INFORME DE PROYECTO
CONTROL DE VELOCIDAD DE AUTO EN VHDL

Alumno: Patricio Germán Silva*
Profesor: Germán Hatchmann**

Introducción a VHDL
Ingeniería en Mecatrónica

5 de diciembre de 2024

*Correo electrónico: silvap@fcal.uner.edu.ar
**Correo electrónico: hachmanng@fcal.uner.edu.ar

Índice

1. Introducción **1**
1.1. Herramientas . . . 1

2. Diseño de la solución **3**
2.1. EL puente H **L293D** . . . 3
2.2. Puertos de I/O . . . 5
2.3. Esquemático general . . . 5

3. Módulos **7**
3.1. UaRx . . . 7
3.2. UaTx . . . 7
3.3. CommProtRx . . . 8
3.4. DecodeCmd . . . 9
3.5. HBridgeCtrl . . . 9
3.6. ToDisplay . . . 10
3.7. Módulos auxiliares . . . 11
3.7.1. ModuleCounter . . . 11
3.7.2. Counter . . . 12
3.7.3. BaudRateGen . . . 12
3.7.4. TTrigger . . . 12
3.7.5. PwmGen . . . 13
3.7.6. HexToSevSeg . . . 13

4. Simulación **14**
4.1. Prueba en la placa de desarrollo **Arty A7-100T** . . . 15

5. Conclusiones **17**

6. Anexos: Archivos VHDL **18**
6.1. Archivo **BaudRateGen.vhd** . . . 18
6.2. Archivo **CommProtRx.vhd** . . . 19
6.3. Archivo **Counter.vhd** . . . 21
6.4. Archivo **DecodeCmd.vhd** . . . 22
6.5. Archivo **HBridgeCtrl.vhd** . . . 25
6.6. Archivo **HexToSevSeg.vhd** . . . 27
6.7. Archivo **IMain.vhd** . . . 28
6.8. Archivo **ModuleCounter.vhd** . . . 30
6.9. Archivo **PwmGen.vhd** . . . 31
6.10. Archivo **ToDisplay.vhd** . . . 32
6.11. Archivo **TTrigger.vhd** . . . 34
6.12. Archivo **Uart.vhd** . . . 35
6.13. Archivo **UaRx.vhd** . . . 36

6.14. Archivo **UaTx.vhd** 38

7. Anexos: Archivos de TESTBENCH **40**
7.1. Archivo **BaudRateGen_tb.vhd** 40
7.2. Archivo **CommProtRx_tb.vhd** 42
7.3. Archivo **Counter_tb.vhd** 45
7.4. Archivo **DecodeCmd_tb.vhd** 47
7.5. Archivo **HBridgeCtrl_tb.vhd** 50
7.6. Archivo **HexToSevSeg_tb.vhd** 52
7.7. Archivo **IMain_tb.vhd** 53
7.8. Archivo **ModuleCounter_tb.vhd** 57
7.9. Archivo **PwmGen_tb.vhd** 59
7.10. Archivo **ToDisplay_tb.vhd** 61
7.11. Archivo **TTrigger_tb.vhd** 62
7.12. Archivo **Uart_tb.vhd** 64
7.13. Archivo **UaRx_tb.vhd** 67
7.14. Archivo **UaTx_tb.vhd** 69

1. Introducción

Se realizará el diseño, descripción en VHDL y simulación de un sistema capaz de realizar el control de un auto bimotor seguidor de linea. El diseño debe contemplar:

- Control de dirección y velocidad para un driver L293D.
- Control de un display de siete segmentos donde se muestra información de velocidad y modo de funcionamiento.
- Comunicación mediante interfaz UART y protocolo rs232 (TTL) con una velocidad de 9600 baudios, 8 bits, de datos, 1 bit de stop, sin paridad, sin control de flujo. Solo se requiere recepción.
- Para la comunicación se agrega una capa de protocolo, de frame de ancho fijo, compuesto por un byte de header, un byte de comando, dos bytes de datos y un byte de tráiler, sin suma de comprobación.

1.1. Herramientas

Para el desarrollo se utiliza el software **Xilinx Vivado 2024.1**. Por ultimo, se comprobará el correcto funcionamiento del desarrollo en una placa **DIGILENT Artix-7 FPGA modelo Arty A7-100T (xc7a100tcsg324-1)**.

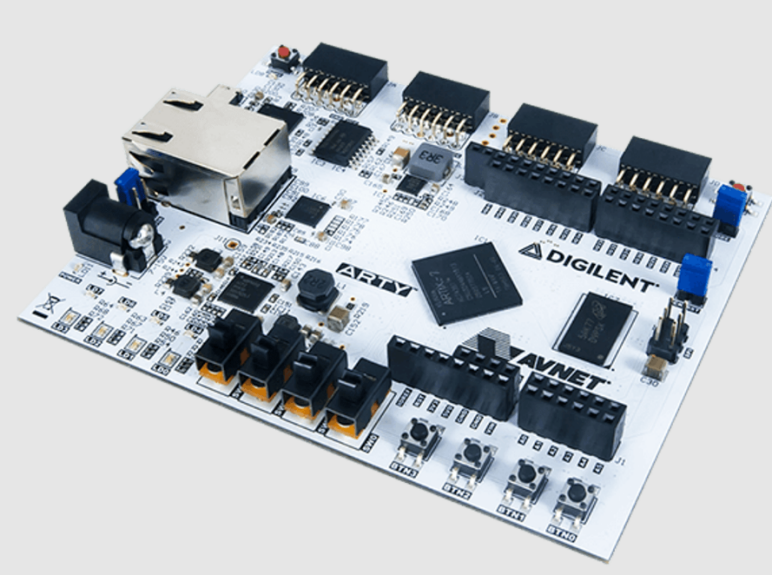


Figura 1: DIGILENT Artix-7 FPGA modelo Arty A7-100T

Los comandos son los siguientes:

COMANDO	Descripción
D000Z	STOP
D1nnZ	Velocidad Motor Derecho. nn = 10 a 90 en pasos de a 5.
D2nnZ	Velocidad Motor Izquierdo. nn = 10 a 90 en pasos de a 5.
D3nnZ	Selecciona la velocidad media para recorrer la pista. nn = 10 a 90 en pasos de a 5.
D4sxZ	Simulador de sensores. s: Simula Sensores encendidos. x: Cualquier valor.
D5sxZ	Selecciona modo de control. s: 0 Control desde PC s: 1 Control con sensores. x: Cualquier valor

2. Diseño de la solución

El diseño se lleva a cabo de forma modular, cada módulo (entity) tiene un fin específico para cada tarea:

- UaRx: modulo Uart RX para la recepción desde la PC
- UaTx: modulo Uart TX, cuya finalidad es hacer echo de la información recibida por UaRx.
- CommProtRx: modulo que valida un paquete del protocolo de comunicación, e incorpora una detección de timeout.
- DecodeCmd: Interpreta los comandos recibidos.
- HBridgeCtrl: recibe una dirección y velocidad y genera una salida PWM según los parámetros establecidos.
- ToDisplay: Envia información a un display de 7 segmentos con punto decimal
- Módulos auxiliares: módulos contadores, generadores de baud rate y PWM y decodificadores de 7 segmentos.
- EL modulo principal IMain que instancia los módulos y los interconecta.

2.1. EL puente H L293D

EL integrado **L293D** es un doble puente H para el control de pequeños motores (estrictamente es un cuádruple medio puente H).

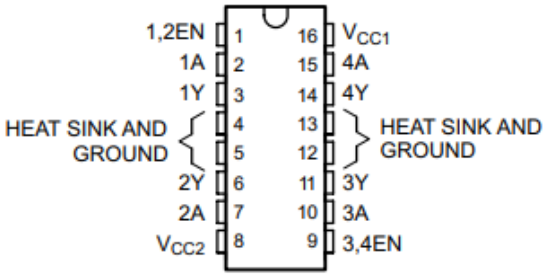


Figura 2: Pinout del integrado **L293D**

Por cada puente H posee dos entradas de dirección y una entrada Enable, se lo puede controlar mediante PWM para variar la potencia de los motores, típicamente las entradas de dirección 1A y 2A para el motor 1 (pines 2 y 7) y 3A y 4A (pines 10 y 15) se alimentan con tensión constante y la señal PWM se envía a la entrada Enable de cada puente H, 1,2EN para el motor 1 (pin 1) y 3,4EN para el motor 2 (pin 9)

Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect to printed-circuit-board ground plane with multiple solid vias
V_{CC1}	16	—	5-V supply for internal logic translation
V_{CC2}	8	—	Power VCC for drivers 4.5 V to 36 V

Figura 3: Función de los pines del integrado **L293D**

La conexión del integrado a los motores puede hacerse del siguiente modo

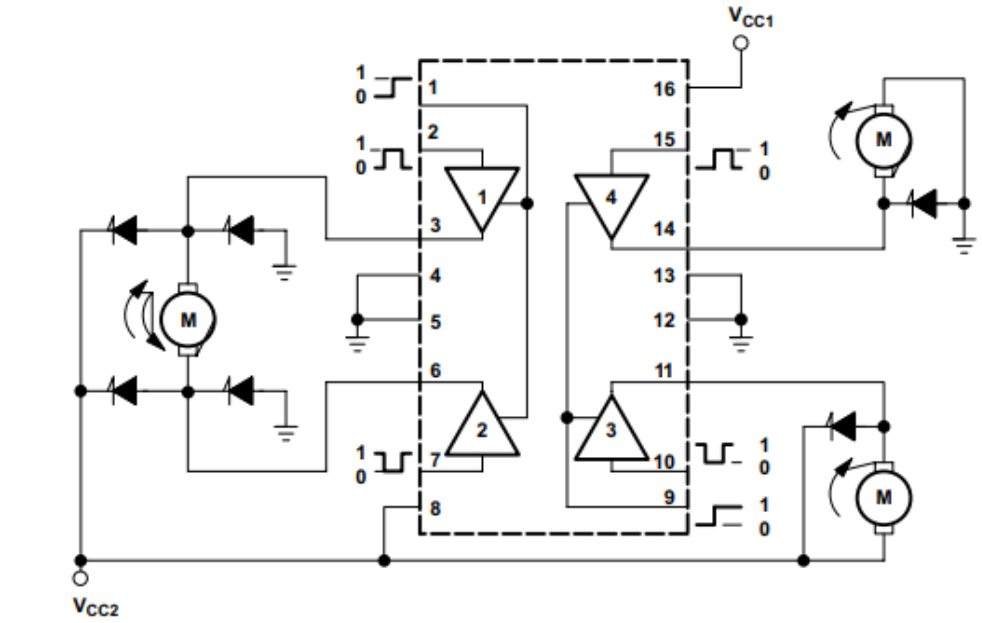


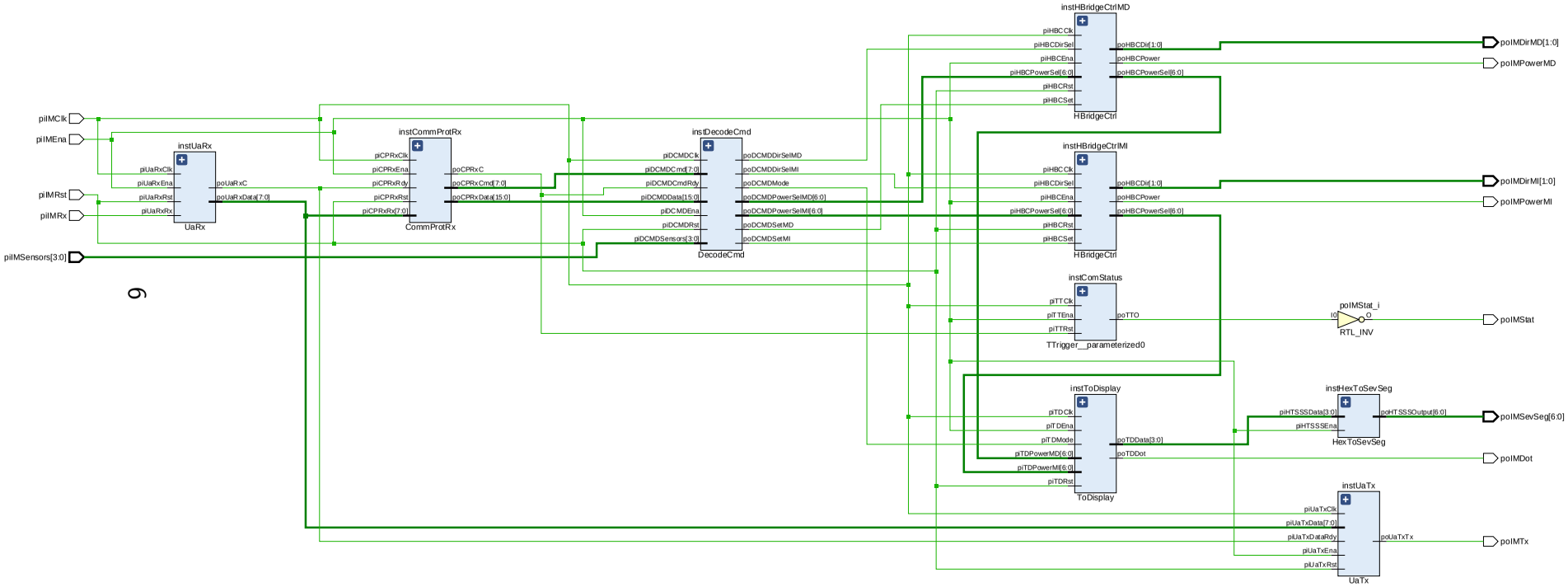
Figura 4: Ejemplo de conexión para el integrado **L293D**

2.2. Puertos de I/O

NOMBRE	DIRECCION	PIN	HEADER	DESCRIPCION
piIMClk	IN	E3	-	Clock de sistema 100MHz
piIMEna	IN	A8	SW0	Enable
piIMRst	IN	C11	SW1	Reset
piIMRx	IN	A9	-	Salida al puerto TX del PC
piIMSensors[0]	IN	B8	BTN0	Simula sensor externo izquierdo
piIMSensors[1]	IN	B9	BTN1	Simula sensor interno izquierdo
piIMSensors[2]	IN	C9	BTN2	Simula sensor interno derecho
piIMSensors[3]	IN	D9	BTN3	Simula sensor externo derecho
poIMDirMD[0]	OUT	J5	LED4	Dir1 – Segun L243D
poIMDirMD[1]	OUT	H5	LED5	
poIMDirMI[0]	OUT	T10	LED6	Dir2 – Segun L243D
poIMDirMI[1]	OUT	T9	LED7	
poIMDot	OUT	U11	IO26	Punto decimal del display de 7 segmentos
poIMPowerMD	OUT	N17	IO40	Salida PWM derecho – 100Hz
poIMPowerMI	OUT	P18	IO41	Salida PWM izquierdo – 100Hz
poIMSevSeg[0]	OUT	V16	IO33	Display – Segmento A
poIMSevSeg[1]	OUT	M13	IO32	Display – Segmento B
poIMSevSeg[2]	OUT	R10	IO31	Display – Segmento C
poIMSevSeg[3]	OUT	R11	IO30	Display – Segmento D
poIMSevSeg[4]	OUT	R13	IO29	Display – Segmento E
poIMSevSeg[5]	OUT	R15	IO28	Display – Segmento F
poIMSevSeg[6]	OUT	P15	IO27	Display – Segmento G
poIMStat	OUT	G6	LED0_R	200ms blink en cada comando valido
poIMTx	OUT	D10	-	Salida al puerto RX del PC

2.3. Esquemático general

Todos los módulos son instanciados e interconectados dentro del bloque principal IMain, el diseño general es el siguiente:


piIMClk
piIMEna
piIMRst
piIMRx
piIMSensors[3:0]
instUaRx
piUaRxClk
piUaRxEna
piUaRxRst
piUaRxRx
poUaRxC
poUaRxData[7:0]
UaRx
instCommProtRx
piCPRxClk
piCPRxEna
piCPRxRst
piCPRxRx[7:0]
poCPRxC
poCPRxCmd[7:0]
poCPRxData[15:0]
CommProtRx
instDecodeCmd
piDCMDClk
piDCMDCmd[7:0]
piDCMDCmdRdy
piDCMDData[15:0]
piDCMDEna
piDCMDRst
piDCMDSensors[3:0]
poDCMDDirSelMD
poDCMDDirSelMI
poDCMDMode
poDCMDPowerSelMD[6:0]
poDCMDPowerSelMI[6:0]
poDCMDSetMD
poDCMDSetMI
DecodeCmd
instHBridgeCtrlMD
piHBCClk
piHBCDirSel
piHBCEna
piHBCPowerSel[6:0]
piHBCRst
piHBCSet
poHBCDir[1:0]
poHBCPower
poHBCPowerSel[6:0]
HBridgeCtrl
poIMDirMD[1:0]
poIMPowerMD
instHBridgeCtrlMI
poIMDirMI[1:0]
poIMPowerMI
instComStatus
piTTClk
piTTEna
piTTRst
poTTO
TTrigger__parameterized0
poIMStat_i
RTL_INV
poIMStat
instToDisplay
piTDClk
piTDEna
piTDMode
piTDPowerMD[6:0]
piTDPowerMI[6:0]
piTDRst
poTDData[3:0]
poTDDot
ToDisplay
instHexToSevSeg
piHTSSSData[3:0]
piHTSSSEna
poHTSSSOutput[6:0]
HexToSevSeg
poIMSevSeg[6:0]
poIMDot
instUaTx
piUaTxClk
piUaTxData[7:0]
piUaTxDataRdy
piUaTxEna
piUaTxRst
poUaTxTx
UaTx
poIMTx

3. Módulos

Cada modulo es desarrollado y testeado de forma independiente, a continuación se describen los módulos desarrollados:

3.1. UaRx

Formado por una maquina de estado y un generador de baudrate. El puerto RX conectado a un conversor UART recibe desde la PC paquetes de datos de 8 bits a 9600 baudios, sin control de paridad ni control de flujo, con 1 bit de stop, totalizando un PDU de 10 bytes. Cuando un nuevo dato se recibe se genera un pulso de 1 clock de duración que notifica el evento.

