuentan con validaciones de todos los campos respectivos y choque de direcciones de memoria para todos los componentes.

Timer

Para agregar un módulo temporizador basta con presionar el botón con la reseña *Timer*, seguidamente se despliega la siguiente pantalla.

The screenshot shows a configuration window titled "Configuraciones Basicas" for a "Temporizador" (Timer) component. On the left, there are four sections: "Tamano de Conteo" with a dropdown set to 8; "Periodo" with a text input of 50 and a unit dropdown set to "us"; an "Editable" checkbox which is unchecked; and "Direccion" with a "Base" text input. On the right, a "Diagrama de Bloques" shows a box labeled "Temporizador" with three connection points: "Reloj", "Reset", and "Esclavo". At the bottom right are "Cancelar" and "Aceptar" buttons.

Los parámetros configurables del temporizador son los siguientes:

- ❖ **Tamaño de Conteo:** representa la cantidad de bits con la cual se va a realizar el conteo.
- ❖ **Periodo:** representa el periodo que posee el timer, cabe la posibilidad de seleccionar las unidades deseadas, ya sean milis, micros, segundos o clocks.
- ❖ **Editable:** permite habilitar la edición del periodo del temporizador en tiempo de ejecución.
- ❖ **Dirección:** corresponde a la dirección asignada en memoria al timer.

Periféricos

Para agregar periféricos se debe presionar el botón *PIO* y se despliega la siguiente pantalla de configuración.

The screenshot shows a configuration window titled "Configuraciones Basicas" for a "Periféricos (PIO)" component. On the left, there are three sections: "Ancho" with a text input of 31; "Direccion" with a box containing "Input" and "Output"; and "Base" with a text input. On the right, a "Diagrama de Bloques" shows a box labeled "Periféricos (PIO)" with three connection points: "Reloj", "Reset", and "Esclavo". At the bottom right are "Cancelar" and "Aceptar" buttons.

Los parámetros de configuración para los periféricos son los siguientes:

- ❖ **Ancho:** corresponde al tamaño en bits que se le asigna al periférico.
- ❖ **Dirección:** representa si se trata de una salida o entrada.
- ❖ **Dirección Base:** corresponde a la dirección asignada en memoria al periférico.

UART

Para agregar un módulo de comunicación basta con presionar el botón con la reseña *UART*, seguidamente se despliega la siguiente pantalla.

The screenshot shows a configuration window titled "Configuraciones Basicas". On the left, there are three settings: "Baud Rate" with a dropdown menu showing "115200", "Bit de Datos" with a text input field showing "8", and "Bit de Detenimiento" with a text input field showing "1". On the right, there is a "Diagrama de Bloques" section with a box labeled "UART". Three lines connect the box to labels: "Reloj", "Reset", and "Esclavo". At the bottom right, there are two buttons: "Cancelar" and "Aceptar".

Los parámetros de configuración son los siguientes:

- ❖ **Baut Rate:** corresponde a la velocidad de transferencia de los datos.
- ❖ **Bit de Datos:** representa la cantidad de bits que se transfieren.
- ❖ **Bit de Detenimiento:** corresponde al bit en el cual se detiene la lectura y/o escritura.

RAM

Para agregar la memoria RAM se debe presionar el botón *RAM* y se despliega la siguiente pantalla de configuración.

