INFORME DE PROYECTO
CONTROL DE VELOCIDAD DE AUTO

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

Introducción a VHDL
Ingeniería en Mecatrónica

2 de diciembre de 2024

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

Índice

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.

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)**.

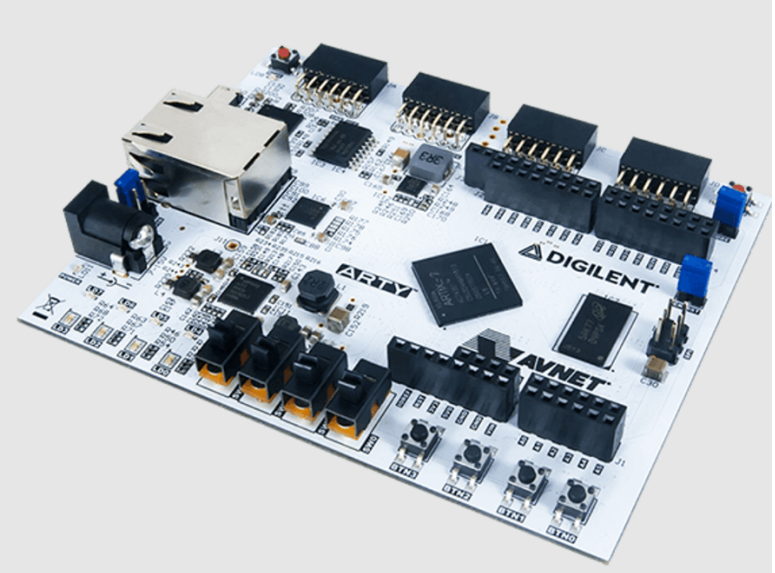


Figura 1: DIGILENT Artix-7 FPGA modelo Arty A7-100T

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 en diferentes módulos específicos para cada tarea, que serán los siguientes:

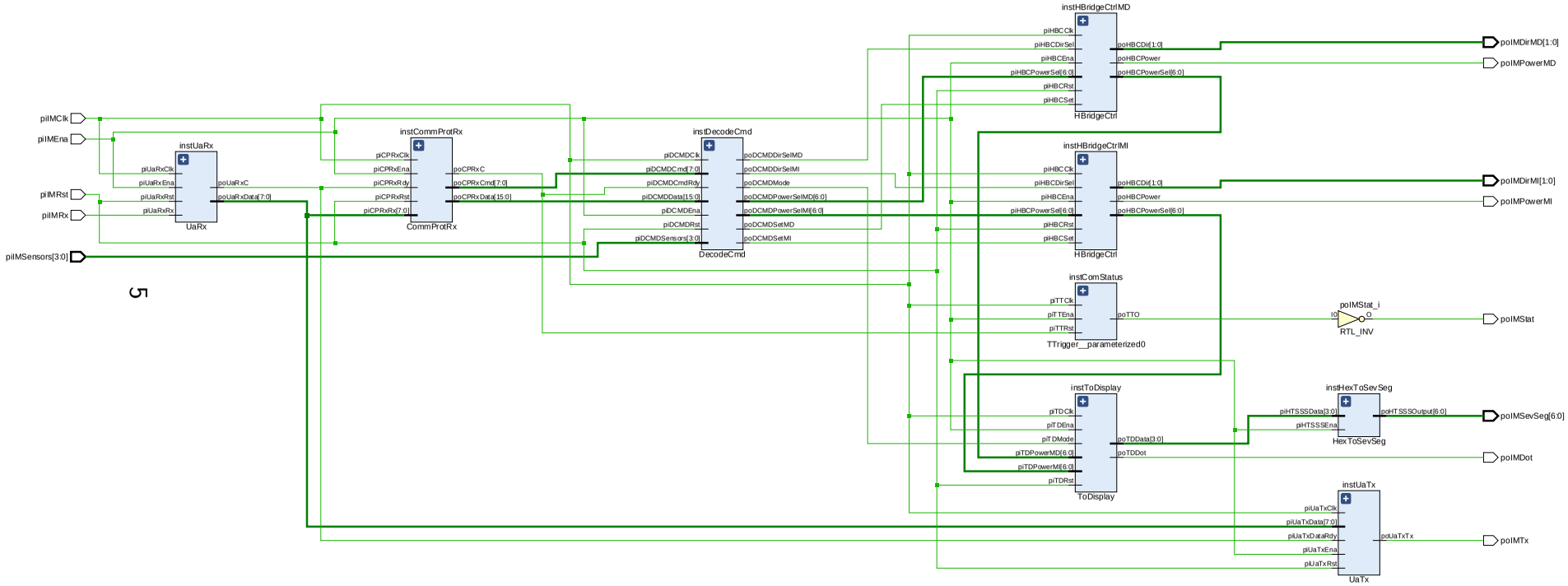
- UaRx: modulo Uart RX para la recepción desde la PC
- UaTx: modulo Uart TX, cuya única finalidad es hacer echo de la información recibida por UaRx.
- CommProtRx: modulo que valida un paquete del protocolo de comunicación, e incorpora una detección de timeout.
- DecodeCmd: Interpreta los comandos recibidos.
- 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.