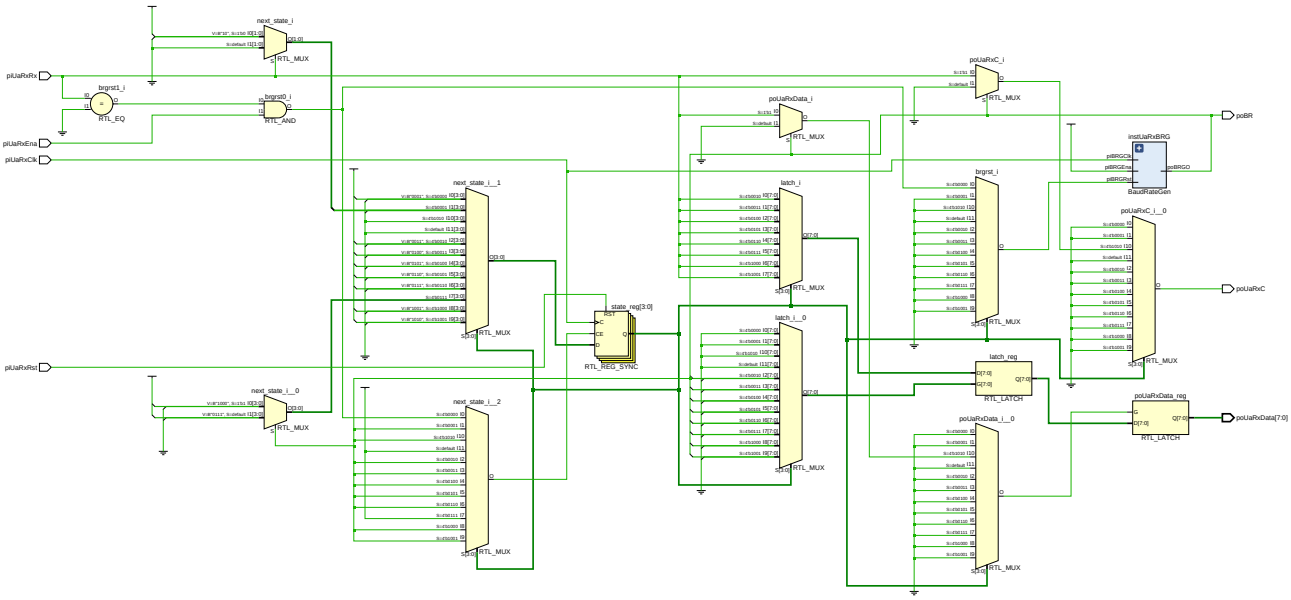


Figura 5: Esquemático del módulo **UaRx**

El diseño contempla un error de baudrate del 3.5 %, y la recepción de un nuevo bit de start de hasta un 45 % antes de finalizar el periodo de bit stop del PDU anterior.

El generador de baudrate genera, tras un reset, un primer pulso de un periodo **T/2**

3.2. UaTx

Formado por una maquina de estado y un generador de baudrate. El puerto TX conectado a un conversor UART envía a la PC paquetes de datos con la misma especificación que el modulo UaRx. El envío de un nuevo paquete de datos se inicia mediante un pulso en el puerto piUaTxDataReady, cuando se finaliza el envío el modulo notifica con un pulso de un clock de duración-

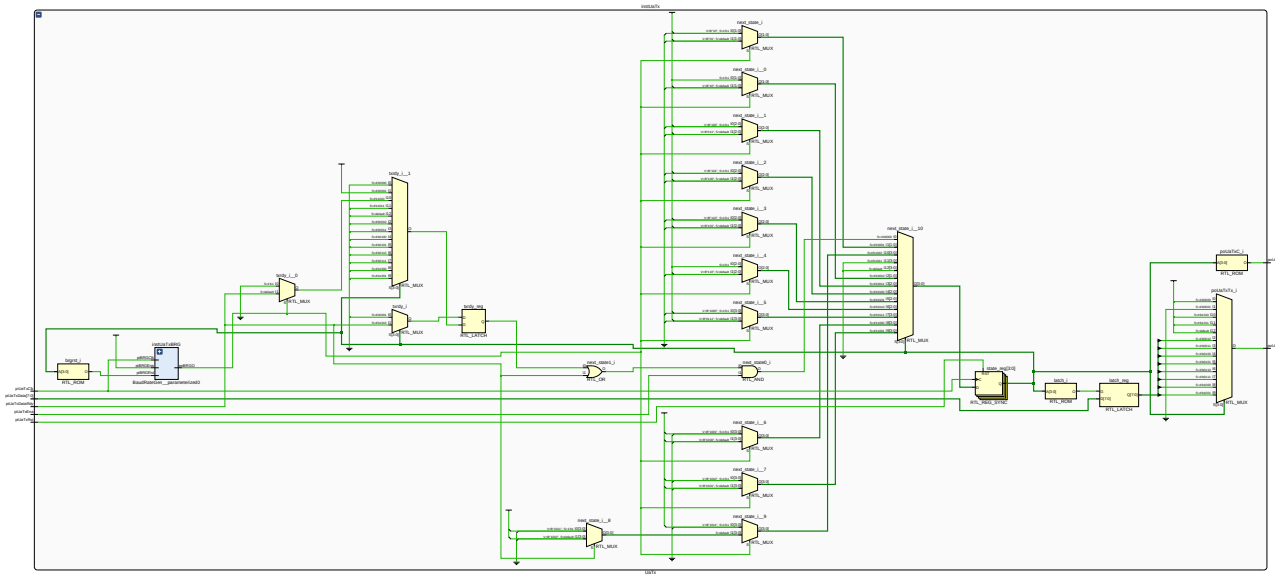


Figura 6: Esquemático del módulo **UaTx**

El diseño contempla iniciar el envío de un nuevo paquete de datos durante el envío del bit de stop del PDU anterior.

3.3. CommProtRx

Formado por una maquina de estado y un temporizador de tipo TTrigger, por cada PDU RX recibido analiza si conforma un paquete de protocolo válido, y genera un pulso de clock, poniendo el byte de comando y los dos bytes de datos en el bus de salida.

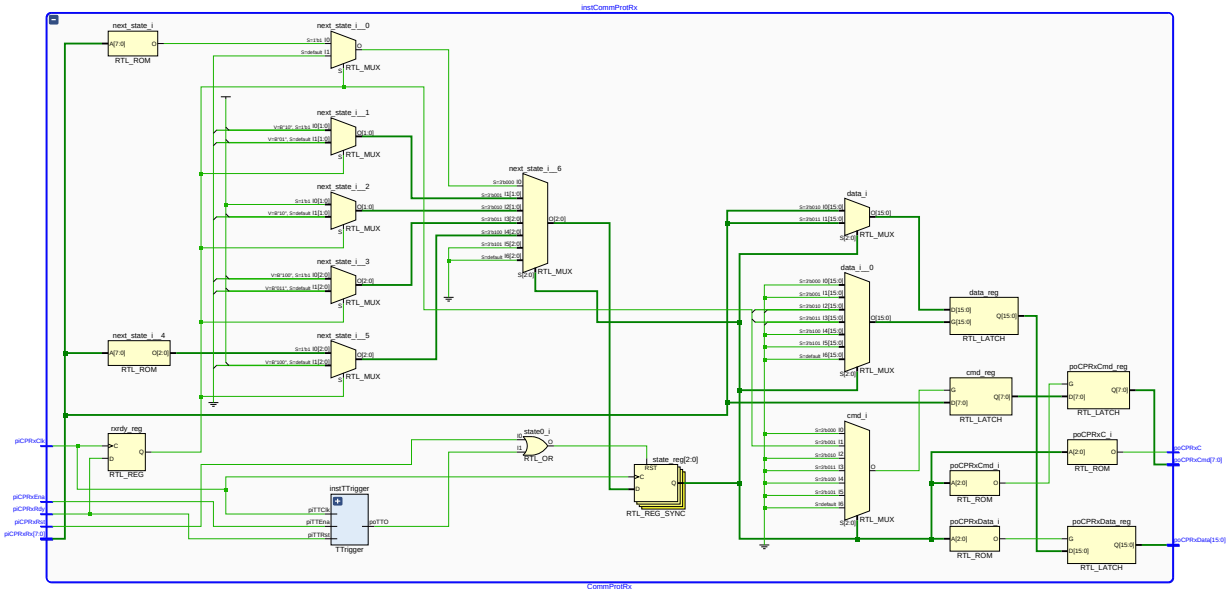


Figura 7: Esquemático del módulo **CommProtRx**

Si el periodo entre la recepción de los bytes desde el modulo RX en algún momento supera los 100ms, se dispara el trigger de timeout y la detección del paquete de protocolo se reinicia.

3.4. DecodeCmd

Cuando se recibe un comando en CommProtRx el mismo se interpreta en este modulo, si el comando corresponde a un comando válido, se interpreta los datos de entrada y se modifica el estado de los motores o sensores si corresponde.

El módulo incorpora un contador de modulo que controla y ajusta la velocidad de los motores cada 10ms según el valor de los sensores, para cuando estos operan en modo automático. Este control periódico no es necesario por tratarse de un control con un valor de corrección fijo, que es aún mas básico de que un control proporcional, pero si lo seria si se incorpora un control PD o PID.

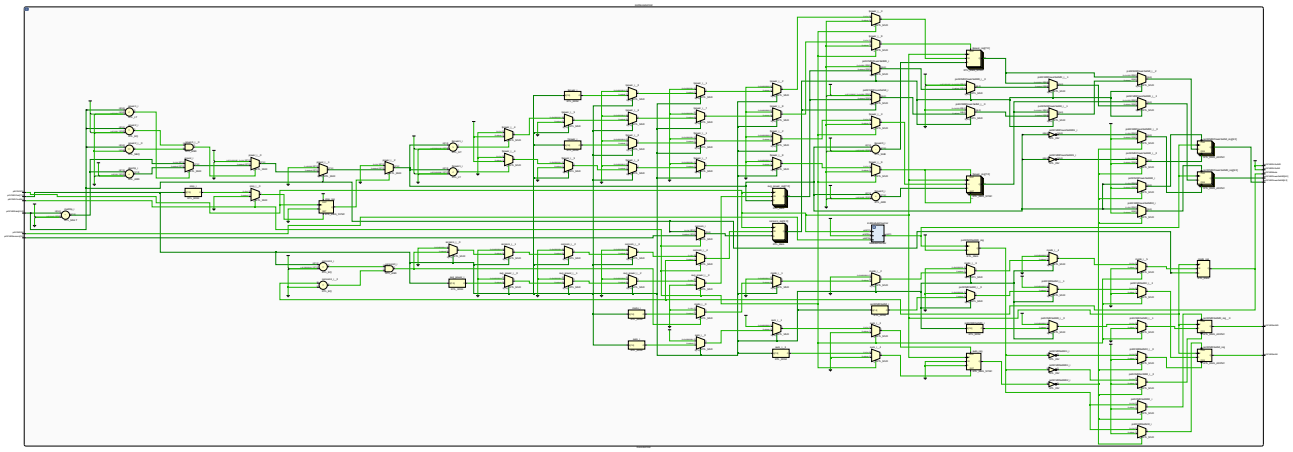


Figura 8: Esquemático del módulo **DecodeCmd**

A la lista de comandos se agrega el comando $D6nsZ$ que indica la direccion de giro de cada motor, que se indican el los 2 ultimos bits menos significativos del primer byte de datos.

3.5. HBridgeCtrl

Al recibir un pulso en el puerto set, el módulo registra las entradas duty cycle y dirección de giro, y mediante un modulo PwmGen genera una salida Pwm y setea las salidas de dirección. El diseño posee dos módulos HBridgeCtrl, uno para cada motor.

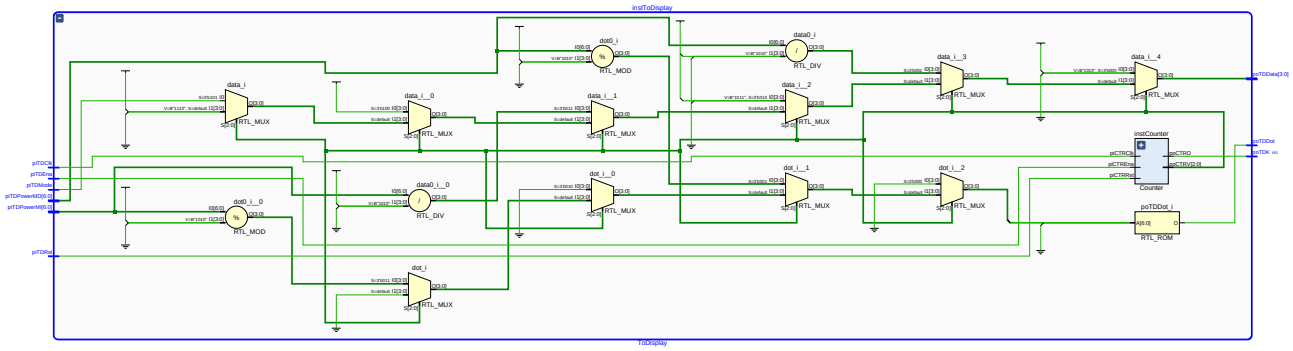


Figura 9: Esquemático del módulo **HBridgeCtrl**

El módulo almacena internamente el Duty Cylcle y direccion, por lo que no es necesario mantener los valores en el puerto de entrada. Adicionalmente, este modulo tiene como salida los valores actuales de duty cycle y direccion almacenados, para que sean tomados por el modulo ToDisplay.

3.6. ToDisplay

Tiene como entradas la dirección y duty cycle de cada motor además del modo de operación actual del auto, y mediante un modulo Counter muestra de manera cíclica durante 0.5 segundos los valores de salida, del siguiente modo y en el siguiente orden:

1. La letra **A**
2. La decena de la velocidad del motor Derecho, por ejemplo para un 85 % muestra el numero 8
3. El punto decimal, para el caso donde el duty cycle es 5 % en exceso a la decena, para 85 % se muestra el punto, para 80 % no se muestra.
4. La letra **b**
5. El duty cilce del motor Izquierdo, del mismo modo que para el motor Derecho
6. La letra **F**
7. El modo de control, **0** para modo **desde PC** o 1 para modo **desde sensores**.

La salida de este módulo es un valor hexadecimal de 4 bits mas una salida para el punto, la salida debe decodificarse para poder ser mostrada.

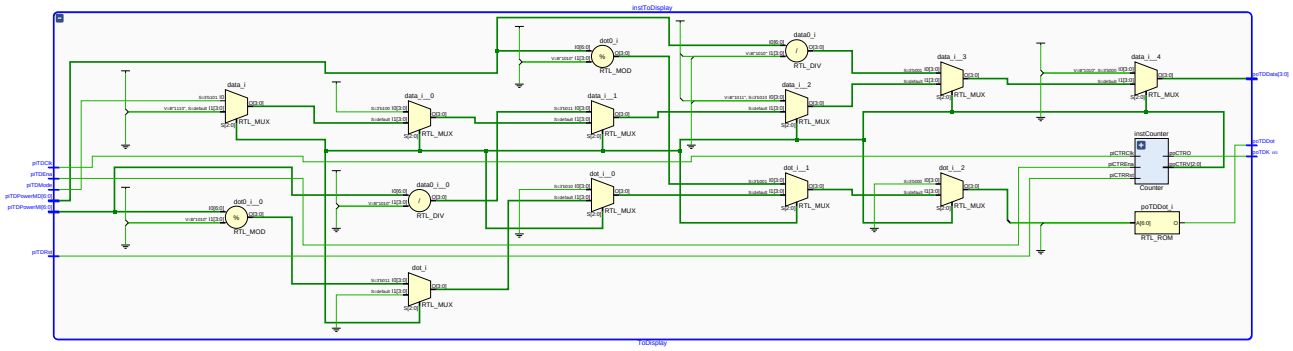


Figura 10: Esquemático del módulo **ToDisplay**

3.7. Módulos auxiliares

Son módulos de índole mas genérica, muchos de ellos son variaciones de contadores de modulo ajustados al propósito que se busca.

Esto es porque, a diferencia de un programa destinado a generar código ejecutable donde la modularización mediante funciones permite reducir notablemente el tamaño del programa en memoria, en hardware no se posee esta ventaja, un mismo modulo que se instancia dos veces da como resultado dos implementaciones independientes dentro del FPGA. Por esto puede resultar conveniente que cada diseño se ajuste exactamente a lo necesario.

3.7.1. ModuleCounter

Diseño básico de un contador de módulo. Se tiene como entrada una señal de clock, un contador interno mantiene un valor que se incrementa con cada clock, cuando se alcanza la cantidad configurada por diseño se genera un pulso de salida y reinicia la cuenta.

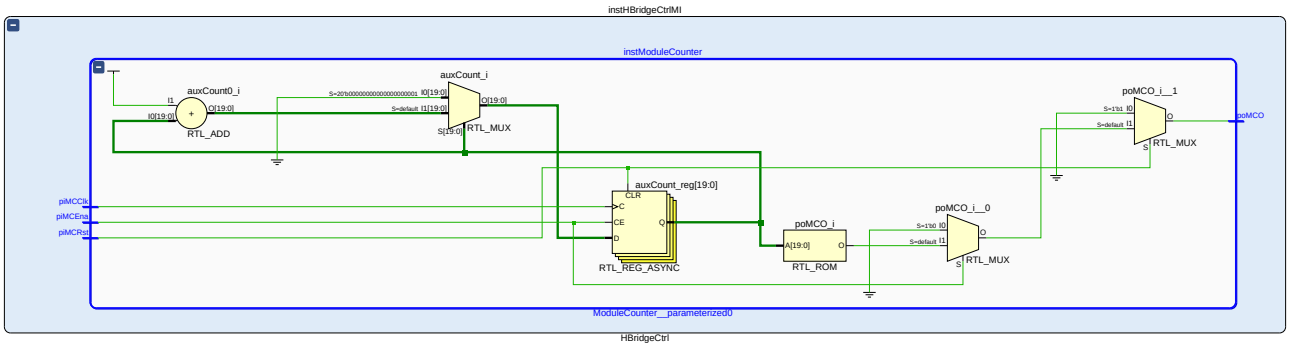


Figura 11: Esquemático del módulo **ModuleCounter**

Este diseño lo utiliza el modulo DecodeCmd para realizar la actualización de la velocidad de los motores segun la lectura de los sensores.

3.7.2. Counter

Se trata de un modulo que engloba un contador mas un contador de modulo. La salida es un bus de N bits que se va incrementando cada vez que el contador de modulo interno alcanza su valor.

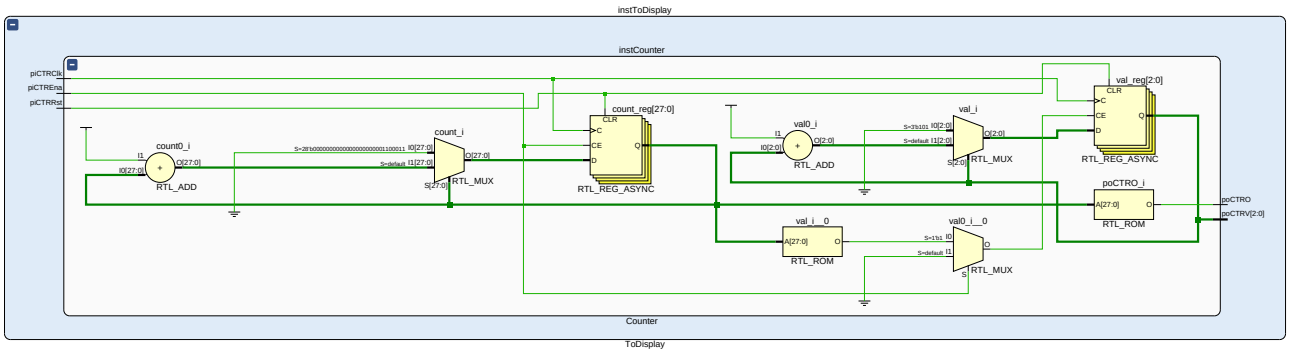


Figura 12: Esquemático del módulo **Counter**

Es utilizado por el modulo ToDisplay para rotar entre los valores a mostrar en el display de siete segmentos.

3.7.3. BaudRateGen

Genera los pulsos de sincronización de baudios para los módulos UaRx y UaTx, se preconfigura con los valores de **Max** y **First**, el primer pulso generado tras un reset es de **First** cantidad de clocks mientras que el resto es de **Max** cantidad de clocks.