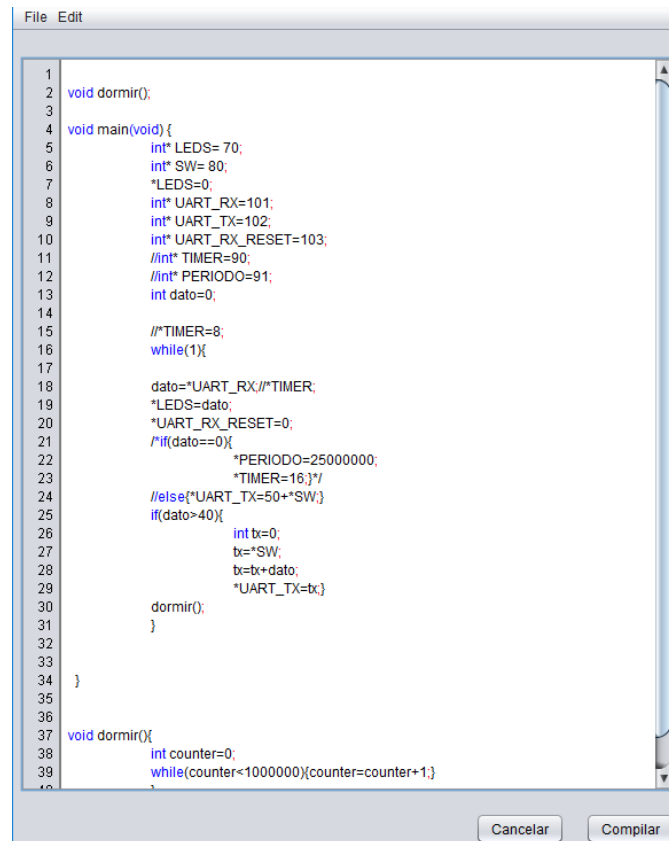
The screenshot shows a configuration window titled "Configuraciones Basicas". On the left, there are two settings: "Ancho" with a dropdown menu showing "8" and "Tamaño" with a text input field followed by the unit "bytes". On the right, there is a "Diagrama de Bloques" section with a box labeled "Memoria RAM". Three lines connect the box to labels: "Reloj", "Esclavo", and "Reset". At the bottom right, there are two buttons: "Cancelar" and "Aceptar".

Los parámetros de configuración corresponden a los siguientes.

- ❖ **Ancho:** corresponde al ancho del bus.
- ❖ **Tamaño:** representa el tamaño que se le asignará a la memoria.

Configuración de CPU

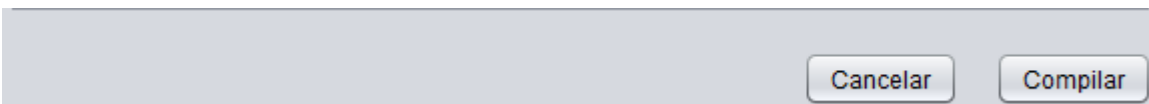
Para configurar el CPU basta con presionar el botón con la reseña *CPU*, seguidamente se despliega la siguiente pantalla.



```
1 void dormir();
2
3
4 void main(void) {
5     int* LEDS= 70;
6     int* SW= 80;
7     *LEDS=0;
8     int* UART_RX=101;
9     int* UART_TX=102;
10    int* UART_RX_RESET=103;
11    //int* TIMER=90;
12    //int* PERIODO=91;
13    int dato=0;
14
15    //TIMER=8;
16    while(1){
17
18        dato=*UART_RX;//TIMER;
19        *LEDS=dato;
20        *UART_RX_RESET=0;
21        /*if(dato==0){
22            *PERIODO=25000000;
23            *TIMER=16;}*/
24        //else{*UART_TX=50+*SW;}
25        if(dato>40){
26            int tx=0;
27            tx=*SW;
28            tx=tx+dato;
29            *UART_TX=tx;
30
31            dormir();
32        }
33    }
34
35
36
37 void dormir(){
38     int counter=0;
39     while(counter<1000000){counter=counter+1;
40 }
```