2.1. 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

2.2. 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

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 al 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.

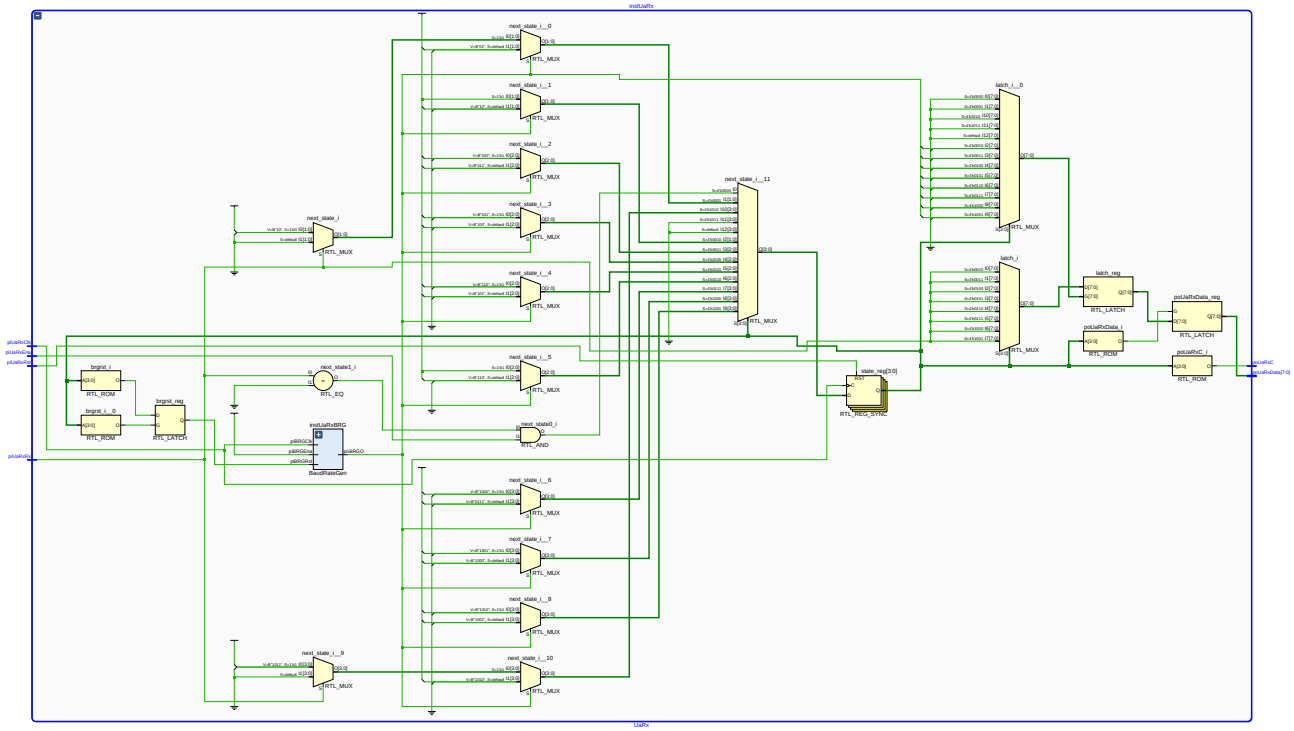


Figura 2: Esquemático del módulo **UaRx**

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 al UART envía a la PC paquetes de datos con la misma especificación que el modulo UaRx. El envio de un nuevo paquete de datos se inicia mediante un pulso en el puerto piUaTxDataReady, cuando se finaliza el envio el modulo notifica con un pulso de un clock de duración-