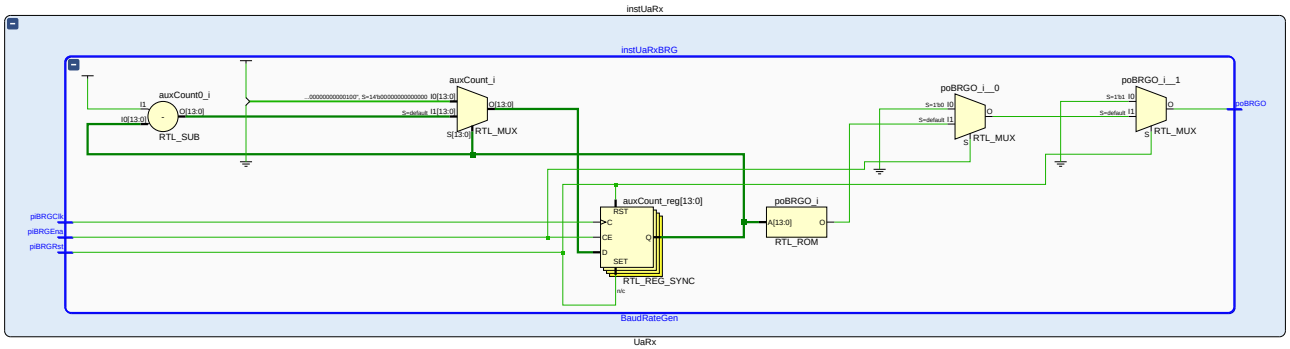


Figura 13: Esquemático del módulo **BaudRateGen**

EL modulo UaTx no requiere que el primer pulso sea de menor duración por lo que su configuración para ese caso es $Max = First$

3.7.4. TTrigger

Un contador de modulo, pero donde su salida no es un pulso sino que queda latcheada en alto cuando se alcanza el valor configurado. Se vuelve a low tras un reset y la cuenta se reinicia.

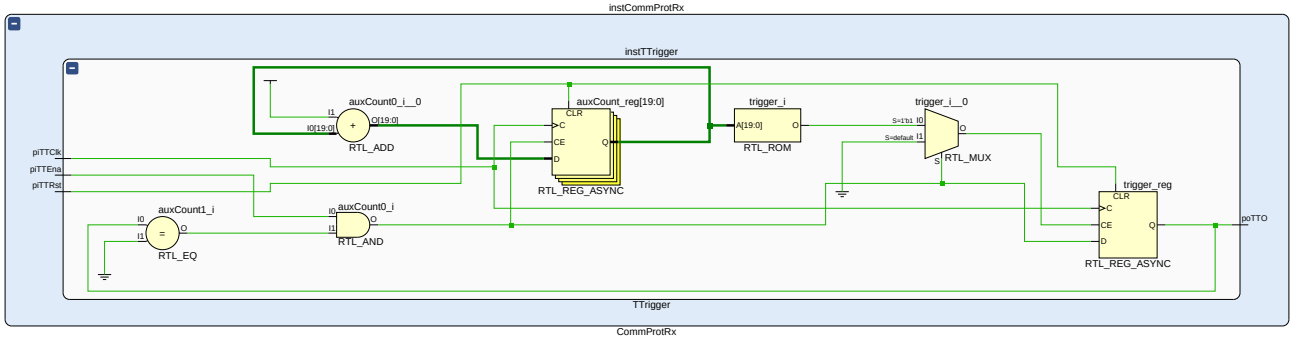


Figura 14: Esquemático del módulo **TTrigger**

Es utilizado por el modulo CommProtRx para detectar timeout.

3.7.5. PwmGen

Genera una salida PWM de periodo y resolución variable.

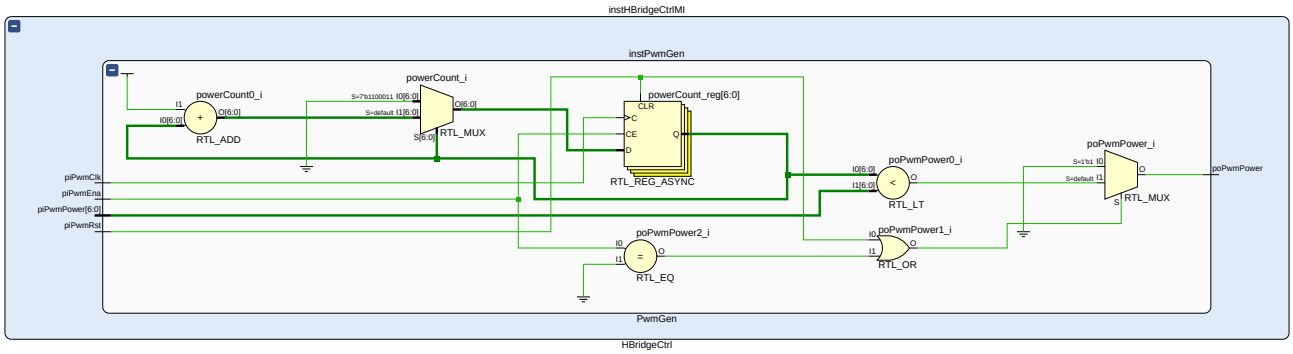


Figura 15: Esquemático del módulo **PwmGen**

3.7.6. HexToSevSeg

Decodifica un Hexadecimal de 4 bits en su correspondiente valor de 7 bits para un display de 7 segmentos.

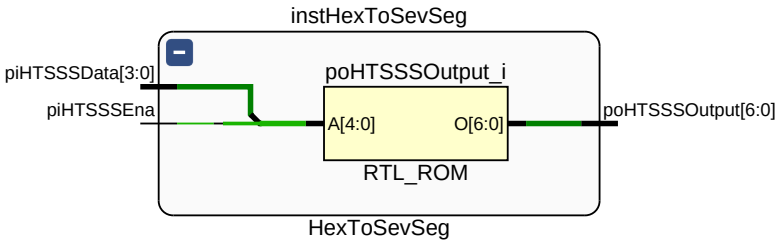


Figura 16: Esquemático del módulo **HexToSevSeg**

4. Simulación

Cada modulo se desarrollo con su propio testbench y, adicionalmente, se desarrolló un testbench para la implementación completa, de modo de visualizar el comportamiento global.

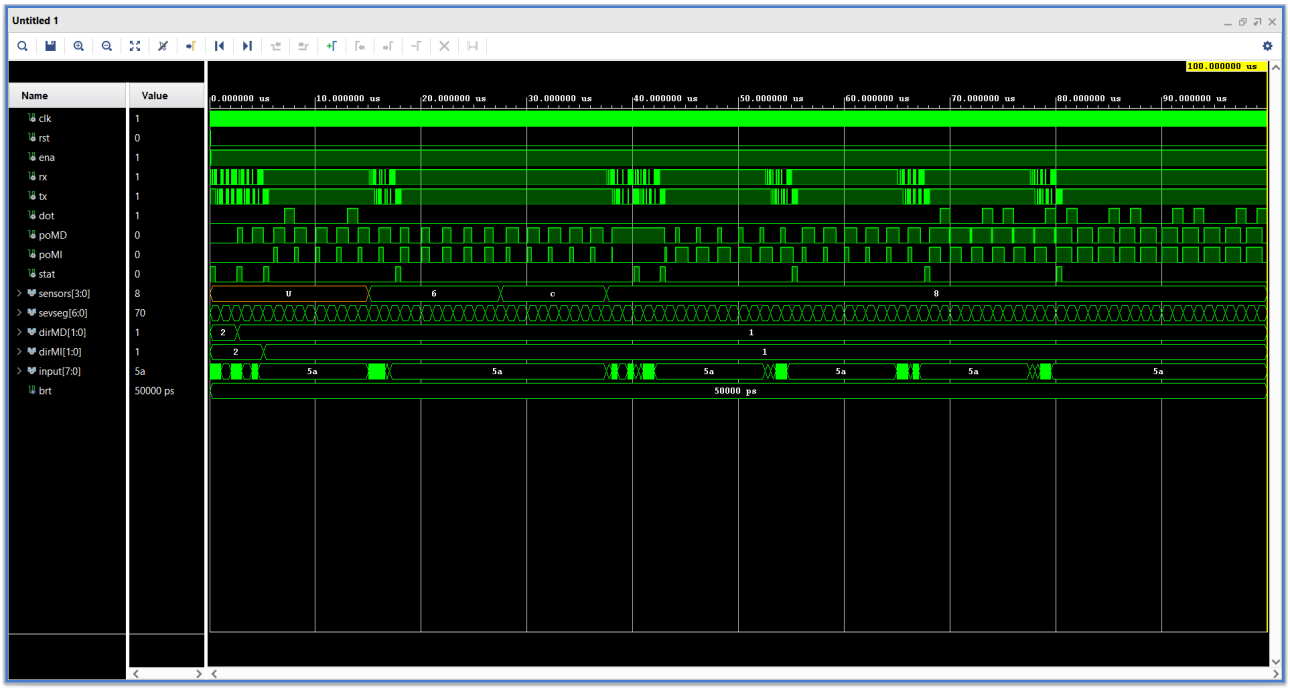


Figura 17: Simulación del sistema

La simulación consiste en enviar una serie de comandos y ver la respuesta del sistema con un tiempo de espera entre ellos, la respuesta es la esperada. Los comandos son los siguientes:

1. Seteo de la velocidad del Motor Derecho a **55 %** y velocidad Motor izquierdo al **20 %**.
2. Seteo a modo automático, aplicando velocidad media **40 %** y sensores físicos con lectura **0110**
3. Cambio en estado de los sensores a **1100**, se espera un incremento en **20 %** del PWM derecho y reducción en **10 %** del Izquierdo
4. Cambio en estado de los sensores **1000**, se espera un **100 %** en el PWM derecho y **0 %** en el izquierdo.
5. Cambio a modo de control 1: **desde PC**, control simulado de sensores a **0011**.
6. Control simulado de sensores a **1100**
7. Velocidad media del **75 %**
8. Control manual de sensores a **0110**

La siguiente imagen muestra el detalle de la simulación durante la recepción del segundo comando, que hace el seteo a modo automático, aplicando velocidad media **40 %**.

El modulo UaRx recibe los bytes mediante rs232, el modulo UaTx responde con cada byte recibido. Una vez que el modulo CommProtRx detecta un paquete de protocolo válido dispara el modulo TTrigger que mantiene en alto la salida poMIStat, conectada al led de la placa **LED0_R** y se modifica el valor de salida PWM para los dos motores

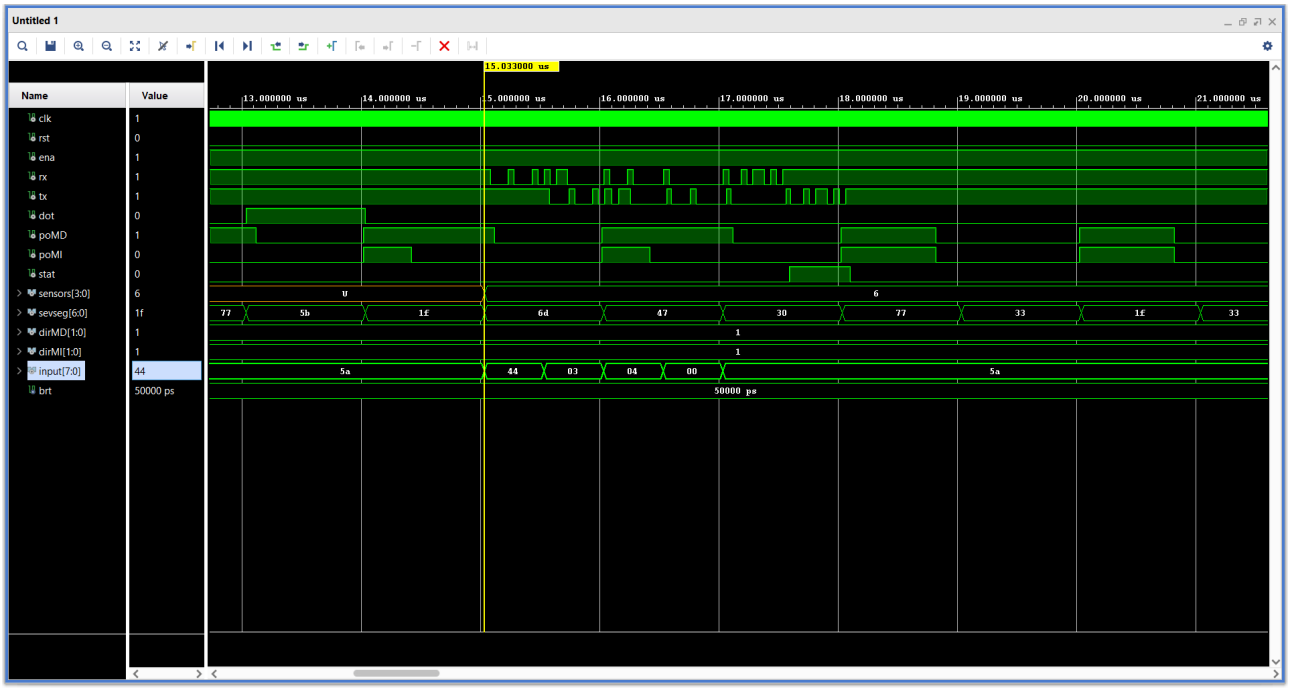


Figura 18: Simulación del sistema - Detalle de la recepción de un comando

4.1. Prueba en la placa de desarrollo Arty A7-100T

A continuación se muestra el funcionamiento sobre la placa **Arty A7-100T**, el displey muestra la velocidad del motor Derecho, que se encuentra al **10 %**

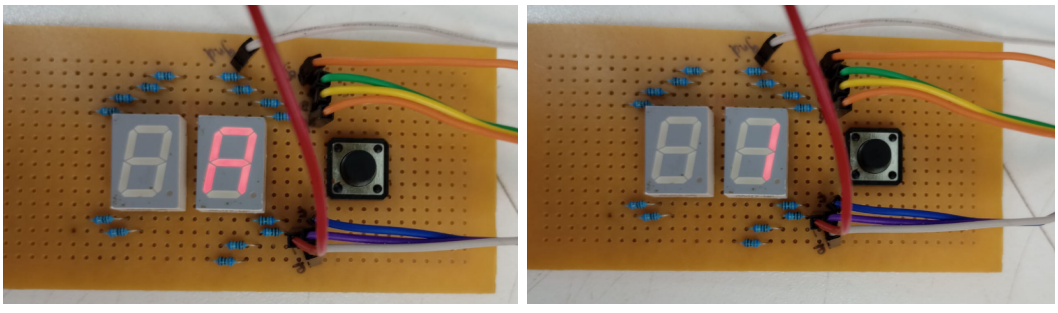


Figura 19: Velocidad del motor Derecho

En la siguiente imagen se muestra en un osciloscopio la salida PWM para el motor Derecho, con un Duty Cycle de **10 %** y una frecuencia de **100Hz**

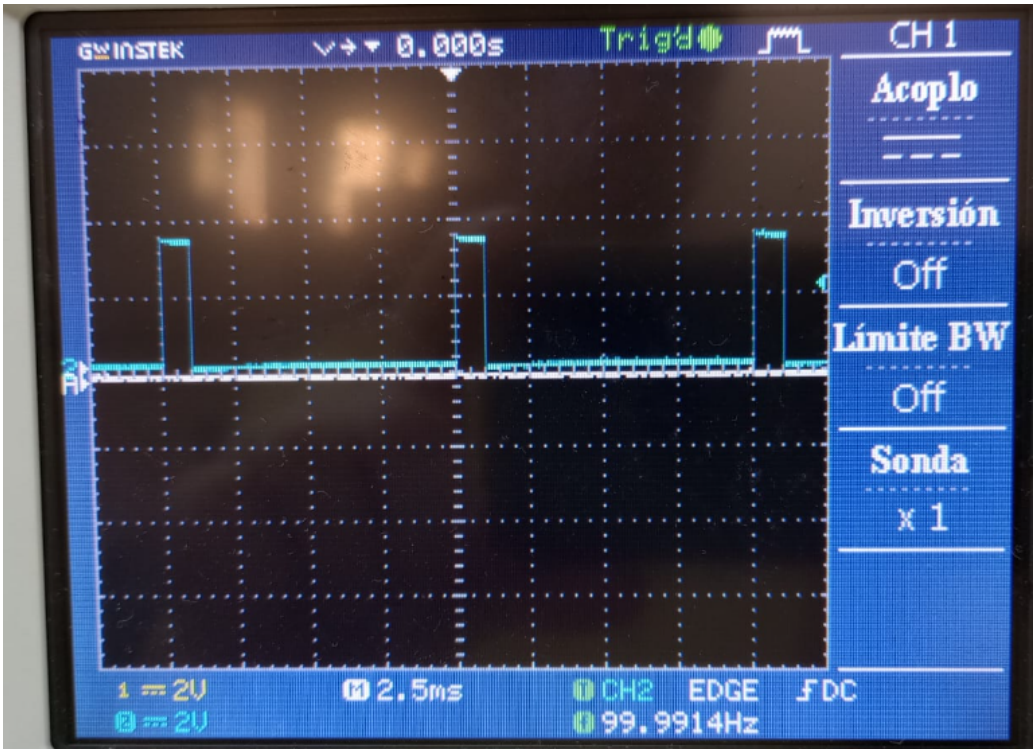


Figura 20: PWM del motor Derecho

EL código de los diferentes módulos se anexa aparte

5. Conclusiones

El desarrollo del control para el auto seguidor de línea en VHDL permitió integrar y aplicar conceptos fundamentales del diseño digital combinacional y secuencial, como el diseño y uso de módulos UART, PWM, maquinas de estado y contadores, que jugaron un papel clave en la generación de señales de control y temporización. Este proyecto consolidó el conocimiento técnico y ofreció una perspectiva práctica sobre cómo los sistemas digitales interactúan con el entorno físico.

Uno de los principales desafíos enfrentados fue la sincronización, particularmente en el módulo de recepción (RX) del UART en el lado del FPGA, donde se identificaron problemas relacionados con el manejo de señales asíncronas y la aplicación de diferentes técnicas de programación para solucionar problemas de metaestabilidad, superar los desafíos que implican la depuración en un entrono de desarrollo de este tipo y lograr asi la correcta alineación temporal de los datos recibidos

En conclusión, este proyecto cumplió con los objetivos planteados, superando desafíos técnicos significativos y proporcionando posibles mejoras a futuro, como podría ser la implementación de algoritmos de control PID en lugar de un control de valor fijo.