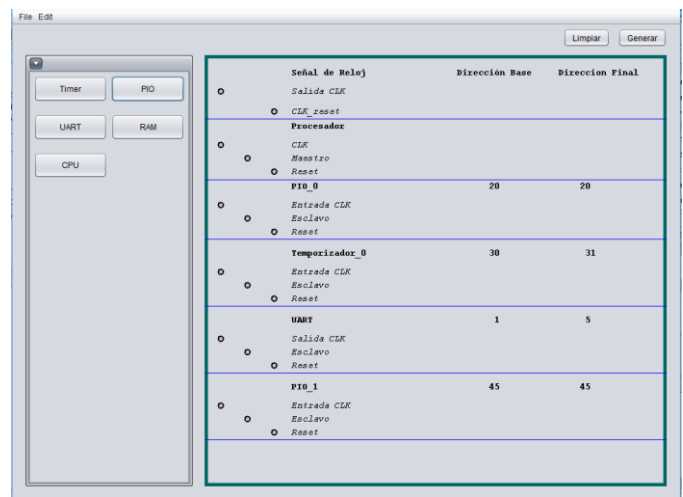
Compilar el archivo

Al haber creado las rutinas necesarias para el procesador se debe realizar la compilación respectiva, puede ser mediante un código generado desde cero o uno previamente escrito, basta con presionar el botón *Compilar* de la pantalla de configuración, como se muestra en la siguiente imagen, además si se desea cancelar la edición se debe presionar el botón *Cancelar*.

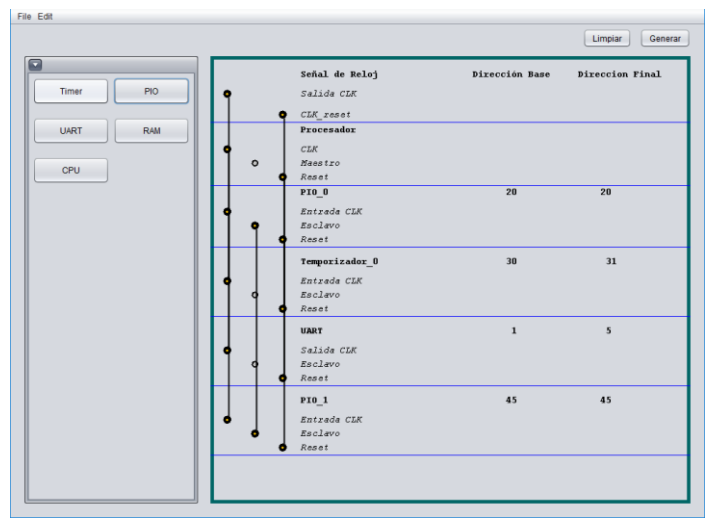


Interconexión de componentes

Al momento de agregar distintos módulos en interfaz gráfica existe la posibilidad de interconectarlos, cabe destacar que por defecto se comparten la señal de reloj y de reset con el procesador. Por ejemplo, si se desean interconectar los dos periféricos de la siguiente imagen basta con pulsar en los pines respectivos de esclavo en cada uno de ellos.



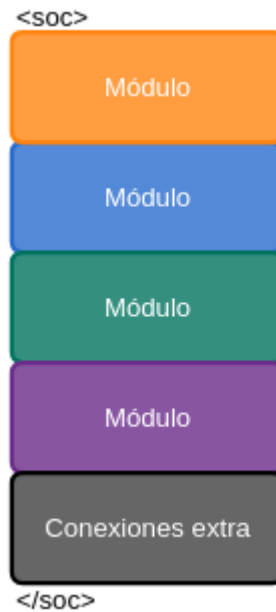
Dando como resultado lo siguiente, además si se desea se pueden conectar las demás señales.



Generador de código

Este archivo XML se puede crear manualmente y se puede utilizar el generador de código para crear el sistema en chip especificado, para esto se debe entender las propiedades de cada componente y cómo utilizarlo. Este manual detallará la composición de cada componente y después la integración entre ellos. Algunas de las propiedades de cada componente se auto configuran desde la interfaz gráfica, pero en la edición manual se deben especificar, por eso algunas de las propiedades de la siguiente sección no se observan desde la interfaz gráfica.

La estructura del grafo de dependencias es la siguiente:



En el archivo XML se inicia y se cierra con la etiqueta `<soc>`, entre estas etiquetas se encuentran las definiciones de los componentes a utilizar. Cada componente corresponde a un módulo definido entre las etiquetas `<module>`, para cada módulo se tienen sus propiedades respectivas que se detallarán a continuación.

Componentes

Cada componente se diferencia de los demás con un atributo `"type"` en la etiqueta `<module>` donde se especifica el tipo de elemento.