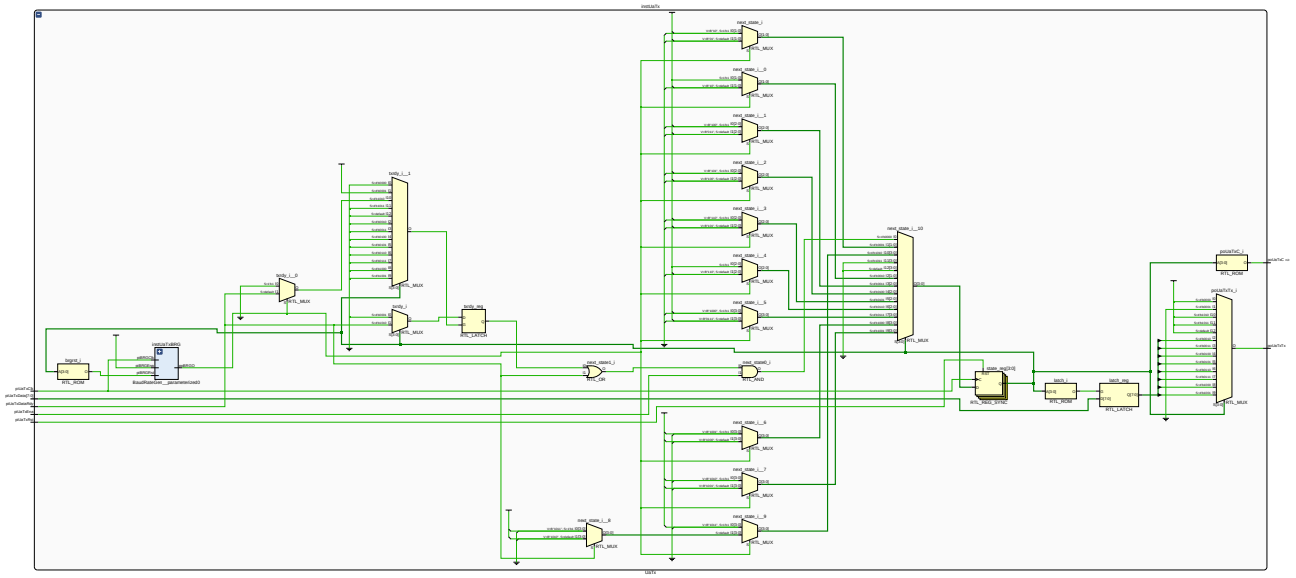


Figura 3: Esquemático del módulo **UaTx**

El diseño contempla iniciar el envío de un nuevo paquete de datos durante el envio 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 los dos bytes de datos en el bus de salida.