6. Anexos: Archivos VHDL

6.1. Archivo BaudRateGen.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 10/17/2024 05:32:34 PM
-- Design Name:
-- Module Name: BaudRateGen - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description: contador de modulo que genera un pulso de inizializacion de
--                1/2 periodo y luego continua a periodo configurado
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use ieee.numeric_std.all;

entity BaudRateGen is
    Generic (NBits: natural := 25;
             Max: natural := 25000000;        -- Periodo
             First: natural := 13500000);     -- subperiodo para el primer pulso
    Port (  piBRGClk : in STD_LOGIC;
             piBRGEna : in STD_LOGIC;
             piBRGRst : in STD_LOGIC;
             poBRGO : out STD_LOGIC
        );
end BaudRateGen;

architecture A_BaudRateGen of BaudRateGen is

signal auxCount: unsigned(NBits-1 downto 0) := to_unsigned(0, NBits);

begin

    process(auxCount, piBRGClk, piBRGEna, piBRGRst)
    begin
        if rising_edge(piBRGClk) then
           if piBRGRst = '1' then
               auxCount <= to_unsigned(First, NBits);
           elsif piBRGEna = '1' then
               if auxCount = to_unsigned(0, NBits) then
                  auxCount <= to_unsigned(Max-1, NBits);
               else
                  auxCount <= auxCount - to_unsigned(1, NBits);
               end if;
           end if;
        end if;
    end process;

    poBRGO <= '0' when piBRGRst = '1' else
              '0' when piBRGEna = '0' else
              '1' when auxCount = to_unsigned(0, NBits) else '0';


end A_BaudRateGen;
```

6.2. Archivo CommProtRx.vhd

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date: 11/01/2024 04:37:44 PM
6   -- Design Name:
7   -- Module Name: CommProtRx - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description: Protocolo de frame de ancho fijo, 1 byte de header
12  --              1 byte de trailer, sin checksum - Receiver
13  --
14  -- Dependencies:
15  --
16  -- Revision:
17  -- Revision 0.01 - File Created
18  -- Additional Comments:
19  --
20  ----------------------------------------------------------------------------------
21
22
23  library IEEE;
24  use IEEE.STD_LOGIC_1164.ALL;
25
26  -- Uncomment the following library declaration if using
27  -- arithmetic functions with Signed or Unsigned values
28  use IEEE.NUMERIC_STD.ALL;
29
30  -- Uncomment the following library declaration if instantiating
31  -- any Xilinx leaf cells in this code.
32  --library UNISIM;
33  --use UNISIM.VComponents.all;
34
35  entity CommProtRx is
36      Generic(HEADER_CHAR : NATURAL := 68;                                -- D
37              TRAILER_CHAR : NATURAL := 90;                               -- Z
38              TIMEOUT: NATURAL := 1000000);                               -- Timeout en ciclos de reloj, si no se
                ↪  completó un paquete se resetea la comunicación
39      Port (  piCPRxClk : in STD_LOGIC;                                   -- clock
40              piCPRxRst : in STD_LOGIC;                                   -- Reset
41              piCPRxEna : in STD_LOGIC;                                   -- Enable
42              piCPRxRdy : in STD_LOGIC;                                   -- Caracter de entrada listo
43              piCPRxRx : in STD_LOGIC_VECTOR(7 downto 0);                 -- Byte de entrada
44              poCPRxCmd : out STD_LOGIC_VECTOR(7 downto 0);               -- Comnado, 1 byte
45              poCPRxData : out STD_LOGIC_VECTOR(15 downto 0);             -- Dato, 2 byte
46              poCPRxC : out STD_LOGIC                                     -- Se recibió un nuevo paquete de comando
47      );
48  end CommProtRx;
49
50  architecture A_CommProtRx of CommProtRx is
51  Type TStates is (S0, S1, S2, S3, S4);
52  signal state, next_state: TStates;
53  signal cmd, r_cmd: STD_LOGIC_VECTOR(7 downto 0);
54  signal data, r_data: STD_LOGIC_VECTOR(15 downto 0);
55  signal tout: STD_LOGIC := '0';
56  signal rxrdy: STD_LOGIC := '0';
57
58  begin
59
60
61      instTTrigger: entity work.TTrigger(A_TTrigger)
62          generic map(NBits => 20, Max => TIMEOUT)
63          port map(piTTClk => piCPRxClk, piTTEna => piCPRxEna, piTTRst => piCPRxRdy, poTTO => tout);
64
65
66      SYNC_PROC : process(piCPRxClk)
67      begin
68          if rising_edge(piCPRxClk) then
69              rxrdy <= piCPRxRdy;
70              if (piCPRxRst = '1') or (tout = '1') then
71                  state <= S0;
72              else
73                  state <= next_state;
74                  r_cmd <= cmd;
75                  r_data <= data;
76              end if;
```

```
77          end if;
78      end process;
79
80
81      NEXT_STATE_DECODE : process (state, rxrdy)
82      begin
83          cmd <= r_cmd;
84          data <= r_data;
85          poCPRxC <= '0';
86          next_state <= state;
87          case (state) is
88              when S0 =>  -- Header
89                  if rxrdy = '1' then
90                      if piCPRxRx = STD_LOGIC_VECTOR(to_unsigned(HEADER_CHAR, 8)) then
91                          next_state <= S1;
92                      else
93                          next_state <= S0;
94                      end if;
95                  end if;
96              when S1 =>
97                  if rxrdy = '1' then
98                      cmd <= piCPRxRx;
99                      next_state <= S2;
100                 end if;
101             when S2 =>
102                 if rxrdy = '1' then
103                     data(15 downto 8) <= piCPRxRx;
104                     next_state <= S3;
105                 end if;
106             when S3 =>
107                 if rxrdy = '1' then
108                     data(7 downto 0) <= piCPRxRx;
109                     next_state <= S4;
110                 end if;
111             when S4 =>  --Trailer
112                 if rxrdy = '1' then
113                     if piCPRxRx = STD_LOGIC_VECTOR(to_unsigned(TRAILER_CHAR, 8)) then
114                         poCPRxC <= '1';
115                     end if;
116                     next_state <= S0;
117                 end if;
118             when others =>
119                 next_state <= S0;
120         end case;
121     end process;
122
123
124     poCPRxCmd <= r_cmd;
125     poCPRxData <= r_data;
126
127
128 end A_CommProtRx;
```

6.3. Archivo Counter.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 10/04/2024 04:14:28 PM
6  -- Design Name:
7  -- Module Name: Counter - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description: Contador con divisor
12 -- Cada Max numero de clocks va ciclando poCTRV entre 0 y NVal
13 --
14 -- Dependencies:
15 --
16 -- Revision:
17 -- Revision 0.01 - File Created
18 -- Additional Comments:
19 --
20 ----------------------------------------------------------------------------------
21
22
23 library IEEE;
24 use IEEE.STD_LOGIC_1164.ALL;
25
26 -- Uncomment the following library declaration if using
27 -- arithmetic functions with Signed or Unsigned values
28 use IEEE.NUMERIC_STD.ALL;
29
30 -- Uncomment the following library declaration if instantiating
31 -- any Xilinx leaf cells in this code.
32 --library UNISIM;
33 --use UNISIM.VComponents.all;
34
35 entity Counter is
36 Generic (NBitsMax: natural := 28;
37          NBitsVal: natural := 2;
38          Max: natural := 100000000;
39          NVal: natural := 4);
40     Port ( piCTRClk : in STD_LOGIC;
41            piCTREna : in STD_LOGIC;
42            piCTRRst : in STD_LOGIC;
43            poCTRO : out std_logic;
44            poCTRV : out std_logic_vector(NBitsVal-1 downto 0));
45 end Counter;
46
47 architecture A_Counter of Counter is
48
49 signal count : unsigned(NBitsMax-1 downto 0) := TO_UNSIGNED(0, NBitsMax);
50 signal val : unsigned(NBitsVal-1 downto 0) := TO_UNSIGNED(0, NBitsVal);
51 begin
52     process(piCTRClk, piCTREna, piCTRRst)
53     begin
54         if piCTRRst = '1' then
55             count <= TO_UNSIGNED(0, NBitsMax);
56             val <= TO_UNSIGNED(0, NBitsVal);
57         elsif rising_edge(piCTRClk) and piCTREna = '1' then
58             count <= count + TO_UNSIGNED(1, NBitsMax);
59             if count = TO_UNSIGNED(Max-1, NBitsMax) then
60                 count <= TO_UNSIGNED(0, NBitsMax);
61                 val <= val + TO_UNSIGNED(1, NBitsVal);
62                 if val = TO_UNSIGNED(NVal-1, NBitsVal) then
63                     val <= TO_UNSIGNED(0, NBitsVal);
64                 end if;
65             end if;
66         end if;
67     end process;
68
69     poCTRO <= '1' when count = TO_UNSIGNED(Max-1, NBitsMax) else '0';
70     poCTRV <= std_logic_vector(val);
71 end A_Counter;
```

6.4. Archivo DecodeCmd.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 11/08/2024 04:03:04 PM
6  -- Design Name:
7  -- Module Name: DecodeCmd - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments: Interpreta comandos de entrada y ejecuta
18 --
19 ----------------------------------------------------------------------------------
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity DecodeCmd is
35     Generic( POWER_SEL_WIDTH: NATURAL:=7;     -- Ancho en bits del selector de PWM
36              CTRL_PERIOD: NATURAL:=1000000); -- Timepo de actualización de la velocidad
37     Port (  piDCMDClk : in STD_LOGIC;
38             piDCMDRst : in STD_LOGIC;
39             piDCMDEna : in STD_LOGIC;
40             piDCMDCmdRdy: in STD_LOGIC; -- Recibido nuevo comando
41             piDCMDCmd : in STD_LOGIC_VECTOR(7 downto 0);
42             piDCMDData : in STD_LOGIC_VECTOR (15 downto 0);
43             piDCMDSensors: in STD_LOGIC_VECTOR(3 downto 0);
44             poDCMDSetMD : out STD_LOGIC;
45             poDCMDDirSelMD : out STD_LOGIC;
46             poDCMDPowerSelMD: out STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
47             poDCMDSetMI : out STD_LOGIC;
48             poDCMDDirSelMI : out STD_LOGIC;
49             poDCMDPowerSelMI: out STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
50             poDCMDMode: out STD_LOGIC
51     );
52 end DecodeCmd;
53
54 architecture A_DecodeCmd of DecodeCmd is
55
56     constant CMD_STOP: STD_LOGIC_VECTOR(7 downto 0) := STD_LOGIC_VECTOR(TO_UNSIGNED(48, 8));            -- 0x30 -> '0'
57     constant CMD_VEL_MOTOR_DER: STD_LOGIC_VECTOR(7 downto 0) := STD_LOGIC_VECTOR(TO_UNSIGNED(49, 8));   -- 0x31 -> '1'
58     constant CMD_VEL_MOTOR_IZQ: STD_LOGIC_VECTOR(7 downto 0) := STD_LOGIC_VECTOR(TO_UNSIGNED(50, 8));   -- 0x32 -> '2'
59     constant CMD_VEL_MEDIA: STD_LOGIC_VECTOR(7 downto 0) := STD_LOGIC_VECTOR(TO_UNSIGNED(51, 8));       -- 0x33 -> '3'
60     constant CMD_SIM_SENSOR: STD_LOGIC_VECTOR(7 downto 0) := STD_LOGIC_VECTOR(TO_UNSIGNED(52, 8));      -- 0x34 -> '4'
61     constant CMD_MODE: STD_LOGIC_VECTOR(7 downto 0) := STD_LOGIC_VECTOR(TO_UNSIGNED(53, 8));            -- 0x35 -> '5'
62     constant CMD_DIR: STD_LOGIC_VECTOR(7 downto 0) := STD_LOGIC_VECTOR(TO_UNSIGNED(54, 8));             -- 0x36 -> '6',
       ↪  el seteo de la direccion no se aplica en modo manual, para que se aplique hay que setear velocidad nuevamente
63     constant PC_CONTROL_MODE: STD_LOGIC := '0';
64     constant SENSORS_CONTROL_MODE: STD_LOGIC := '1';
65
66     signal power : STD_LOGIC_VECTOR(7 downto 0);
67     signal avg_power, fpower, lpower: UNSIGNED(7 downto 0);
68     signal sensors: STD_LOGIC_VECTOR(3 downto 0);
69     signal clk10ms: STD_LOGIC;
70     signal mode: STD_LOGIC := '1';  -- Modo 1: control por sensores
71     signal auto, stop, dirMD, dirMI: STD_LOGIC := '0';
72
73     -- LATCH
74     signal r_CmdRdy : STD_LOGIC; -- Recibido nuevo comando
75     signal r_Cmd : STD_LOGIC_VECTOR(7 downto 0);
76     signal r_Data :  STD_LOGIC_VECTOR (15 downto 0);
```

```
77
78  begin
79
80      -- Tiempo de actualizacion de las velocidades segun los sensores
81      instModuleCounter: entity work.ModuleCounter(A_ModuleCounter)
82          generic map(NBits => 20, Max => CTRL_PERIOD)
83          port map( piMCClk => piDCMDClk, piMCEna => '1', piMCRst => piDCMDRst, poMCO => clk10ms);
84
85
86
87      PROCESS_CMD: process(piDCMDClk, piDCMDCmdRdy, clk10ms)
88      begin
89          r_CmdRdy <= piDCMDCmdRdy;
90          r_Cmd <= piDCMDCmd;
91          r_Data <= piDCMDData;
92          if rising_edge(piDCMDClk) then
93              poDCMDSetMD <= '0';
94              poDCMDSetMI <= '0';
95              stop <= '0';
96              if r_CmdRdy = '1' then          -----    COMANDOS A EJECUTAR
97                  if r_Cmd = CMD_STOP then
98                      auto <= '0';
99                      poDCMDSetMD <= '1';
100                     poDCMDSetMI <= '1';
101                     stop <= '1';
102                 elsif r_Cmd = CMD_VEL_MOTOR_DER then
103                     auto <= '0';
104                     poDCMDSetMD <= '1';
105                 elsif r_Cmd = CMD_VEL_MOTOR_IZQ then
106                     auto <= '0';
107                     poDCMDSetMI <= '1';
108                 elsif r_Cmd = CMD_VEL_MEDIA then
109                     -- Activo mod Automatico, asigno potencia media y calculo velocidad baja y alta
110                     auto <= '1';
111                     avg_power <= UNSIGNED(power);
112                     if UNSIGNED(power) < 80 then
113                         fpower <= UNSIGNED(power) + TO_UNSIGNED(20, 8);
114                     else
115                         fpower <= TO_UNSIGNED(100, 8);
116                     end if;
117                     if UNSIGNED(power) > 20 then
118                         lpower <= UNSIGNED(power) - TO_UNSIGNED(20, 8);
119                     else
120                         lpower <= TO_UNSIGNED(0, 8);
121                     end if;
122                 elsif (r_Cmd = CMD_SIM_SENSOR) and (mode = PC_CONTROL_MODE) then
123                     sensors <= r_Data(11 downto 8);
124                 elsif r_Cmd = CMD_MODE then
125                     mode <= r_Data(8);
126                 elsif r_Cmd = CMD_DIR then
127                     dirMD <= r_Data(9);
128                     dirMI <= r_Data(8);
129                 end if;
130             else
131                 if mode = SENSORS_CONTROL_MODE then
132                     sensors <= piDCMDSensors;
133                 end if;
134             end if;
135         end if;
136
137         if auto = '1' then  --- Actualización en modo automatico
138             if rising_edge(clk10ms) then
139                 if sensors = "1100" then
140                     poDCMDSetMD <= '1';
141                     poDCMDSetMI <= '1';
142                     poDCMDPowerSelMD <= STD_LOGIC_VECTOR(fpower(6 downto 0));
143                     poDCMDPowerSelMI <= STD_LOGIC_VECTOR(lpower(6 downto 0));
144                 elsif sensors = "0011" then
145                     poDCMDSetMD <= '1';
146                     poDCMDSetMI <= '1';
147                     poDCMDPowerSelMD <= STD_LOGIC_VECTOR(lpower(6 downto 0));
148                     poDCMDPowerSelMI <= STD_LOGIC_VECTOR(fpower(6 downto 0));
149                 elsif sensors = "1000" then
150                     poDCMDSetMD <= '1';
151                     poDCMDSetMI <= '1';
152                     poDCMDPowerSelMD <= STD_LOGIC_VECTOR(TO_UNSIGNED(100, 7));
153                     poDCMDPowerSelMI <= STD_LOGIC_VECTOR(TO_UNSIGNED(0, 7));
154                 elsif sensors = "0001" then
155                     poDCMDSetMD <= '1';
```

```
156                     poDCMDSetMI <= '1';
157                     poDCMDPowerSelMD <= STD_LOGIC_VECTOR(TO_UNSIGNED(0, 7));
158                     poDCMDPowerSelMI <= STD_LOGIC_VECTOR(TO_UNSIGNED(100, 7));
159                 else
160                     poDCMDSetMD <= '1';
161                     poDCMDSetMI <= '1';
162                     poDCMDPowerSelMD <= STD_LOGIC_VECTOR(avg_power(6 downto 0));
163                     poDCMDPowerSelMI <= STD_LOGIC_VECTOR(avg_power(6 downto 0));
164                 end if;
165             end if;
166         else
167             poDCMDPowerSelMD <= STD_LOGIC_VECTOR(power(6 downto 0));
168             poDCMDPowerSelMI <= STD_LOGIC_VECTOR(power(6 downto 0));
169         end if;
170     end process PROCESS_CMD;
171
172
173
174
175     -- Calculo potencia, sea que los datos sean de potencia o no. Despues se ve si se los usa
176     power <= STD_LOGIC_VECTOR(TO_UNSIGNED(0, 8)) when stop = '1' else
177              STD_LOGIC_VECTOR(TO_UNSIGNED(0, 8)) when UNSIGNED(piDCMDData(11 downto 8)) = 0 else
178              STD_LOGIC_VECTOR(TO_UNSIGNED(100, 8)) when (UNSIGNED(piDCMDData(11 downto 8)) = 9) and
                 ↪  (UNSIGNED(piDCMDData(3 downto 0)) /= 0) else
179              STD_LOGIC_VECTOR(UNSIGNED(piDCMDData(11 downto 8)) * 10) when UNSIGNED(piDCMDData(3 downto 0)) < 5 else
180              STD_LOGIC_VECTOR(UNSIGNED(piDCMDData(11 downto 8)) * 10 + 5);
181
182     -- Direccion, seleccionable con comando CMD_DIR, bits de datos 8 y 8
183     poDCMDDirSelMD <= '1' when dirMD = '1' else '0';
184     poDCMDDirSelMI <= '1' when dirMI = '1' else '0';
185
186     poDCMDMode <= mode;
187
188 end A_DecodeCmd;
```

6.5. Archivo HBridgeCtrl.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 10/22/2024 12:58:28 PM
6  -- Design Name:
7  -- Module Name: HBridgeCtrl - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity HBridgeCtrl is
35     Generic( POWER_SEL_WIDTH: NATURAL:=7;     -- Ancho en bits del selector de PWM
36              PWM_DIV: NATURAL:=100;           -- Resolucion del PWM
37              PWM_PERIOD: NATURAL:=10000);     -- Cantidad de pulsos de clock para cada unidad de PWM_DIV - Clock 100MHz,
                 ↪  DIV 100 -> T=10000
38     Port(piHBCClk : in STD_LOGIC;
39          piHBCRst : in STD_LOGIC;
40          piHBCEna : in STD_LOGIC;
41          piHBCSet : in STD_LOGIC;
42          piHBCDirSel : in STD_LOGIC;
43          piHBCPowerSel: in STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
44          poHBCDir : out STD_LOGIC_VECTOR(1 downto 0);
45          poHBCPower : out STD_LOGIC;
46          poHBCDirSel : out STD_LOGIC;                                        -- Señal de salida conectada al valor
             ↪  del latch
47          poHBCPowerSel: out STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0)        -- Señal de salida conectada al valor
             ↪  del latch
48          );
49 end HBridgeCtrl;
50
51 architecture A_HBridgeCtrl of HBridgeCtrl is
52 signal powerSel: STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
53 signal pwmClk, dirSel: STD_LOGIC;
54 begin
55
56 instModuleCounter: entity work.ModuleCounter(A_ModuleCounter)
57     generic map(NBits => 20, Max => PWM_PERIOD)
58     port map(piMCClk => piHBCClk, piMCEna => piHBCEna, piMCRst => piHBCRst, poMCO => pwmClk);
59
60 instPwmGen: entity work.PwmGen(A_PwmGen)
61     generic map(PWM_WIDTH => 7, ARR => PWM_DIV)
62     port map(piPwmClk => pwmClk, piPwmEna => piHBCEna, piPwmRst => piHBCRst, piPwmPower => powerSel, poPwmPower =>
     ↪  poHBCPower);
63
64
65     P_SetPowerDir: process(piHBCClk, piHBCRst)
66     begin
67         if piHBCRst = '1' then
68             powerSel <= std_logic_vector(to_unsigned(0, POWER_SEL_WIDTH));
69         elsif rising_edge(piHBCClk) then
70             if piHBCSet = '1' then
71                powerSel <= piHBCPowerSel;
72                dirSel <= piHBCDirSel;
73             end if;
```