CLK

Este componente especifica la frecuencia de operación que tiene el sistema, su atributo `"type"` es `"clk"`. La única propiedad para este módulo es frecuencia.

frequency: es la frecuencia de operación del sistema en pulsos de reloj.

Ejemplo de grafo:

```
<module type = 'clk'>  
  <frequency>50000000</frequency>  
</module>
```

CPU

Este componente se encarga de crear el procesador y cargar su memoria de instrucciones, su atributo “type” es “cpu”. No contiene propiedades configurables manualmente, siempre se configura con un tamaño reservado de 512 direcciones para interactuar con otros elementos, el tamaño de la memoria RAM es de 8000 direcciones y la memoria ROM se autoconfigura con el tamaño del programa que debe ejecutar. Este elemento se debe especificar en el grafo de dependencias, aunque no posea configuraciones manuales.

Ejemplo de grafo:

```
<module type='cpu' >  
</module>
```

PIO

Este componente se encarga de determinar puertos de entrada y salida para el SoC, su atributo “type” es “gpio” seguido por un número correspondiente a su ID, separados por una barra baja.

-*size*: corresponde al tamaño del puerto, el máximo es de 32 y mínimo de 1.

-*mode*: es la dirección de los datos del puerto, input para puerto de entrada y output para puerto de salida.

-*direction*: es la dirección que se le asignará al puerto a través de la cual el procesador del SoC puede transmitir o recibir información.

Ejemplo de grafo:

```
<module type='gpio_4'>  
  <size>7</size>  
  <mode>output</mode>  
  <direction>74</direction>  
</module>
```

TIMER

Este componente se encarga de manejar controles de tiempo, requiere de la frecuencia de la placa a utilizar para ajustarse a las medidas de tiempo requeridas en el periodo. Este elemento lleva un conteo decreciente que cambia según el periodo determinado. Su atributo “type” es “timer” seguido con el ID del elemento separado por una barra baja.

-*counter*: es el tamaño en bits del contador a utilizar, no puede ser de cero.

-*period*: es el periodo para realizar los cambios en el contador.

-*unit*: es la unidad del periodo, puede ser s, ms, us y clocks.

-*editable*: esta propiedad permite modificar el periodo del timer desde el código c del procesador si es 1, si en 0 no se permite editar el periodo.

-*direction*: es la dirección base que tiene este elemento, utiliza 2 direcciones en total.

Mapeo:

-*direction*: es la dirección base, corresponde al contador del timer, una lectura a esta posición indica cuantas iteraciones del periodo falta para llegar a cero, una escritura a la dirección hace que el timer empiece a contar la cantidad de veces especificada cada una con la duración del ciclo especificado.

-*direction +1*: es la dirección del periodo del timer, una lectura muestra el periodo actual que posee el timer en ciclos de reloj, una escritura a esa dirección cambia el periodo del timer (si esta editable en 1) a la cantidad de ciclos indicada, el periodo siempre tiene la unidad en ciclos de reloj, si el periodo se especifica en el grafo en s, ms o us se realiza la conversión a pulsos de reloj.

Ejemplo de grafo:

```
<module type='timer_0'>
  <counter>32</counter>
  <period>1</period>
  <unit>ms</unit>
  <editable>1</editable>
  <direction>90</direction>
</module>
```

UART

Este componente se encarga de realizar transmisiones seriales desde y hacia otros computadores. Está configurado automáticamente para operar con 8 bits de datos, 1 bit de alto y sin control de flujo en hardware y software. Las frecuencias de operación soportadas son todas las menores a 115200 incluyéndola, aunque puede tener algunas pérdidas de datos en esta última. Los registros temporales internos sólo reservan la última lectura y se debe limpiar el registro de lectura a través de una dirección reservada cuando se desee desechar el dato leído. El atributo de “type” es “uart”.

-*baudrate*: es la tasa de transferencia deseada, debe ser una de los estándares para este protocolo.

-*direction*: es la dirección base para este componente, necesita de 4 direcciones en total.

Mapeo:

-*direction*: es la dirección base, sólo es lectura y muestra la tasa de transferencia.