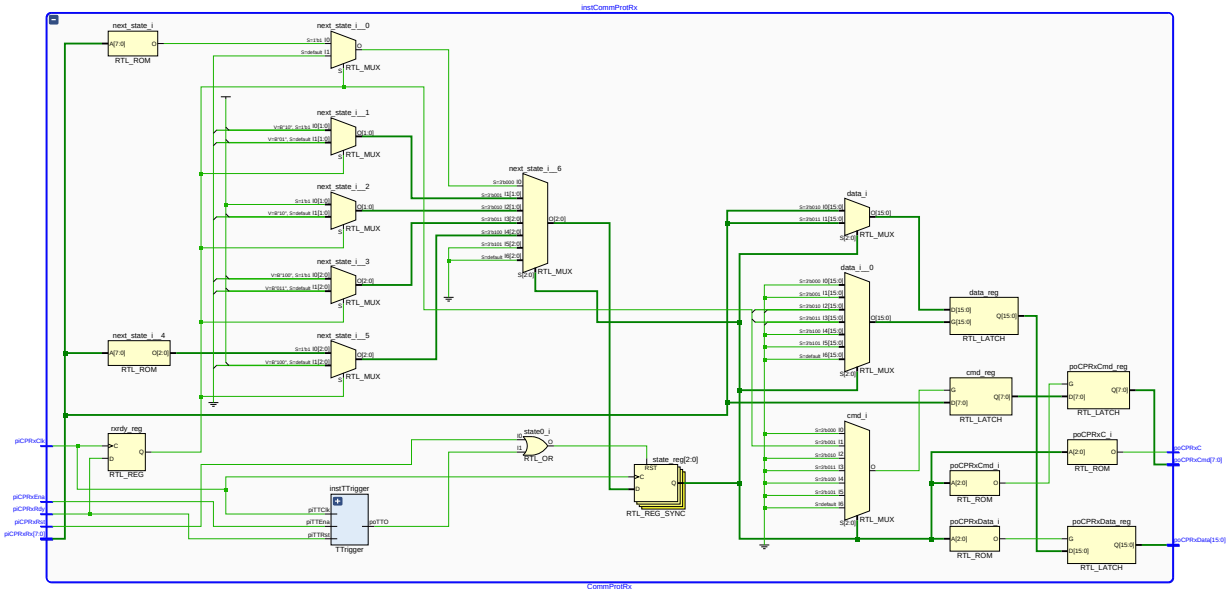


Figura 4: Esquemático del módulo **CommProtRx**

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 la velocidad de los motores cada 10ms segun el valor de los sensores, para cuando estos operan en modo automático. Este control periódico no es necesario por tratarse de un control proporcional pero si lo seria si se utilizaría si se incorpora un control PD o PID

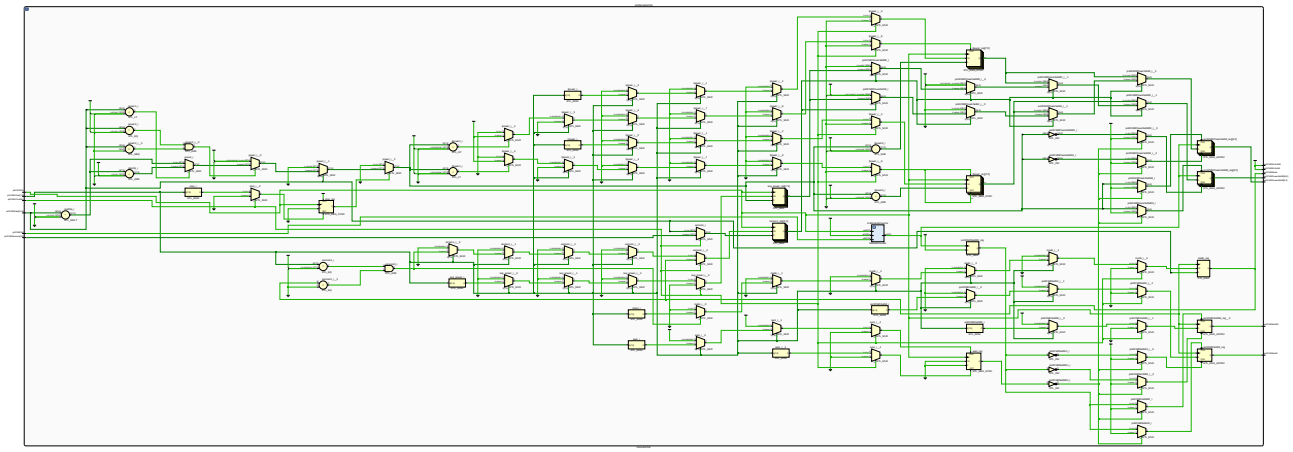


Figura 5: Esquemático del módulo **DecodeCmd**

3.5. HBridgeCtrl

AL recibir un pulso en el puerto set, el modulo modulo 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.