```
74            end if;
75        end process P_SetPowerDir;
76
77        poHBCDirSel <= dirSel;
78        poHBCPowerSel <= powerSel;
79        poHBCDir <= "00" when piHBCRst = '1' else
80                    "01" when dirSel = '1' else
81                    "10";
82
83    end A_HBridgeCtrl;
```

6.6. Archivo HexToSevSeg.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity HexToSevSeg is
6      port (
7          piHTSSSEna: in std_logic;
8          piHTSSSData: in std_logic_vector(3 downto 0);
9          poHTSSSOutput: out std_logic_vector(6 downto 0)
10     );
11 end entity HexToSevSeg;
12 --           A B C D E F G
13 -- 0: 0000 : 1 1 1 1 1 1 0
14 -- 1: 0001 : 0 1 1 0 0 0 0
15 -- 2: 0010 : 1 1 0 1 1 0 1
16 -- 3: 0011 : 1 1 1 1 0 0 1
17 -- 4: 0100 : 0 1 1 0 0 1 1
18 -- 5: 0101 : 1 0 1 1 0 1 1
19 -- 6: 0110 : 1 0 1 1 1 1 1
20 -- 7: 0111 : 1 1 1 0 0 0 0
21 -- 8: 1000 : 1 1 1 1 1 1 1
22 -- 9: 1001 : 1 1 1 1 0 1 1
23 -- A: 1010 : 1 1 1 0 1 1 1
24 -- B: 1011 : 0 0 1 1 1 1 1
25 -- C: 1100 : 1 0 0 1 1 1 0
26 -- D: 1101 : 0 1 1 1 1 0 1
27 -- E: 1110 : 1 0 0 1 1 1 1
28 -- F: 1111 : 1 0 0 0 1 1 1
29
30 architecture A_HexToSevSeg of HexToSevSeg is
31     signal data: STD_LOGIC_VECTOR(4 downto 0);
32 begin
33     data <= piHTSSSEna&piHTSSSData;
34     with data select
35      poHTSSSOutput <= "1111110" when "10000",
36                       "0110000" when "10001",
37                       "1101101" when "10010",
38                       "1111001" when "10011",
39                       "0110011" when "10100",
40                       "1011011" when "10101",
41                       "1011111" when "10110",
42                       "1110000" when "10111",
43                       "1111111" when "11000",
44                       "1111011" when "11001",
45                       "1110111" when "11010",
46                       "0011111" when "11011",
47                       "1001110" when "11100",
48                       "0111101" when "11101",
49                       "1001111" when "11110",
50                       "1000111" when "11111",
51                       "0000000" when others;
52 end architecture A_HexToSevSeg;
```

6.7. Archivo IMain.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 11/01/2024 04:37:44 PM
-- Design Name:
-- Module Name: IMain - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity IMain is
    Port (  piIMClk : in STD_LOGIC;              --                                                  Port E3
            piIMRst : in STD_LOGIC;              --                                                       SW1
            piIMEna : in STD_LOGIC;              --                                                       SW0
            piIMRx : in STD_LOGIC;               --                                                  Port A9
            poIMTx : out STD_LOGIC;              --                                                  Port D10
            poIMRxTest : out STD_LOGIC;               --                                                 IO38 - T18
            poIMTxTest : out STD_LOGIC;              --                                                  IO37 - U17
            piIMSensors : in STD_LOGIC_VECTOR(3 downto 0);  -- Sensores fisicos                         BTN0 - BTN3
            poIMSevSeg : out STD_LOGIC_VECTOR(6 downto 0);  -- Al display de 7 segmentos                IO32 - IO27
            poIMDot : out STD_LOGIC;                        -- Al punto del display de 7 segmentos -    IO26
            poIMPowerMD : out STD_LOGIC;                    -- Al pin Enable del L293D motor derecho -  IO41
            poIMDirMD : out STD_LOGIC_VECTOR(1 downto 0);   -- A los pin dir del L293D -                LED4 y LED5
            poIMPowerMI : out STD_LOGIC;                    -- Al pin Enable del L293D motor izquierdo - IO40
            poIMDirMI : out STD_LOGIC_VECTOR(1 downto 0);   -- A los pin dir del L293D -                LED6 y LED7
            poIMStat : out STD_LOGIC                        -- Led de estado - BLink de 200ms en CMD In LED0_R (G6)
    );
end IMain;

architecture A_IMain of IMain is

--  GLOBAL
signal clk, rst, ena : STD_LOGIC;

-- UaRX
signal rx, rxc : STD_LOGIC;
signal rxdata : STD_LOGIC_VECTOR(7 downto 0);

-- CommProtRx
signal cmdc : STD_LOGIC;
signal cmd : STD_LOGIC_VECTOR(7 downto 0);
signal data : STD_LOGIC_VECTOR(15 downto 0);

-- UaTx
signal tx, txc, txrdy : STD_LOGIC;

-- DecodeCmd
signal setMD, setMI, dirMD, dirMI, mode : STD_LOGIC;
signal powerMD, powerMI : STD_LOGIC_VECTOR(6 downto 0);

-- HBridgeCtrl
signal latchPoMD, latchPoMI : STD_LOGIC_VECTOR(6 downto 0);
signal latchDirMD, latchDirMI : STD_LOGIC;

```

```
78  -- ToDisplay
79  signal dispData : STD_LOGIC_VECTOR(3 downto 0);
80
81  -- Led status
82  signal ledstat : STD_LOGIC;
83
84  begin
85
86      instUaRx: entity work.UaRx(A_UaRx)
87          generic map(RxDIV => 10416)
88          port map( piUaRxClk => clk, piUaRxRst => rst, piUaRxEna => ena, piUaRxRx => rx, poUaRxC => rxc, poUaRxData =>
            ↪  rxdata);
89
90      instCommProtRx: entity work.CommProtRx(A_CommProtRx)
91          generic map(HEADER_CHAR => 68, TRAILER_CHAR => 90, TIMEOUT => 10000000)
92          port map( piCPRxClk => clk, piCPRxRst => rst, piCPRxEna => ena, piCPRxRdy => rxc, piCPRxRx => rxdata, poCPRxCmd
            ↪  => cmd, poCPRxData => data, poCPRxC => cmdc);
93
94      instUaTx: entity work.UaTx(A_UaTx)
95          generic map(TxDIV => 10416)
96          port map(piUaTxClk => clk, piUaTxRst => rst, piUaTxEna => ena, poUaTxTx => tx, piUaTxDataRdy => rxc, poUaTxC =>
            ↪  txc, piUaTxData => rxdata );  -- loopback
97
98      instDecodeCmd: entity work.DecodeCmd(A_DecodeCmd)
99          generic map(POWER_SEL_WIDTH => 7, CTRL_PERIOD => 1000000)
100         port map(piDCMDClk => clk, piDCMDRst => rst, piDCMDEna => ena, piDCMDCmdRdy => cmdc, piDCMDCmd => cmd, piDCMDData
            ↪  => data, piDCMDSensors => piIMSensors,
101                  poDCMDSetMD => setMD, poDCMDDirSelMD => dirMD, poDCMDPowerSelMD => powerMD, poDCMDSetMI => setMI,
                     ↪  poDCMDDirSelMI => dirMI, poDCMDPowerSelMI => powerMI, poDCMDMode => mode );
102
103     instHBridgeCtrlMD: entity work.HBridgeCtrl(A_HBridgeCtrl)
104         generic map(POWER_SEL_WIDTH => 7, PWM_DIV => 100, PWM_PERIOD => 10000)
105         port map( piHBCClk => clk, piHBCRst => rst, piHBCEna => ena, piHBCSet => setMD, piHBCDirSel => dirMD,
            ↪  piHBCPowerSel => powerMD,
106                   poHBCDir => poIMDirMD, poHBCPower => poIMPowerMD, poHBCDirSel => latchDirMD, poHBCPowerSel =>
                      ↪  latchPoMD);
107
108     instHBridgeCtrlMI: entity work.HBridgeCtrl(A_HBridgeCtrl)
109         generic map(POWER_SEL_WIDTH => 7, PWM_DIV => 100, PWM_PERIOD => 10000)
110         port map( piHBCClk => clk, piHBCRst => rst, piHBCEna => ena, piHBCSet => setMI, piHBCDirSel => dirMI,
            ↪  piHBCPowerSel => powerMI,
111                   poHBCDir => poIMDirMI, poHBCPower => poIMPowerMI, poHBCDirSel => latchDirMI, poHBCPowerSel =>
                      ↪  latchPoMI);
112
113     instToDisplay: entity work.ToDisplay(A_ToDisplay)
114         generic map( POWER_SEL_WIDTH => 7, Max => 50000000)
115         port map( piTDClk => clk, piTDRst => rst, piTDEna => ena, piTDPowerMD => latchPoMD, piTDPowerMI => latchPoMI,
            ↪  piTDMode => mode, poTDData => dispData, poTDDot => poIMDot);
116
117     instHexToSevSeg: entity work.HexToSevSeg(A_HexToSevSeg)
118         port map( piHTSSSEna => ena, piHTSSSData => dispData, poHTSSSOutput => poIMSevSeg );
119
120     instComStatus: entity work.TTrigger(A_TTrigger)
121         generic map( NBits => 24, Max => 10000000)
122         port map( piTTClk => clk, piTTEna => ena, piTTRst => cmdc, poTTO => ledstat);
123
124
125     clk <= piIMClk;
126     rst <= piIMRst;
127     ena <= piIMEna;
128     rx <= piIMRx;
129     poIMTx <= tx;
130     poIMStat <= not ledstat;
131
132     -- Test de UART
133     poIMRxTest <= piIMRx;
134     poIMTxTest <= tx;
135
136 end A_IMain;
137
```

6.8. Archivo ModuleCounter.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 10/17/2024 05:32:34 PM
-- Design Name:
-- Module Name: ModuleCounter - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use ieee.numeric_std.all;

entity ModuleCounter is
    Generic (NBits: natural := 25;
                Max: natural := 25000000);
    Port (  piMCClk : in STD_LOGIC;
            piMCEna : in STD_LOGIC;
            piMCRst : in STD_LOGIC;
            poMCO : out std_logic
                );
end ModuleCounter;

architecture A_ModuleCounter of ModuleCounter is

signal auxCount: unsigned(NBits-1 downto 0) := to_unsigned(0, NBits);

begin

    process(piMCClk, piMCEna, piMCRst)
    begin
        if piMCRst = '1' then
            auxCount <= to_unsigned(0, NBits);
        elsif rising_edge(piMCClk) then
                        if piMCEna = '1' then
                                auxCount <= auxCount + to_unsigned(1, NBits);
                                if auxCount = to_unsigned(Max-1, NBits) then
                                        auxCount <= to_unsigned(0, NBits);
                                end if;
                        end if;
        end if;
        end process;

    poMCO <= '0' when piMCRst = '1' else
             '0' when piMCEna = '0' else
             '1' when auxCount = to_unsigned(Max-1, NBits) else '0';


end A_ModuleCounter;
```

6.9. Archivo PwmGen.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 10/22/2024 01:21:18 PM
6  -- Design Name:
7  -- Module Name: PwmGen - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34
35 entity PwmGen is
36     Generic(PWM_WIDTH: NATURAL:=12;
37             ARR: NATURAL := 4096
38             );
39     Port(piPwmClk : in STD_LOGIC;
40          piPwmEna : in STD_LOGIC;
41          piPwmRst : in STD_LOGIC;
42          piPwmPower: in STD_LOGIC_VECTOR(PWM_WIDTH-1 downto 0);
43          poPwmPower : out STD_LOGIC
44          );
45 end PwmGen;
46
47 architecture A_PwmGen of PwmGen is
48
49 signal powerCount: unsigned(PWM_WIDTH-1 downto 0);
50
51 begin
52
53     pwmProcess: process(piPwmClk, piPwmEna, piPwmRst, piPwmPower)
54     begin
55         if piPwmRst = '1' then
56            powerCount <= to_unsigned(0, PWM_WIDTH);
57         elsif rising_edge(piPwmClk) and piPwmEna = '1' then
58             powerCount <= powerCount + to_unsigned(1, PWM_WIDTH);
59             if powerCount = to_unsigned(ARR-1, PWM_WIDTH) then
60                 powerCount <= to_unsigned(0, PWM_WIDTH);
61             end if;
62         end if;
63     end process pwmProcess;
64
65     poPwmPower <= '0' when (piPwmRst = '1' or piPwmEna = '0') else
66                   '1' when powerCount < unsigned(piPwmPower) else '0';
67
68
69 end A_PwmGen;
```

6.10. Archivo ToDisplay.vhd

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date: 10/22/2024 12:58:28 PM
6   -- Design Name:
7   -- Module Name: ToDisplay - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  -- Uncomment the following library declaration if using
26  -- arithmetic functions with Signed or Unsigned values
27  use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx leaf cells in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity ToDisplay is
35      Generic( POWER_SEL_WIDTH: NATURAL:=7;
36               Max: NATURAL := 100000000);    -- Tiemp de ciclo del display
37      Port(piTDClk: in STD_LOGIC;
38           piTDRst: in STD_LOGIC;
39           piTDEna: in STD_LOGIC;
40           piTDPowerMD : in STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
41           piTDPowerMI: in STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
42           piTDMode: in STD_LOGIC;
43           poTDData: out STD_LOGIC_VECTOR (3 downto 0);
44           poTDDot: out STD_LOGIC
45           );
46  end ToDisplay;
47
48  architecture A_ToDisplay of ToDisplay is
49  signal co: STD_LOGIC;
50  signal cv: STD_LOGIC_VECTOR(2 downto 0);
51  signal data, dot: UNSIGNED(POWER_SEL_WIDTH-1 downto 0);
52  begin
53
54  instCounter: entity work.Counter(A_Counter)
55      generic map(NBitsMax => 28, NBitsVal => 3, Max => Max, NVal => 6)
56      port map(piCTRClk => piTDClk, piCTREna => piTDEna, piCTRRst => piTDRst, poCTRV => cv, poCTRO => co);
57
58      process(piTDClk)
59      begin
60          if cv = "000" then  -- A
61              data <= TO_UNSIGNED(10, POWER_SEL_WIDTH);
62              dot <= TO_UNSIGNED(0, POWER_SEL_WIDTH);
63          elsif cv = "001" then
64              data <= UNSIGNED(piTDPowerMD) /10;
65              dot <= UNSIGNED(piTDPowerMD) mod 10;
66          elsif cv = "010" then  -- B
67              data <= TO_UNSIGNED(11, POWER_SEL_WIDTH);
68              dot <= TO_UNSIGNED(0, POWER_SEL_WIDTH);
69          elsif cv = "011" then
70              data <= UNSIGNED(piTDPowerMI) /10;
71              dot <= UNSIGNED(piTDPowerMI) mod 10;
72          elsif cv = "100" then -- F
73              data <= TO_UNSIGNED(15, POWER_SEL_WIDTH);
74              dot <= TO_UNSIGNED(0, POWER_SEL_WIDTH);
75          elsif cv = "101" then
76              if piTDMode = '1' then
77                  data <= TO_UNSIGNED(1, POWER_SEL_WIDTH);
```

```
78                  else
79                      data <= TO_UNSIGNED(0, POWER_SEL_WIDTH);
80                  end if;
81                  dot <= TO_UNSIGNED(0, POWER_SEL_WIDTH);
82              else   -- E
83                  data <= TO_UNSIGNED(14, POWER_SEL_WIDTH);
84                  dot <= TO_UNSIGNED(0, POWER_SEL_WIDTH);
85              end if;
86          end process;
87
88          poTDData <= STD_LOGIC_VECTOR(data(3 downto 0));
89          poTDdot <= '0' when dot = TO_UNSIGNED(0, POWER_SEL_WIDTH) else '1';
90
91      end A_ToDisplay;
92
```

6.11. Archivo TTrigger.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 10/17/2024 05:32:34 PM
-- Design Name:
-- Module Name: TTrigger - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use ieee.numeric_std.all;

entity TTrigger is
    Generic (NBits: natural := 25;
                Max: natural := 25000000);
    Port (  piTTClk : in STD_LOGIC;
            piTTEna : in STD_LOGIC;
            piTTRst : in STD_LOGIC;
            poTTO : out std_logic
                );
end TTrigger;

architecture A_TTrigger of TTrigger is

signal auxCount: unsigned(NBits-1 downto 0) := TO_UNSIGNED(0, NBits);
signal trigger: STD_LOGIC := '0';

begin

    process(piTTClk, piTTEna, piTTRst, trigger)
    begin
        if piTTRst = '1' then
            auxCount <= to_unsigned(0, NBits);
            trigger <= '0';
        elsif rising_edge(piTTClk) then
                        if (piTTEna = '1') and (trigger = '0') then
                                auxCount <= auxCount + to_unsigned(1, NBits);
                                if auxCount = to_unsigned(Max-1, NBits) then
                                        trigger <= '1';
                                end if;
                        end if;
        end if;
        end process;

    poTTO <= trigger;


end A_TTrigger;
```

6.12. Archivo Uart.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 11/08/2024 04:03:04 PM
6  -- Design Name:
7  -- Module Name: Uart - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 -- Solo recepcion, rs232, sin paridad, sin flow control, 8 bits
19 --
20 ----------------------------------------------------------------------------------
21
22
23 library IEEE;
24 use IEEE.STD_LOGIC_1164.ALL;
25
26 -- Uncomment the following library declaration if using
27 -- arithmetic functions with Signed or Unsigned values
28 use IEEE.NUMERIC_STD.ALL;
29
30 -- Uncomment the following library declaration if instantiating
31 -- any Xilinx leaf cells in this code.
32 --library UNISIM;
33 --use UNISIM.VComponents.all;
34
35 entity Uart is
36     Generic ( DIV: NATURAL:= 10417); -- 100Mhz/DIV -> 9600 baud
37     Port (  piUartClk : in STD_LOGIC; -- Clock de entrada
38             piUartRst : in STD_LOGIC; -- Reset
39
40             piUartRxEna : in STD_LOGIC; -- RX Enable
41             piUartTxEna : in STD_LOGIC; -- TX Enable
42
43             piUartRx : in STD_LOGIC;    -- Puerto RX
44             poUartRxC: out STD_LOGIC; -- Receive complete - Hay datos para leer en el buffer poUartRxData
45             poUartRxData : out STD_LOGIC_VECTOR (8-1 downto 0);
46
47             piUartTxDataRdy : in STD_LOGIC;   -- Transmit ready - Los datos en piUartTxData estan listos para ser
               ↪  enviados
48             piUartTxData : in STD_LOGIC_VECTOR (8-1 downto 0);
49             poUartTx : out STD_LOGIC;    -- Puerto TX
50             poUartTxC : out STD_LOGIC   -- Transmit Complete - Los datos en piUartTxData fueron enviados
51     );
52 end Uart;
53
54 architecture A_Uart of Uart is
55
56 begin
57
58     instUaRx: entity work.UaRx(A_UaRx)
59         generic map(RxDIV => DIV)
60         port map(piUaRxClk => piUartClk, piUaRxRst => piUartRst, piUaRxEna => piUartRxEna, piUaRxRx => piUartRx,
           ↪  poUaRxData => poUartRxData, poUaRxC => poUartRxC);
61
62     instUaTx: entity work.UaTx(A_UaTx)
63         generic map(TxDIV => DIV)
64         port map(piUaTxClk => piUartClk, piUaTxRst => piUartRst, piUaTxEna => piUartTxEna, poUaTxTx => poUartTx,
           ↪  piUaTxData => piUartTxData, piUaTxDataRdy => piUartTxDataRdy, poUaTxC => poUartTxC);
65
66 end A_Uart;
```