-*direction +1*: es la dirección para RX del UART, esta dirección muestra el dato del registro de lectura, obtener este dato no limpia el registro. Si no se limpia el registro en nuevas lecturas se sobrescribe la información recibida.

-*direction +2*: es la dirección de TX de UART, se encarga de transmitir datos, contiene un registro interno de 8 bits para transmitir al puerto, las trasmisiones se realizan cuando el puerto de TX esté libre.

-*direction +3*: es la dirección de RX_RESET para limpiar el registro de lectura.

Ejemplo de grafo:

```
<module type='uart'>
  <baudrate>19200</baudrate>
  <direction>100</direction>
</module>
```

Estos componentes son los que se pueden configurar manualmente, los demás componentes que se pueden encontrar en el proyecto se utilizan con las instancias de los módulos presentados en esta sección.

En la siguiente sección se va a mostrar como interactuar con estos módulos desde el programa en C que se va a cargar a la memoria de instrucciones.

Programa del SoC

El programa del procesador interno del SoC utiliza el lenguaje C, sólo puede utilizar unidades de entero (aunque tiene soporte para punto flotante y 64 bits si se aplica la versión completa del micro puesto en el git). Para utilizar métodos se deben definir primero. El uso de variables es el mismo que en C y para acceder a los dispositivos se utilizan punteros a las direcciones establecidas en el grafo.

Por ejemplo, en los componentes se utilizó la dirección 74 para el PIO y era de salida, usando ese caso se puede establecer un ejemplo para mostrar la comunicación, puntero y uso de variables.

Ejemplo:

```
void mostrar_gpio(int*,int);

void main(void)
{
  int* LEDS=74;
  int variable=5;

  while(1)
  {
    mostrar_gpio(LEDS,digito);
  }
}

void mostrar_gpio(int* dir,int dato)
{
  int* ptr= dir;|
  *ptr=dato;
}
```

En este ejemplo se tiene la definición de un método, el uso de un puntero y una variable. Este ejemplo busca demostrar cómo se define un método y su utilización. El puntero (int* LEDS = 74;) está apuntando a un dispositivo en la dirección 74 que a como se mostró en el grado de dependencias para el componente PIO se tiene una instancia de esta en esa dirección

y es de tipo salida por lo que el dato de la variable 5 está en el puerto de salida, si ese puerto está mostrándose en unos leds de una placa entonces en los leds se va a mostrar un 5 en binario. El método “mostrar_gpio” tiene de parámetros un puntero y un dato, este mismo método muestra cómo utilizar un puntero y como enviar el dato hacia la dirección a donde apunta (“int* ptr” es un puntero y “*ptr” accede a la dirección a donde apunta). Este es el ejemplo de uso de PIO, los siguientes son los demás componentes.

Para el TIMER se puede modificar el ejemplo anterior:

```
void main(void)
{
    int* LEDS=74;
    int* TIMER=90;
    int dato=0;
    int total=0;

    while(1)
    {

        dato=*TIMER;
        dato=*TIMER;

        if(dato==0)
        {
            total=total+1;
            *TIMER=1000;
            *LEDS=total;
        }
    }
}
```

En este ejemplo se está utilizando directamente los punteros sin métodos, se tiene el mismo puntero a los LEDS en la dirección 74 y se tiene el TIMER en la dirección 90. El TIMER tiene el periodo en 1 ms por lo tanto 1000 ms equivalen a un segundo, este ejemplo lo que hace es llevar un contador de cuantos segundos han pasado desde que inicio la ejecución del programa a través del TIMER con periodo de 1 segundo y los LEDS. Cada vez que el TIMER llega a cero es porque 1 segundo ha pasado, se aumentan el contador, se muestra en los LEDS y se reinicia el TIMER para que empiece a contar desde 1000. Aquí se muestra uno de los problemas que puede tener el proyecto, al utilizar un valor proveniente desde un dispositivo e inmediatamente tratar de utilizarlo puede fallar el valor que se encuentre en el procesador puesto que aún no se ha guardado correctamente, por eso hay 2 lecturas al TIMER para que en el momento de la comparación le haya dado tiempo de guardar el dato correctamente, otra solución es utilizar un método de delay que realice un ciclo pequeño antes de utilizar un valor para darle el tiempo necesario a ser guardado antes de usarse.