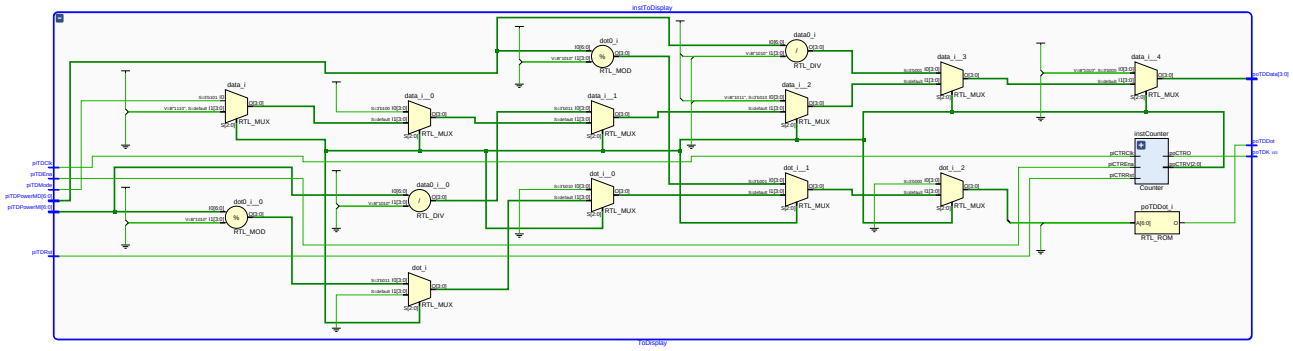


Figura 6: Esquemático del módulo **HBridgeCtrl**

Adicionalmente, este modulo tiene como salida los valores actuales de duty cycle y direccion, 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 1 segundo 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 es un valor hexadecimal de 4 bits mas una salida para el punto.

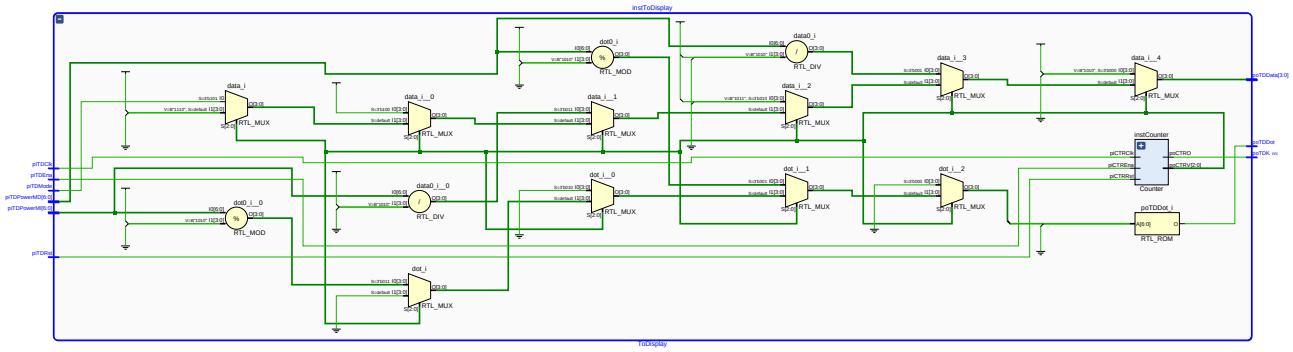


Figura 7: Esquemático del módulo **ToDisplay**

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.