6.13. Archivo UaRx.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 11/08/2024 04:03:04 PM
-- Design Name:
-- Module Name: UaRx - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- Solo recepcion, rs232, sin paridad, sin flow control, 8 bits
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity UaRx is
    Generic ( RxDIV: NATURAL:= 10417); -- 100Mhz/DIV -> 9600 baud
    Port (  piUaRxClk : in STD_LOGIC; -- Clock de entrada
            piUaRxRst : in STD_LOGIC; -- Reset
            piUaRxEna : in STD_LOGIC; -- RX Enable
            piUaRxRx : in STD_LOGIC;    -- Puerto RX
            poUaRxC: out STD_LOGIC; -- Receive complete - Hay datos para leer en el buffer poUaRxData
            poUaRxData : out STD_LOGIC_VECTOR (8-1 downto 0)
    );
end UaRx;

architecture A_UaRx of UaRx is

Type TStates is (S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10);
signal state, next_state: TStates;
signal brgrst: STD_LOGIC := '1';
signal brgclk: STD_LOGIC;
signal synlatch, latch: STD_LOGIC_VECTOR(8-1 downto 0);


begin

   instUaRxBRG: entity work.BaudRateGen(A_BaudRateGen)
      generic map( NBits => 14, Max => RxDIV, First => RxDIV/2)
      port map(piBRGClk => piUaRxClk, piBRGEna => '1', piBRGRst => brgrst, poBRGO => brgclk);

    SYNC_PROC : process (piUaRxClk, piUaRxRst)
    begin
        if rising_edge(piUaRxClk) then
            if piUaRxRst = '1' then
                state <= S0;
            else
                state <= next_state;
                synlatch <= latch;
            end if;
        end if;
    end process SYNC_PROC;


    NEXT_STATE_DECODE : process (state, brgclk, piUaRxRx, piUaRxEna)
    begin
        next_state <= state;
        poUaRxC <= '0';
```

```
78          brgrst <= '0';
79          latch <= synlatch;
80          case (state) is
81              when S0 =>  -- Espera bit start
82                 if (piUaRxRx = '0') and (piUaRxEna = '1') then
83                      brgrst <= '1';
84                      next_state <= S1;
85                  end if;
86              when S1 =>  -- Chequeo que es bit start tras medio periodo
87                 if brgclk = '1' then
88                    if piUaRxRx = '0' then
89                       next_state <= S2;
90                    else
91                       next_state <= S0;
92                    end if;
93                 end if;
94              when S2 => -- recepcion de datos - bit 0
95                  if brgclk = '1' then
96                      latch(0) <= piUaRxRx;
97                      next_state <= S3;
98                  end if;
99              when S3 => -- recepcion de datos - bit 1
100                 if brgclk = '1' then
101                     latch(1) <= piUaRxRx;
102                     next_state <= S4;
103                 end if;
104             when S4 => -- recepcion de datos - bit 2
105                 if brgclk = '1' then
106                     latch(2) <= piUaRxRx;
107                     next_state <= S5;
108                 end if;
109             when S5 => -- recepcion de datos - bit 3
110                 if brgclk = '1' then
111                     latch(3) <= piUaRxRx;
112                     next_state <= S6;
113                 end if;
114             when S6 => -- recepcion de datos - bit 4
115                 if brgclk = '1' then
116                     latch(4) <= piUaRxRx;
117                     next_state <= S7;
118                 end if;
119             when S7 => -- recepcion de datos - bit 5
120                 if brgclk = '1' then
121                     latch(5) <= piUaRxRx;
122                     next_state <= S8;
123                 else
124                     next_state <= S7;
125                 end if;
126             when S8 => -- recepcion de datos - bit 6
127                 if brgclk = '1' then
128                     latch(6) <= piUaRxRx;
129                     next_state <= S9;
130                 end if;
131             when S9 => -- recepcion de datos - bit 7
132                 if brgclk = '1' then
133                     latch(7) <= piUaRxRx;
134                     next_state <= S10;
135                 end if;
136             when S10 => -- recepcion del bit de stop
137                if (brgclk = '1') then
138                     next_state <= S0;
139                     if (piUaRxRx = '1') then
140                         poUaRxC <= '1';
141                     end if;
142                end if;
143             when others =>
144                 next_state <= S0;
145         end case;
146     end process NEXT_STATE_DECODE;
147
148     poUaRxData <= synlatch;
149
150 end A_UaRx;
```

6.14. Archivo UaTx.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 11/08/2024 04:03:04 PM
-- Design Name:
-- Module Name: UaTx - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
-- Solo recepcion, rs232, sin paridad, sin flow control, 8 bits
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity UaTx is
    Generic ( TxDIV: NATURAL:= 10417); -- 100Mhz/DIV -> 9600 baud
    Port (  piUaTxClk : in STD_LOGIC; -- Clock de entrada
            piUaTxRst : in STD_LOGIC; -- Reset
            piUaTxEna : in STD_LOGIC; -- TX Enable

            poUaTxTx : out STD_LOGIC;     -- Puerto TX
            piUaTxDataRdy : in STD_LOGIC;    -- Transmit ready - Los datos en piUaTxData estan listos para ser enviados
            poUaTxC : out STD_LOGIC;    -- Transmit Complete - Los datos en piUaTxData fueron enviados
            piUaTxData : in STD_LOGIC_VECTOR (8-1 downto 0)
    );
end UaTx;

architecture A_UaTx of UaTx is

Type TStates is (S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11);
signal state, next_state: TStates;
signal brgrst: STD_LOGIC;  -- reset del Baud rate generator
signal brgclk: STD_LOGIC;  -- Clock del Baud rate Generator
signal txrdy: STD_LOGIC := '0';  -- Solo cuando piUaTxDataRdy en 1 durante el bit stop, activa un nuevo envio
↪  inmediatamente antes de terminar
signal latch: STD_LOGIC_VECTOR(8-1 downto 0);

begin

   instUaTxBRG: entity work.BaudRateGen(A_BaudRateGen)
      generic map( NBits => 14, Max => TxDIV, First => TxDIV)
      port map(piBRGClk => piUaTxClk, piBRGEna => '1', piBRGRst => brgrst, poBRGO => brgclk);

    SYNC_PROC : process (piUaTxClk)
    begin
        if rising_edge(piUaTxClk) then
            if (piUaTxRst = '1') then
                state <= S0;
            else
                state <= next_state;
            end if;
        end if;
    end process;


    OUTPUT_DECODE : process (state)
    begin
```

```
77      brgrst <= '0';
78      poUaTxC <= '0';
79      case (state) is
80          when S0 =>
81              brgrst <= '1';
82          when S11 =>
83              poUaTxC <= '1';
84          when others =>
85      end case;
86      end process OUTPUT_DECODE;
87
88
89      RX_NEXT_STATE_DECODE : process (state, brgclk, piUaTxEna, piUaTxDataRdy)
90      begin
91          poUaTxTx <= '1';
92          case (state) is
93              when S0 =>  -- En espera
94                  if (txrdy = '1' or piUaTxDataRdy = '1') and (piUaTxEna = '1') then
95                      next_state <= S1;
96                  else
97                      next_state <= S0;
98                  end if;
99              when S1 => -- envio de datos - bit start
100                 poUaTxTx <= '0';
101                 latch <= piUaTxData;
102                 txrdy <= '0';
103                 if brgclk = '1' then
104                     next_state <= S2;
105                 else
106                     next_state <= S1;
107                 end if;
108             when S2 => -- envio de datos - bit 0
109                 poUaTxTx <= latch(0);
110                 if brgclk = '1' then
111                     next_state <= S3;
112                 else
113                     next_state <= S2;
114                 end if;
115             when S3 => -- envio de datos - bit 1
116                 poUaTxTx <= latch(1);
117                 if brgclk = '1' then
118                     next_state <= S4;
119                 else
120                     next_state <= S3;
121                 end if;
122             when S4 => -- envio de datos - bit 2
123                 poUaTxTx <= latch(2);
124                 if brgclk = '1' then
125                     next_state <= S5;
126                 else
127                     next_state <= S4;
128                 end if;
129             when S5 => -- envio de datos - bit 3
130                 poUaTxTx <= latch(3);
131                 if brgclk = '1' then
132                     next_state <= S6;
133                 else
134                     next_state <= S5;
135                 end if;
136             when S6 => -- envio de datos - bit 4
137                 poUaTxTx <= latch(4);
138                 if brgclk = '1' then
139                     next_state <= S7;
140                 else
141                     next_state <= S6;
142                 end if;
143             when S7 => -- envio de datos - bit 5
144                 poUaTxTx <= latch(5);
145                 if brgclk = '1' then
146                     next_state <= S8;
147                 else
148                     next_state <= S7;
149                 end if;
150             when S8 => -- envio de datos - bit 6
151                 poUaTxTx <= latch(6);
152                 if brgclk = '1' then
153                     next_state <= S9;
154                 else
155                     next_state <= S8;
```

```
                end if;
            when S9 => -- envio de datos - bit 7
                poUaTxTx <= latch(7);
                if brgclk = '1' then
                    next_state <= S10;
                else
                    next_state <= S9;
                end if;
            when S10 => -- envio del bit de stop
               if (brgclk = '1') then
                   next_state <= S11;
               elsif piUaTxDataRdy = '1' then
                   txrdy <= '1';
                   next_state <= S11;
               else
                   next_state <= S10;
               end if;
            when S11 => -- Se finalizó la transmisión, poUaTxC durante 1 clock de sistema
               next_state <= S0;
            when others =>
                next_state <= S0;
        end case;
    end process;


end A_UaTx;
```

7. Anexos: Archivos de TESTBENCH

7.1. Archivo BaudRateGen_tb.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 10/17/2024 05:36:16 PM
-- Design Name:
-- Module Name: BaudRateGen_tb - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;

use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity BaudRateGen_TB is
end BaudRateGen_TB;


architecture A_BaudRateGen_TB of BaudRateGen_TB is

component BaudRateGen is
    Generic (NBits: natural;
             Max: natural;
             First: natural);
    Port (  piBRGClk : in STD_LOGIC;
            piBRGEna : in STD_LOGIC;
            piBRGRst : in STD_LOGIC;
            poBRGO : out STD_LOGIC
        );
end component BaudRateGen;

signal clk, ena, rst, x: std_logic;

```

```
44  begin
45
46     instBRG: BaudRateGen
47        generic map( NBits => 8, Max => 4, First => 2)
48        port map(piBRGClk => clk, piBRGEna => ena, piBRGRst => rst, poBRGO => x);
49
50     pClk: process
51     begin
52        clk <= '1';
53        wait for 10 ns;
54        clk <= '0';
55        wait for 10 ns;
56     end process;
57
58
59     process
60     begin
61        rst <= '1';
62        ena <= '0';
63        wait for 40 ns;
64        rst <= '0';
65        wait until falling_edge(clk);
66        ena <= '1';
67        wait for 500 ns;
68        wait until falling_edge(clk);
69        rst <= '1';
70        wait until falling_edge(clk);
71        rst <= '0';
72        wait for 500 ns;
73        ena <= '0';
74        wait;
75     end process;
76
77
78  end A_BaudRateGen_TB;
```

7.2. Archivo CommProtRx_tb.vhd

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date: 11/08/2024 04:04:23 PM
6   -- Design Name:
7   -- Module Name: CommProtRx_TB - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  -- Uncomment the following library declaration if using
26  -- arithmetic functions with Signed or Unsigned values
27  use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx leaf cells in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity CommProtRx_TB is
35  --  Port ( );
36  end CommProtRx_TB;
37
38  architecture Behavioral of CommProtRx_TB is
39     component CommProtRx is
40      Generic(HEADER_CHAR : NATURAL := 68;                              -- D
41              TRAILER_CHAR : NATURAL := 90;                             -- Z
42              TIMEOUT: NATURAL := 1000000);                             -- Timeout en ciclos de reloj, si no se
                ↪  completó un paquete se resetea la comunicación
43      Port (  piCPRxClk : in STD_LOGIC;                                 -- clock
44              piCPRxRst : in STD_LOGIC;                                 -- Reset
45              piCPRxEna : in STD_LOGIC;                                 -- Enable
46              piCPRxRdy : in STD_LOGIC;                                 -- Caracter de entrada listo
47              piCPRxRx : in STD_LOGIC_VECTOR(7 downto 0);               -- Byte de entrada
48              poCPRxCmd : out STD_LOGIC_VECTOR(7 downto 0);             -- Comnado, 1 byte
49              poCPRxData : out STD_LOGIC_VECTOR(15 downto 0);           -- Dato, 2 byte
50              poCPRxC : out STD_LOGIC                                   -- Se recibió un nuevo paquete de comando
51      );
52     end component CommProtRx;
53
54     signal clk, rst, ena, c, rdy, rdyena: STD_LOGIC;
55     signal data: STD_LOGIC_VECTOR(15 downto 0);
56     signal cmd, rx: STD_LOGIC_VECTOR(7 downto 0);
57  begin
58
59         instCommProtRx: CommProtRx
60         generic map(HEADER_CHAR => 68,
61                     TRAILER_CHAR => 90,
62                     TIMEOUT => 10)
63         Port map ( piCPRxClk => clk,
64              piCPRxRst => rst,
65              piCPRxRdy => rdy,
66              piCPRxEna => ena,
67              piCPRxRx => rx,
68              poCPRxCmd => cmd,
69              poCPRxC => c,
70              poCPRxData => data);
71
72     pClk: process
73          begin
74                  clk <= '1';
75                  wait for 5 ns;
76                  clk <= '0';
```

```
77                  wait for 5 ns;
78          end process;
79
80
81      pRdy: process
82          begin
83                  rdy <= rdyena;
84                  wait for 10 ns;
85                  rdy <= '0';
86                  wait for 33 ns;
87          end process;
88
89
90       process
91       begin
92          rst <= '1';
93          ena <= '0';
94          rdyena <= '1';
95          wait for 63 ns;
96          rst <= '0';
97          ena <= '1';
98
99          wait until falling_edge(rdy);
100         rx <= STD_LOGIC_VECTOR(to_unsigned(68, 8)); -- Header
101         wait until falling_edge(rdy);
102         rx <= "00001110";  -- CMD
103         wait until falling_edge(rdy);
104         rx <= "00110011";  -- Data
105         wait until falling_edge(rdy);
106         rx <= "01010101";  -- Data
107         wait until falling_edge(rdy);
108         rx <= STD_LOGIC_VECTOR(to_unsigned(90, 8)); -- Trailer
109         wait until falling_edge(rdy);
110         wait;
111
112
113         -- Worng header
114         wait until falling_edge(rdy);
115         rx <= STD_LOGIC_VECTOR(to_unsigned(37, 8)); -- Header
116         wait until falling_edge(rdy);
117         rx <= "00001101";  -- CMD
118         wait until falling_edge(rdy);
119         rx <= "00110011";  -- Data
120         wait until falling_edge(rdy);
121         rx <= "01010101";  -- Data
122         wait until falling_edge(rdy);
123         rx <= STD_LOGIC_VECTOR(to_unsigned(90, 8)); -- Trailer
124
125         -- OK
126         wait until falling_edge(rdy);
127         rx <= STD_LOGIC_VECTOR(to_unsigned(68, 8)); -- Header
128         wait until falling_edge(rdy);
129         rx <= "00001111";  -- CMD
130         wait until falling_edge(rdy);
131         rx <= "00000011";  -- Data
132         wait until falling_edge(rdy);
133         rx <= "01111111";  -- Data
134         wait until falling_edge(rdy);
135         rx <= STD_LOGIC_VECTOR(to_unsigned(90, 8)); -- Trailer
136
137
138         -- Timeout
139         wait until falling_edge(rdy);
140         rx <= STD_LOGIC_VECTOR(to_unsigned(68, 8)); -- Header
141         wait until falling_edge(rdy);
142         rx <= "00011111";  -- CMD
143         wait until falling_edge(rdy);
144         rx <= "00000011";  -- Data
145         rdyena <= '0';
146         wait for 150 ns;
147         rdyena <= '1';
148         wait until falling_edge(rdy);
149         rx <= "00000000";  -- Data
150         wait until falling_edge(rdy);
151         rx <= STD_LOGIC_VECTOR(to_unsigned(90, 8)); -- Trailer
152
153
154         -- OK
155         wait until falling_edge(rdy);
```

```
156         rx <= STD_LOGIC_VECTOR(to_unsigned(68, 8)); -- Header
157         wait until falling_edge(rdy);
158         rx <= "00001011";  -- CMD
159         wait until falling_edge(rdy);
160         rx <= "00000001";  -- Data
161         wait until falling_edge(rdy);
162         rx <= "11111111";  -- Data
163         wait until falling_edge(rdy);
164         rx <= STD_LOGIC_VECTOR(to_unsigned(90, 8)); -- Trailer
165 
166         -- Worng trailer
167         wait until falling_edge(rdy);
168         rx <= STD_LOGIC_VECTOR(to_unsigned(68, 8)); -- Header
169         wait until falling_edge(rdy);
170         rx <= "00000111";  -- CMD
171         wait until falling_edge(rdy);
172         rx <= "00110011";  -- Data
173         wait until falling_edge(rdy);
174         rx <= "01010101";  -- Data
175         wait until falling_edge(rdy);
176         rx <= STD_LOGIC_VECTOR(to_unsigned(50, 8)); -- Trailer
177 
178         wait for 33 ns;
179 
180         wait;
181 
182     end process;
183 
184 end Behavioral;
```