Para el UART el ejemplo es el siguiente:

```

void uart_send(int);
int uart_receive();
void delay();

void main(void) {
    int* LEDS=74;
    int datouart=0;
    while(1)
    {
        datouart=uart_receive();
        delay();
        *LEDS=datouart;
        if(datouart==64)//@
        {
            uart_send(72); //H
            uart_send(73); //I
            uart_send(33); //!
            uart_send(13); //Carriage return
            uart_send(10); //New line
        }
    }
}

void delay()
{
    int temp=0;
    while(temp<800000){temp=temp+1;}
}

int uart_receive()
{
    int* UART_RX=101;
    int* UART_RX_RESET=103;
    int inrx=0;
    inrx=*UART_RX;
    *UART_RX_RESET=0;
    return inrx;
}

void uart_send (int digit)
{
    int* ptr=102;
    *ptr= digit;
}

```

Este ejemplo es más complejo, utiliza la dirección base y las direcciones reservadas del componente UART para transmitir/recibir desde un PC. Este ejemplo se mantiene recibiendo datos, cuando el dato es una arroba “@” (valor ASCII 64) envía una respuesta al PC “HI!” y un cambio de línea. El valor que sea leído por RX se va a mostrar también en los LEDS. Para este ejemplo se crearon métodos encargados de RX, TX y crear un delay entre las lecturas. Nótese que después de leer el registro RX del componente UART se limpia por medio de su dirección reservada para no confundir futuras lecturas con un valor ya leído.

Para este ejemplo se utilizó minicom con la configuración de 19200 bauds, 8 bits de datos, 1 bit de alto y sin control de hardware ni software.

Importante: Para utilizar el micro se debe de configurar un pin de RESET y usarlo para que inicie el programa.

Script

Esta sección describe el comportamiento de todo el script del generador de código. Este script tiene como entrada un archivo XML que se explicó anteriormente (contiene los módulos requeridos en el SoC) y un archivo de programa (el .c que se va a cargar en el procesador o un .c previamente compilado).

El Script opera por medio de una consola de comandos, el código fuente se encuentra en git del proyecto y con las herramientas del manual de instalación previamente configuradas se puede compilar el código fuente del script con la siguiente línea de comandos:

```
gcc SoC.c -o SoC `xml2-config --cflags --libs` -lm
```

El ejecutable creado es el generador de código Verilog, este script crea una carpeta llamada OutputFiles con 4 archivos cada vez que se ejecuta:

CodigoPrograma.txt: es un compilado de las instrucciones que se van a cargar al procesador, este archivo es portable y se puede volver a cargar en caso que no se tenga el código fuente de programa a cargar al SoC.

CodigoProgramaHIGH.txt y *CodigoProgramaLOW.txt*: son archivos con las instrucciones del programa a cargar estructuradas para el procesador, estos archivos se deben copiar al directorio donde se vaya a crear el SoC y su síntesis para FPGA, contiene las instrucciones para el procesador.

Top.v: es el archivo que contiene todas las instancias de los componentes especificados y conectados, para que este archivo sea sintetizable se debe agregar al directorio donde se desea realizar la síntesis del proyecto para una FPGA (junto con *CodigoProgramaHIGH.txt* y *CodigoProgramaLOW.txt*), también se deben encontrar en ese directorio los módulos requeridos para el proyecto que se encuentran en el git.

Para utilizar el script desde una consola de comandos se requiere de los parámetros que especifican el archivo, -x para el XML, -p para el programa a cargar y -c para el compilado del programa (no es requerido si se especifica el programa).

Ejemplo de comando:

```
./SoC -p sumador3bits.c -x sumador3bits.xml
```

Este comando creará los archivos del programa “sumador3bits.c” en la carpeta OutputFiles y el “Top.v” contendrá los componentes especificados en “sumador3bits.xml”.