7.3. Archivo Counter_tb.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 10/22/2024 05:41:18 PM
-- Design Name:
-- Module Name: Counter_tb - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Counter_tb is
--  Port ( );
end Counter_tb;

architecture Behavioral of Counter_tb is

component Counter is
    Generic (NBitsMax: natural := 28;
         NBitsVal: natural := 2;
         Max: natural := 100000000;
         NVal: natural := 4);
    Port ( piCTRClk : in STD_LOGIC;
           piCTREna : in STD_LOGIC;
           piCTRRst : in STD_LOGIC;
           poCTRO : out std_logic;
           poCTRV : out std_logic_vector(NBitsVal-1 downto 0));
end component;

signal Clk, Ena, Rst, o : std_logic;
signal v: std_logic_vector(2-1 downto 0);


begin
    instCounter: Counter
    generic map( NBitsMax => 3,
                 NBitsVal => 2,
                 Max => 5,
                 NVal => 3)
    port map(piCTRClk => Clk,
             piCTREna => Ena,
             piCTRRst => Rst,
             poCTRO => o,
             poCTRV => v
     );

    pClk: process
    begin
        Clk <= '1';
        wait for 5 ns;
        Clk <= '0';
        wait for 5 ns;
    end process;

    process
```

```
78      begin
79          Rst <= '1';
80          Ena <= '0';
81          wait until falling_edge(Clk);
82
83          Rst <= '0';
84          ena <= '1';
85          wait for 100 ns;
86          ena <= '0';
87          wait for 100 ns;
88          ena <= '1';
89          wait;
90       end process;
91
92
93  end Behavioral;
94
```

7.4. Archivo DecodeCmd_tb.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 11/08/2024 04:04:23 PM
6  -- Design Name:
7  -- Module Name: DecodeCmd_TB - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity DecodeCmd_TB is
35 --  Port ( );
36 end DecodeCmd_TB;
37
38 architecture Behavioral of DecodeCmd_TB is
39    component DecodeCmd is
40     Generic( POWER_SEL_WIDTH: NATURAL:=7;     -- Ancho en bits del selector de PWM
41              CTRL_PERIOD: NATURAL:=1000000); -- Timepo de actualización de la velocidad
42     Port (  piDCMDClk : in STD_LOGIC;
43             piDCMDRst : in STD_LOGIC;
44             piDCMDEna : in STD_LOGIC;
45             piDCMDCmdRdy: in STD_LOGIC; -- Recibido nuevo comando
46             piDCMDCmd : in STD_LOGIC_VECTOR(7 downto 0);
47             piDCMDData : in STD_LOGIC_VECTOR (15 downto 0);
48             piDCMDSensors: in STD_LOGIC_VECTOR(3 downto 0);
49             poDCMDSetMD : out STD_LOGIC;
50             poDCMDDirSelMD : out STD_LOGIC;
51             poDCMDPowerSelMD: out STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
52             poDCMDSetMI : out STD_LOGIC;
53             poDCMDDirSelMI : out STD_LOGIC;
54             poDCMDPowerSelMI: out STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
55             poDCMDMode: out STD_LOGIC
56     );
57    end component DecodeCmd;
58
59    signal clk, rst, ena, cmdrdy, setMD, setMI, dirMD, dirMI, mode: STD_LOGIC;
60    signal cmd: STD_LOGIC_VECTOR(7 downto 0);
61    signal data: STD_LOGIC_VECTOR(15 downto 0);
62    signal poMD: STD_LOGIC_VECTOR(6 downto 0);
63    signal poMI: STD_LOGIC_VECTOR(6 downto 0);
64    signal sensors: STD_LOGIC_VECTOR(3 downto 0);
65 begin
66
67        instDecodeCmd: DecodeCmd
68        generic map(POWER_SEL_WIDTH => 7, CTRL_PERIOD => 4)
69        Port map ( piDCMDClk => clk,
70                                   piDCMDRst => rst,
71                                   piDCMDEna => ena,
72                                   piDCMDCmdRdy => cmdrdy,
73                                   piDCMDCmd => cmd,
74                                   piDCMDData => data,
75                                   piDCMDSensors=> sensors,
76                                   poDCMDSetMD => setMD,
77                                   poDCMDDirSelMD => dirMD,
```

```
78                                      poDCMDPowerSelMD => poMD,
79                                      poDCMDSetMI => setMI,
80                                      poDCMDDirSelMI => dirMI,
81                                      poDCMDPowerSelMI => poMI,
82                                      poDCMDMode => mode
83                                      );
84
85      pClk: process
86           begin
87                   clk <= '1';
88                   wait for 5 ns;
89                   clk <= '0';
90                   wait for 5 ns;
91           end process;
92
93
94       process
95       begin
96          rst <= '1';
97          ena <= '0';
98          cmdrdy <= '0';
99          wait for 33 ns;
100         rst <= '0';
101         ena <= '1';
102         sensors <= "0110";
103
104         -- Set motor D manual
105         wait until falling_edge(clk);
106         cmdrdy <= '1';
107         cmd <= "00110001";
108         data <= "0000010100000101";
109         wait until falling_edge(clk);
110         cmdrdy <= '0';
111
112         -- Set motor D manual
113         wait until falling_edge(clk);
114         cmdrdy <= '1';
115         cmd <= "00110010";
116         data <= "0000001100000000";
117         wait until falling_edge(clk);
118         cmdrdy <= '0';
119
120         -- Set velocidad media auto
121         wait until falling_edge(clk);
122         cmdrdy <= '1';
123         cmd <= "00110011";
124         data <= "0000010100000101";
125         wait until falling_edge(clk);
126         cmdrdy <= '0';
127
128         -- Set stop
129         wait until falling_edge(clk);
130         cmdrdy <= '1';
131         cmd <= "00110000";
132         data <= "0000000000000000";
133         wait until falling_edge(clk);
134         cmdrdy <= '0';
135
136         -- Set auto
137         wait until falling_edge(clk);
138         cmdrdy <= '1';
139         cmd <= "00110011";
140         data <= "0000010100000101";
141         wait until falling_edge(clk);
142         cmdrdy <= '0';
143
144         -- Sensors
145         sensors <= "1100";
146         wait for 100 ns;
147         sensors <= "1000";
148         wait for 100 ns;
149
150         -- Set From PC
151         wait until falling_edge(clk);
152         cmdrdy <= '1';
153         cmd <= "00110101";
154         data <= "0000000000000000";
155         wait until falling_edge(clk);
156         cmdrdy <= '0';
```

```
157
158         -- Set sensors from PC
159         wait for 100 ns;
160         wait until falling_edge(clk);
161         cmdrdy <= '1';
162         cmd <= "00110100";
163         data <= "0000110000000000";
164         wait until falling_edge(clk);
165         cmdrdy <= '0';
166
167         -- Set sensors from PC
168         wait for 100 ns;
169         wait until falling_edge(clk);
170         cmdrdy <= '1';
171         cmd <= "00110100";
172         data <= "0000001100000000";
173         wait until falling_edge(clk);
174         cmdrdy <= '0';
175
176         -- Set sensors from PC
177         wait for 100 ns;
178         wait until falling_edge(clk);
179         cmdrdy <= '1';
180         cmd <= "00110100";
181         data <= "0000000100000000";
182         wait until falling_edge(clk);
183         cmdrdy <= '0';
184
185         -- Set From Sensors
186         wait until falling_edge(clk);
187         cmdrdy <= '1';
188         cmd <= "00110101";
189         data <= "0000000100000000";
190         wait until falling_edge(clk);
191         cmdrdy <= '0';
192
193         -- Set  Direccion
194         wait until falling_edge(clk);
195         cmdrdy <= '1';
196         cmd <= "00110110";
197         data <= "0000001100000000";
198         wait until falling_edge(clk);
199         cmdrdy <= '0';
200
201         wait;
202
203      end process;
204
205   end Behavioral;
206
```

7.5. Archivo HBridgeCtrl_tb.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 10/22/2024 05:41:18 PM
6  -- Design Name:
7  -- Module Name: HBridgeCtrl_TB - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity HBridgeCtrl_TB is
35 --  Port ( );
36 end HBridgeCtrl_TB;
37
38 architecture Behavioral of HBridgeCtrl_TB is
39
40 component HBridgeCtrl is
41     Generic( POWER_SEL_WIDTH: NATURAL:=7;    -- Ancho en bits del selector de PWM
42              PWM_DIV: NATURAL:=100;          -- Resolucion del PWM
43              PWM_PERIOD: NATURAL:=10000);    -- Cantidad de pulsos de clock para cada unidad de PWM_DIV
44     Port(piHBCClk : in STD_LOGIC;
45          piHBCRst : in STD_LOGIC;
46          piHBCEna : in STD_LOGIC;
47          piHBCSet : in STD_LOGIC;
48          piHBCDirSel : in STD_LOGIC;
49          piHBCPowerSel: in STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
50          poHBCDir : out STD_LOGIC_VECTOR(1 downto 0);
51          poHBCPower : out STD_LOGIC;
52          poHBCDirSel : out STD_LOGIC;                                    -- Señal de salida conectada al valor
            ↪  del latch
53          poHBCPowerSel: out STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0)        -- Señal de salida conectada al valor
            ↪  del latch
54          );
55 end component HBridgeCtrl;
56
57 signal clk, rst, ena,set, dirsel, power, diro : std_logic;
58 signal powersel, powerselo: std_logic_vector(7-1 downto 0);
59 signal dir : std_logic_vector(1 downto 0);
60
61 begin
62     instHBridgeCtrl: HBridgeCtrl
63         Generic map( POWER_SEL_WIDTH => 7,
64                  PWM_DIV => 10,
65                  PWM_PERIOD => 2)
66         Port map ( piHBCClk => clk,
67                  piHBCRst => rst,
68                  piHBCEna => ena,
69                  piHBCSet => set,
70                  piHBCDirSel => dirsel,
71                  piHBCPowerSel => powersel,
72                  poHBCDir => dir,
73                  poHBCPower => power,
74                  poHBCDirSel => diro,
75                  poHBCPowerSel => powerselo
```

```
76                  );
77
78          pClk: process
79          begin
80                  clk <= '1';
81                  wait for 1 ns;
82                  clk <= '0';
83                  wait for 1 ns;
84          end process;
85
86      process
87      begin
88          rst <= '1';
89          set <= '0';
90          ena <= '0';
91          wait for 5 ns;
92          rst <= '0';
93          ena <= '1';
94          dirsel <= '1';
95          powersel <= "0000010";
96          wait until falling_edge(clk);
97          set <= '1';
98          wait until falling_edge(clk);
99          dirsel <= '0';
100         powersel <= "0000000";
101         set <= '0';
102         wait for 400 ns;
103
104         dirsel <= '0';
105         powersel <= "0000100";
106         wait until falling_edge(clk);
107         set <= '1';
108         wait until falling_edge(clk);
109         dirsel <= '0';
110         powersel <= "0000000";
111         set <= '0';
112         wait for 400 ns;
113
114         wait;
115     end process;
116
117
118 end Behavioral;
```

7.6. Archivo HexToSevSeg_tb.vhd

```
1   library ieee;
2
3   -- STD_LOGIC and STD_LOGIC_VECTOR types, and relevant functions
4   use ieee.std_logic_1164.all;
5
6   -- SIGNED and UNSIGNED types, and relevant functions
7   use ieee.numeric_std.all;
8
9   entity HexToSevSeg_TB is
10  end entity HexToSevSeg_TB;
11
12
13  architecture Behavioral of HexToSevSeg_TB is
14      component HexToSevSeg is
15      port (
16          piHTSSSEna: in std_logic;
17          piHTSSSData: in std_logic_vector(3 downto 0);
18          poHTSSSOutput: out std_logic_vector(6 downto 0)
19      );
20       end component HexToSevSeg;
21
22       signal D : std_logic_vector(3 downto 0);
23       signal H : std_logic_vector(6 downto 0);
24       signal ena : std_logic;
25
26  begin
27      instHexToSevSeg: HexToSevSeg
28          Port map ( piHTSSSEna => ena,
29                     piHTSSSData => D,
30                     poHTSSSOutput => H
31                     );
32
33
34      process
35      begin
36          ena <='1';
37                  D <= "0001";
38                  wait for 100 ns;
39          ena <='0';
40                  wait for 100 ns;
41          ena <='1';
42                  D <= "1000";
43                  wait for 100 ns;
44          ena <='0';
45                  wait for 100 ns;
46                  D <= "1111";
47          ena <='1';
48                  wait for 100 ns;
49          ena <='0';
50                  wait for 100 ns;
51      wait;
52      end process;
53
54  end architecture Behavioral;
55
56
```

7.7. Archivo IMain_tb.vhd

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date: 11/08/2024 04:04:23 PM
6   -- Design Name:
7   -- Module Name: IMain_TB - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  -- Uncomment the following library declaration if using
26  -- arithmetic functions with Signed or Unsigned values
27  use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx leaf cells in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity IMain_TB is
35  --  Port ( );
36  end IMain_TB;
37
38  architecture Behavioral of IMain_TB is
39      component IMain is
40       Port (  piIMClk : in STD_LOGIC;              --                                                  Port E3
41               piIMRst : in STD_LOGIC;              --                                                       SW1
42               piIMEna : in STD_LOGIC;              --                                                       SW0
43               piIMRx : in STD_LOGIC;               --                                                  Port A9
44               poIMTx : out STD_LOGIC;              --                                                  Port D10
45               piIMSensors : in STD_LOGIC_VECTOR(3 downto 0);  -- Sensores fisicos                          BTN0 - BTN3
46               poIMSevSeg : out STD_LOGIC_VECTOR(6 downto 0);  -- Al display de 7 segmentos                 IO32 - IO27
47               poIMDot : out STD_LOGIC;                        -- Al punto del display de 7 segmentos -     IO26
48               poIMPowerMD : out STD_LOGIC;                    -- Al pin Enable del L293D motor derecho -   IO41
49               poIMDirMD : out STD_LOGIC_VECTOR(1 downto 0);   -- A los pin dir del L293D -                 LED4 y LED5
50               poIMPowerMI : out STD_LOGIC;                    -- Al pin Enable del L293D motor izquierdo - IO40
51               poIMDirMI : out STD_LOGIC_VECTOR(1 downto 0);   -- A los pin dir del L293D -                 LED6 y LED7
52               poIMStat : out STD_LOGIC                        -- Led de estado - BLink de 200ms en CMD In  LED0_R (G6)
53       );
54      end component IMain;
55
56      signal clk, rst, ena, rx, tx, dot, poMD, poMI, stat: STD_LOGIC;
57      signal sensors: STD_LOGIC_VECTOR(3 downto 0);
58      signal sevseg: STD_LOGIC_VECTOR(6 downto 0);
59      signal dirMD, dirMI: STD_LOGIC_VECTOR(1 downto 0);
60
61
62      constant brt: time := 50 ns;
63      signal input: STD_LOGIC_VECTOR(8-1 downto 0);
64
65  begin
66
67          instIMain: IMain
68          Port map ( piIMClk => clk,
69               piIMRst => rst,
70               piIMEna => ena,
71               piIMRx => rx,
72               poIMTx => tx,
73               piIMSensors=> sensors,
74               poIMSevSeg => sevseg,
75               poIMDot => dot,
76               poIMPowerMD => poMD,
77               poIMDirMD => dirMD,
```

```
78                poIMPowerMI => poMI,
79                poIMDirMI => dirMI,
80                poIMStat => stat
81            );
82
83       pClk: process
84       begin
85           clk <= '1';
86           wait for 5 ns;
87           clk <= '0';
88           wait for 5 ns;
89       end process;
90
91
92       process
93          type Tcmd is array (0 to 4) of NATURAL;
94          variable cmd : Tcmd;
95       begin
96          rst <= '1';
97          ena <= '0';
98          wait for 33 ns;
99          ena <= '1';
100         rst <= '0';
101         rx <= '1';
102
103         -- Velocidad Motor Derecho 55%
104         cmd(0) := 68;
105         cmd(1) := 01;
106         cmd(2) := 05;
107         cmd(3) := 05;
108         cmd(4) := 90;
109
110         for n in 0 to 4 loop
111             input <= STD_LOGIC_VECTOR(TO_UNSIGNED(cmd(n), 8));
112             wait for brt;
113             rx <= '0';
114             for i in 0 to 7 loop
115                 wait for brt;
116                 rx <= input(i);
117             end loop;
118             wait for brt;
119             rx <= '1';
120          end loop;
121
122         -- Velocidad Motor izquierdo 20%
123         cmd(0) := 68;
124         cmd(1) := 02;
125         cmd(2) := 02;
126         cmd(3) := 00;
127         cmd(4) := 90;
128
129         for n in 0 to 4 loop
130             input <= STD_LOGIC_VECTOR(TO_UNSIGNED(cmd(n), 8));
131             wait for brt;
132             rx <= '0';
133             for i in 0 to 7 loop
134                 wait for brt;
135                 rx <= input(i);
136             end loop;
137             wait for brt;
138             rx <= '1';
139          end loop;
140
141
142          -- Automatico, velocidad media 40%
143         wait for 10 us;
144         sensors <= "0110";
145         cmd(0) := 68;
146         cmd(1) := 03;
147         cmd(2) := 04;
148         cmd(3) := 00;
149         cmd(4) := 90;
150
151         for n in 0 to 4 loop
152             input <= STD_LOGIC_VECTOR(TO_UNSIGNED(cmd(n), 8));
153             wait for brt;
154             rx <= '0';
155             for i in 0 to 7 loop
156                 wait for brt;
```

```
157                 rx <= input(i);
158             end loop;
159             wait for brt;
160             rx <= '1';
161          end loop;
162 
163 
164         -- Cambio en estado de los sensores
165         wait for 10 us;
166         sensors <= "1100";
167 
168         -- Cambio en estado de los sensores
169         wait for 10 us;
170         sensors <= "1000";
171 
172         -- Control desde PC
173         cmd(0) := 68;
174         cmd(1) := 05;
175         cmd(2) := 00;
176         cmd(3) := 00;
177         cmd(4) := 90;
178 
179         for n in 0 to 4 loop
180             input <= STD_LOGIC_VECTOR(TO_UNSIGNED(cmd(n), 8));
181             wait for brt;
182             rx <= '0';
183             for i in 0 to 7 loop
184                 wait for brt;
185                 rx <= input(i);
186             end loop;
187             wait for brt;
188             rx <= '1';
189          end loop;
190 
191 
192         -- Control manual de sensores a "0011"
193         cmd(0) := 68;
194         cmd(1) := 04;
195         cmd(2) := 03;
196         cmd(3) := 00;
197         cmd(4) := 90;
198 
199         for n in 0 to 4 loop
200             input <= STD_LOGIC_VECTOR(TO_UNSIGNED(cmd(n), 8));
201             wait for brt;
202             rx <= '0';
203             for i in 0 to 7 loop
204                 wait for brt;
205                 rx <= input(i);
206             end loop;
207             wait for brt;
208             rx <= '1';
209          end loop;
210 
211         wait for 10 us;
212 
213         -- Control manual de sensores a "1100"
214         cmd(0) := 68;
215         cmd(1) := 04;
216         cmd(2) := 12;
217         cmd(3) := 00;
218         cmd(4) := 90;
219 
220         for n in 0 to 4 loop
221             input <= STD_LOGIC_VECTOR(TO_UNSIGNED(cmd(n), 8));
222             wait for brt;
223             rx <= '0';
224             for i in 0 to 7 loop
225                 wait for brt;
226                 rx <= input(i);
227             end loop;
228             wait for brt;
229             rx <= '1';
230          end loop;
231 
232 
233          -- velocidad media 75%
234         wait for 10 us;
235         cmd(0) := 68;
```

```
236          cmd(1) := 03;
237          cmd(2) := 07;
238          cmd(3) := 05;
239          cmd(4) := 90;
240
241          for n in 0 to 4 loop
242              input <= STD_LOGIC_VECTOR(TO_UNSIGNED(cmd(n), 8));
243              wait for brt;
244              rx <= '0';
245              for i in 0 to 7 loop
246                  wait for brt;
247                  rx <= input(i);
248              end loop;
249              wait for brt;
250              rx <= '1';
251           end loop;
252
253          -- Control manual de sensores a "0110"
254          wait for 10 us;
255          cmd(0) := 68;
256          cmd(1) := 04;
257          cmd(2) := 06;
258          cmd(3) := 00;
259          cmd(4) := 90;
260
261          for n in 0 to 4 loop
262              input <= STD_LOGIC_VECTOR(TO_UNSIGNED(cmd(n), 8));
263              wait for brt;
264              rx <= '0';
265              for i in 0 to 7 loop
266                  wait for brt;
267                  rx <= input(i);
268              end loop;
269              wait for brt;
270              rx <= '1';
271           end loop;
272
273
274
275
276           wait;
277
278       end process;
279
280   end Behavioral;
```

7.8. Archivo ModuleCounter_tb.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 10/17/2024 05:36:16 PM
6  -- Design Name:
7  -- Module Name: ModuleCounter_tb - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24
25 entity ModuleCounter_TB is
26 end ModuleCounter_TB;
27
28
29 architecture A_ModuleCounter_TB of ModuleCounter_TB is
30
31 component ModuleCounter is
32     Generic (NBits: natural;
33                 Max: natural);
34     Port (  piMCClk : in STD_LOGIC;
35             piMCEna : in STD_LOGIC;
36             piMCRst : in STD_LOGIC;
37             poMCO : out std_logic
38                 );
39 end component ModuleCounter;
40
41 signal clk, ena, rst, x1, x2, x3, x12 : std_logic;
42
43 begin
44
45         instBb1: ModuleCounter
46                 generic map( NBits => 3, Max => 1)
47                 port map(piMCClk => clk, piMCEna => ena, piMCRst => rst, poMCO => x1);
48
49         instBb2: ModuleCounter
50                 generic map( NBits => 3, Max => 2)
51                 port map(piMCClk => clk, piMCEna => ena, piMCRst => rst, poMCO => x2);
52
53         instBb3: ModuleCounter
54                 generic map( NBits => 3, Max => 3)
55                 port map(piMCClk => clk, piMCEna => ena, piMCRst => rst, poMCO => x3);
56
57         instBb12: ModuleCounter
58                 generic map( NBits => 4, Max => 12)
59                 port map(piMCClk => clk, piMCEna => ena, piMCRst => rst, poMCO => x12);
60
61         pClk: process
62         begin
63                 clk <= '1';
64                 wait for 10 ns;
65                 clk <= '0';
66                 wait for 10 ns;
67         end process;
68
69
70         process
71         begin
72                 rst <= '1';
73                 ena <= '0';
74                 wait for 40 ns;
75                 rst <= '0';
76                 wait until falling_edge(clk);
77                 ena <= '1';
```

```
78                  wait for 1000 ns;
79                  wait until falling_edge(clk);
80                  ena <= '0';
81                  wait;
82          end process;
83
84
85  end A_ModuleCounter_TB;
```

7.9. Archivo PwmGen_tb.vhd

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date: 10/22/2024 05:41:18 PM
6   -- Design Name:
7   -- Module Name: PwmGen_tb - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  -- Uncomment the following library declaration if using
26  -- arithmetic functions with Signed or Unsigned values
27  --use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx leaf cells in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity PwmGen_tb is
35  --  Port ( );
36  end PwmGen_tb;
37
38  architecture Behavioral of PwmGen_tb is
39
40      component PwmGen is
41      Generic(PWM_WIDTH: NATURAL:=12;
42              ARR: NATURAL := 4096
43              );
44      Port(piPwmClk : in STD_LOGIC;
45           piPwmEna : in STD_LOGIC;
46           piPwmRst : in STD_LOGIC;
47           piPwmPower: in STD_LOGIC_VECTOR(PWM_WIDTH-1 downto 0);
48           poPwmPower : out STD_LOGIC
49           );
50      end component;
51      signal Clk, Ena, Rst, Power : std_logic;
52      signal PowerSel: std_logic_vector(7-1 downto 0);
53
54
55      begin
56          instPwmGen: PwmGen
57          generic map( PWM_WIDTH => 7, ARR => 10)
58          port map(piPwmClk => Clk,
59                  piPwmEna => Ena,
60                  piPwmRst => Rst,
61                  piPwmPower => PowerSel,
62                  poPwmPower => Power
63           );
64
65          pClk: process
66          begin
67              Clk <= '1';
68              wait for 1 ns;
69              Clk <= '0';
70              wait for 1 ns;
71          end process;
72
73      process
74      begin
75          Rst <= '1';
76          Ena <= '0';
77          wait for 3 ns;--
```

```
78
79          Rst <= '0';
80          ena <= '1';
81          wait for 4 ns;
82
83
84          PowerSel <= "0000001";
85          wait for 200 ns;
86
87
88          PowerSel <= "0000010";
89          wait for 200 ns;
90
91
92          PowerSel <= "0000100";
93          wait for 200 ns;
94
95
96          PowerSel <= "0001000";
97          wait for 200 ns;
98
99
100         Rst <= '1';
101         wait for 3 ns;
102         Rst <= '0';
103         wait;
104       end process;
105
106
107   end Behavioral;
108
```

7.10. Archivo ToDisplay_tb.vhd

```
1   library ieee;
2
3   -- STD_LOGIC and STD_LOGIC_VECTOR types, and relevant functions
4   use ieee.std_logic_1164.all;
5
6   -- SIGNED and UNSIGNED types, and relevant functions
7   use ieee.numeric_std.all;
8
9   entity ToDisplay_TB is
10  end entity ToDisplay_TB;
11
12
13  architecture Behavioral of ToDisplay_TB is
14      component ToDisplay is
15      Generic( POWER_SEL_WIDTH: NATURAL:=7;
16               Max: NATURAL := 100000000);    -- Ancho en bits del selector de PWM
17      Port(piTDClk: in STD_LOGIC;
18           piTDRst: in STD_LOGIC;
19           piTDEna: in STD_LOGIC;
20           piTDPowerMD: in STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
21           piTDPowerMI: in STD_LOGIC_VECTOR (POWER_SEL_WIDTH-1 downto 0);
22           piTDMode: in STD_LOGIC;
23           poTDData: out STD_LOGIC_VECTOR (3 downto 0);
24           poTDDot: out STD_LOGIC
25           );
26       end component ToDisplay;
27
28       signal data : std_logic_vector(3 downto 0);
29       signal poMD, poMI : std_logic_vector(6 downto 0);
30       signal clk, rst, ena, mode, dot : std_logic;
31
32  begin
33      instToDisplay: ToDisplay
34          Generic map( POWER_SEL_WIDTH => 7,
35                       Max => 2)
36          Port map ( piTDClk => clk,
37                                  piTDRst => rst,
38                                  piTDEna => ena,
39                                  piTDPowerMD => poMD,
40                                  piTDPowerMI => poMI,
41                                  piTDMode => mode,
42                                  poTDData => data,
43                                  poTDDot => dot
44                     );
45
46
47          pClk: process
48          begin
49                  clk <= '1';
50                  wait for 5 ns;
51                  clk <= '0';
52                  wait for 5 ns;
53          end process;
54
55      process
56      begin
57          ena <='1';
58          rst <= '0';
59          poMD <= STD_LOGIC_VECTOR(TO_UNSIGNED(40, 7));
60          poMI <= STD_LOGIC_VECTOR(TO_UNSIGNED(88, 7));
61          mode <= '0';
62          wait for 300 ns;
63          poMD <= STD_LOGIC_VECTOR(TO_UNSIGNED(55, 7));
64          poMI <= STD_LOGIC_VECTOR(TO_UNSIGNED(0, 7));
65          mode <= '1';
66          wait for 300 ns;
67          poMD <= STD_LOGIC_VECTOR(TO_UNSIGNED(90, 7));
68          poMI <= STD_LOGIC_VECTOR(TO_UNSIGNED(12, 7));
69          mode <= '0';
70      wait;
71      end process;
72
73  end architecture Behavioral;
74
75
76
```

7.11. Archivo TTrigger_tb.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 10/17/2024 05:36:16 PM
6  -- Design Name:
7  -- Module Name: TTrigger_tb - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21
22 use ieee.std_logic_1164.all;
23 use ieee.numeric_std.all;
24
25 entity TTrigger_TB is
26 end TTrigger_TB;
27
28
29 architecture A_TTrigger_TB of TTrigger_TB is
30
31 component TTrigger is
32     Generic (NBits: natural;
33                 Max: natural);
34     Port (  piTTClk : in STD_LOGIC;
35             piTTEna : in STD_LOGIC;
36             piTTRst : in STD_LOGIC;
37             poTTO : out std_logic
38                 );
39 end component TTrigger;
40
41 signal clk, ena, rst, x1, x2, x3, x12 : std_logic;
42
43 begin
44
45         instBb1: TTrigger
46                 generic map( NBits => 3, Max => 1)
47                 port map(piTTClk => clk, piTTEna => ena, piTTRst => rst, poTTO => x1);
48
49         instBb2: TTrigger
50                 generic map( NBits => 3, Max => 2)
51                 port map(piTTClk => clk, piTTEna => ena, piTTRst => rst, poTTO => x2);
52
53         instBb3: TTrigger
54                 generic map( NBits => 3, Max => 3)
55                 port map(piTTClk => clk, piTTEna => ena, piTTRst => rst, poTTO => x3);
56
57         instBb12: TTrigger
58                 generic map( NBits => 4, Max => 12)
59                 port map(piTTClk => clk, piTTEna => ena, piTTRst => rst, poTTO => x12);
60
61         pClk: process
62         begin
63                 clk <= '1';
64                 wait for 10 ns;
65                 clk <= '0';
66                 wait for 10 ns;
67         end process;
68
69
70         process
71         begin
72                 rst <= '1';
73                 ena <= '0';
74                 wait for 40 ns;
75                 rst <= '0';
76                 wait until falling_edge(clk);
77                 ena <= '1';
```

```
78                  wait for 500 ns;
79                  wait until falling_edge(clk);
80
81                  ena <= '0';
82                  rst <= '1';
83                  wait until falling_edge(clk);
84                  rst <= '0';
85                  wait for 150 ns;
86                  ena <= '1';
87
88                  wait;
89          end process;
90
91
92  end A_TTrigger_TB;
```

7.12. Archivo Uart_tb.vhd

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 11/08/2024 04:04:23 PM
-- Design Name:
-- Module Name: Uart_tb - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Uart_tb is
--  Port ( );
end Uart_tb;

architecture Behavioral of Uart_tb is

   component Uart is
   Generic ( DIV: NATURAL:= 10417); -- 100Mhz/DIV -> 9600 baud
   Port (  piUartClk : in STD_LOGIC; -- Clock de entrada
           piUartRst : in STD_LOGIC; -- Reset

           piUartRxEna : in STD_LOGIC; -- RX Enable
           piUartTxEna : in STD_LOGIC; -- TX Enable

           piUartRx : in STD_LOGIC;    -- Puerto RX
           poUartRxC: out STD_LOGIC; -- Receive complete - Hay datos para leer en el buffer poUartRxData
           poUartRxData : out STD_LOGIC_VECTOR (8-1 downto 0);

           piUartTxDataRdy : in STD_LOGIC;   -- Transmit ready - Los datos en piUartTxData estan listos para ser enviados
           piUartTxData : in STD_LOGIC_VECTOR (8-1 downto 0);
           poUartTx : out STD_LOGIC;    -- Puerto TX
           poUartTxC : out STD_LOGIC   -- Transmit Complete - Los datos en piUartTxData fueron enviados
   );
   end component;


   signal clk, rst, rxena, txena, rx, rxc, tx, txc, txrdy: STD_LOGIC;
   signal input, data: STD_LOGIC_VECTOR(8-1 downto 0);
   constant brt: time := 145 ns;
begin

        instUart: Uart
        generic map(DIV => 15)
        Port map ( piUartClk => clk,
             piUartRst => rst,
             piUartRxEna => rxena,
             piUartTxEna => txena,
             piUartRx => rx,
             poUartRxC => rxc,
             poUartRxData => data,
             piUartTxDataRdy =>txrdy,
             piUartTxData => data,
             poUartTx => tx,
             poUartTxC => txc);
```

```
78
79
80      pClk: process
81           begin
82                   clk <= '1';
83                   wait for 5 ns;
84                   clk <= '0';
85                   wait for 5 ns;
86           end process;
87
88
89       process
90         variable value : NATURAL;
91       begin
92         rx <= '1';
93         rst <= '0';
94         rxena <= '1';
95         txena <= '1';
96
97         value := 1;
98         input <= STD_LOGIC_VECTOR(TO_UNSIGNED(value, 8));
99         for n in 0 to 4 loop
100            wait for brt;
101            rx <= '0';
102            for i in 0 to 7 loop
103                wait for brt;
104                rx <= input(i);
105            end loop;
106            wait for brt;
107            rx <= '1';
108            value := value * 2 + 1;
109            input <= STD_LOGIC_VECTOR(TO_UNSIGNED(value, 8));
110        end loop;
111
112  --       rx <= '0';
113  --       input <= "10101010";
114  --       for i in 0 to 7 loop
115  --           wait for brt;
116  --           rx <= input(i);
117  --       end loop;
118  --       wait for brt;
119  --       rx <= '1';
120  --
121  --       wait for brtl;
122  --       rx <= '0';
123  --       input <= "11001100";
124  --       for i in 0 to 7 loop
125  --           wait for brt;
126  --           rx <= input(i);
127  --       end loop;
128  --       wait for brt;
129  --       rx <= '1';
130  --
131  --       wait for brtl;
132  --       rx <= '0';
133  --       input <= "00110011";
134  --       for i in 0 to 7 loop
135  --           wait for brt;
136  --           rx <= input(i);
137  --       end loop;
138  --       wait for brt;
139  --       rx <= '1';
140  --
141  --       wait for brtl;
142  --       rx <= '0';
143  --       input <= "00001111";
144  --       for i in 0 to 7 loop
145  --           wait for brt;
146  --           rx <= input(i);
147  --       end loop;
148  --       wait for brt;
149  --       rx <= '1';
150
151         wait;
152
153      end process;
154
155      txrdy <= rxc;
156
```

```
157    end Behavioral;
```

7.13. Archivo UaRx_tb.vhd

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 11/08/2024 04:04:23 PM
6  -- Design Name:
7  -- Module Name: UaRx_TB - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool Versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity UaRx_TB is
35 --  Port ( );
36 end UaRx_TB;
37
38 architecture Behavioral of UaRx_TB is
39    component UaRx is
40     Generic ( RxDIV: NATURAL:= 10417); -- 100Mhz/DIV -> 9600 baud
41     Port (  piUaRxClk : in STD_LOGIC; -- Clock de entrada
42             piUaRxRst : in STD_LOGIC; -- Reset
43             piUaRxEna : in STD_LOGIC; -- RX Enable
44             piUaRxRx : in STD_LOGIC;     -- Puerto RX
45             poUaRxC: out STD_LOGIC; -- Receive complete - Hay datos para leer en el buffer poUaRxData
46             poUaRxData : out STD_LOGIC_VECTOR (8-1 downto 0)
47     );
48    end component UaRx;
49
50    signal clk, rst, ena, rx, rxc: STD_LOGIC;
51    signal data: STD_LOGIC_VECTOR(8-1 downto 0);
52    signal test: STD_LOGIC_VECTOR(7 downto 0);
53 begin
54
55        instUaRx: UaRx
56        generic map(RxDIV => 100000000/9600)
57        Port map ( piUaRxClk => clk,
58             piUaRxRst => rst,
59             piUaRxEna => ena,
60             piUaRxRx => rx,
61             poUaRxC => rxc,
62             poUaRxData => data);
63
64    pClk: process
65         begin
66                 clk <= '1';
67                 wait for 5 ns;
68                 clk <= '0';
69                 wait for 5 ns;
70         end process;
71
72
73     process
74     begin
75        rst <= '1';
76        ena <= '0';
77        wait for 200 ns;
```

```
78
79          rx <= '1';
80          rst <= '0';
81          ena <= '1';
82          test <= std_logic_vector(TO_UNSIGNED(85, 8));
83          wait for 345 us;
84
85          rx <= '0';
86          wait for 104 us;
87          for i in 0 to 7 loop
88              rx <= test(i);
89              wait for 104 us;
90          end loop;
91          rx <= '1';
92
93          test <= std_logic_vector(TO_UNSIGNED(195, 8));
94          wait for 104 us;
95          rx <= '0';
96          wait for 104 us;
97          for i in 0 to 7 loop
98              rx <= test(i);
99              wait for 104 us;
100         end loop;
101         rx <= '1';
102         wait for 300 us;
103         wait;
104
105     end process;
106
107 end Behavioral;
108
```

7.14. Archivo UaTx_tb.vhd

```
1   ----------------------------------------------------------------------------------
2   -- Company:
3   -- Engineer:
4   --
5   -- Create Date: 11/08/2024 04:04:23 PM
6   -- Design Name:
7   -- Module Name: UaTx_TB - Behavioral
8   -- Project Name:
9   -- Target Devices:
10  -- Tool Versions:
11  -- Description:
12  --
13  -- Dependencies:
14  --
15  -- Revision:
16  -- Revision 0.01 - File Created
17  -- Additional Comments:
18  --
19  ----------------------------------------------------------------------------------
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  -- Uncomment the following library declaration if using
26  -- arithmetic functions with Signed or Unsigned values
27  --use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx leaf cells in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity UaTx_TB is
35  --  Port ( );
36  end UaTx_TB;
37
38  architecture Behavioral of UaTx_TB is
39     component UaTx is
40      Generic ( TxDIV: NATURAL:= 10417); -- 100Mhz/DIV -> 9600 baud
41      Port (  piUaTxClk : in STD_LOGIC; -- Clock de entrada
42              piUaTxRst : in STD_LOGIC; -- Reset
43              piUaTxEna : in STD_LOGIC; -- TX Enable
44
45              poUaTxTx : out STD_LOGIC;     -- Puerto TX
46              piUaTxDataRdy : in STD_LOGIC;    -- Transmit ready - Los datos en piUaTxData estan listos para ser enviados
47              poUaTxC : out STD_LOGIC;    -- Transmit Complete - Los datos en piUaTxData fueron enviados
48              piUaTxData : in STD_LOGIC_VECTOR (8-1 downto 0)
49      );
50     end component UaTx;
51
52     signal clk, rst, ena, txdrdy, tx, txc: STD_LOGIC;
53     signal data: STD_LOGIC_VECTOR(8-1 downto 0);
54  begin
55
56          instUaTx: UaTx
57          generic map(txDIV => 4)
58          Port map ( piUaTxClk => clk,
59               piUaTxRst => rst,
60               piUaTxEna => ena,
61               poUaTxTx => tx,
62               poUaTxC => txc,
63               piUaTxDataRdy => txdrdy,
64               piUaTxData => data);
65
66     pClk: process
67          begin
68                  clk <= '1';
69                  wait for 1 ns;
70                  clk <= '0';
71                  wait for 1 ns;
72          end process;
73
74
75      process
76      begin
77         rst <= '1';
```

```
78          ena <= '0';
79          wait for 13 ns;
80          rst <= '0';
81          ena <= '1';
82
83          wait for 13 ns;
84          data <= "00110011";
85          txdrdy <= '1';
86          wait for 13 ns;
87          txdrdy <= '0';
88          wait until rising_edge(txc);
89          data <= "10101010";
90          txdrdy <= '1';
91          wait for 13 ns;
92          txdrdy <= '0';
93
94          wait;
95
96       end process;
97
98   end Behavioral;
```