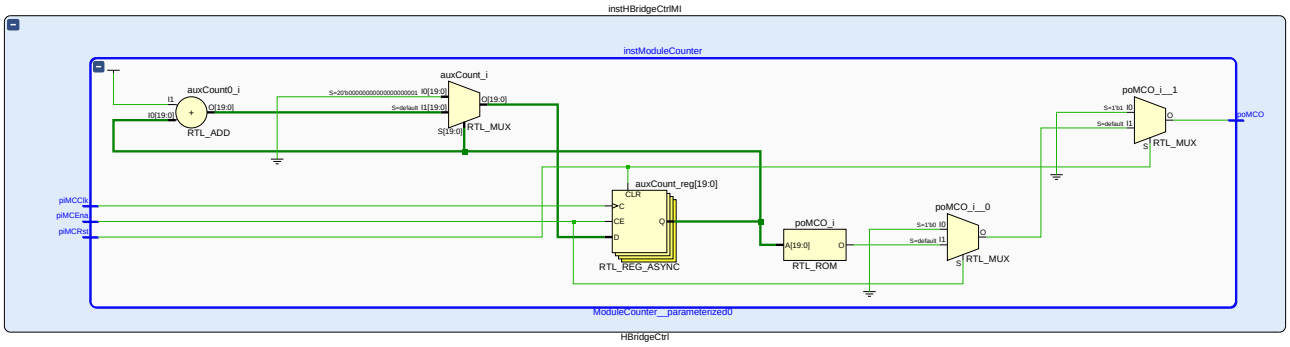


Figura 8: Esquemático del módulo **ModuleCounter**

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.

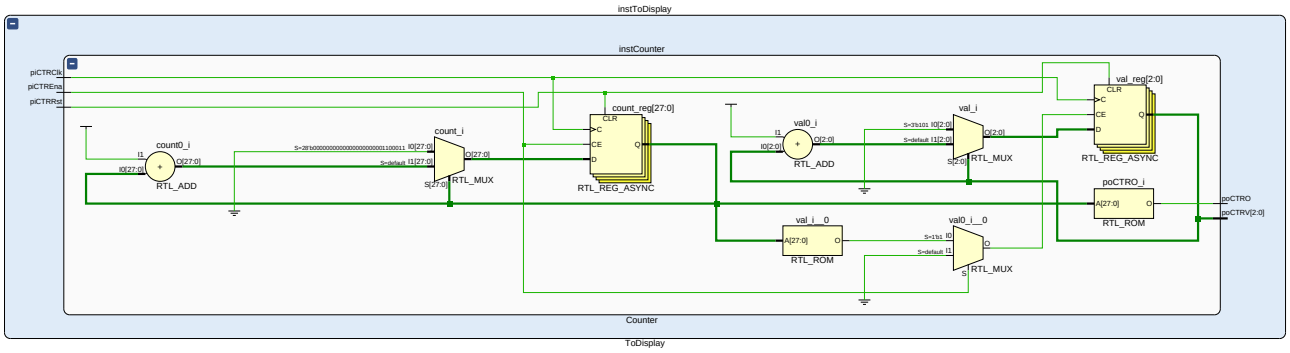


Figura 9: Esquemático del módulo **Counter**

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.

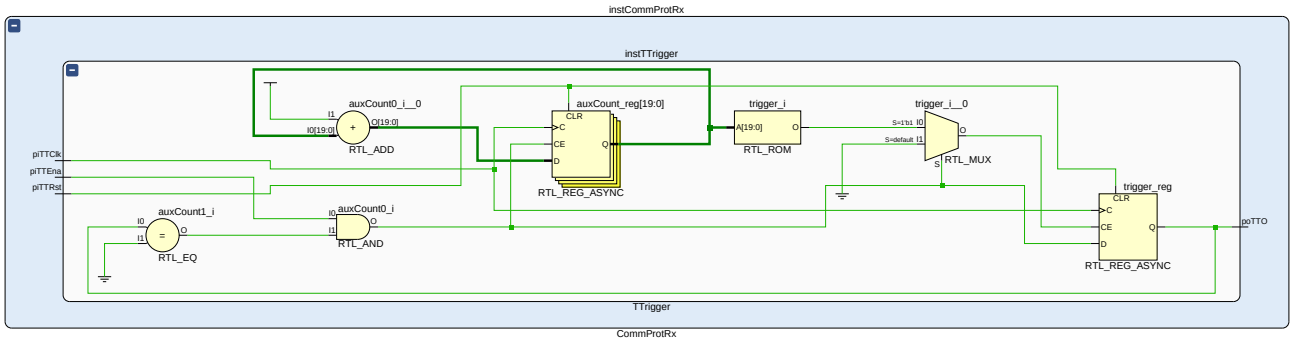


Figura 10: Esquemático del módulo **BaudRateGen**

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.

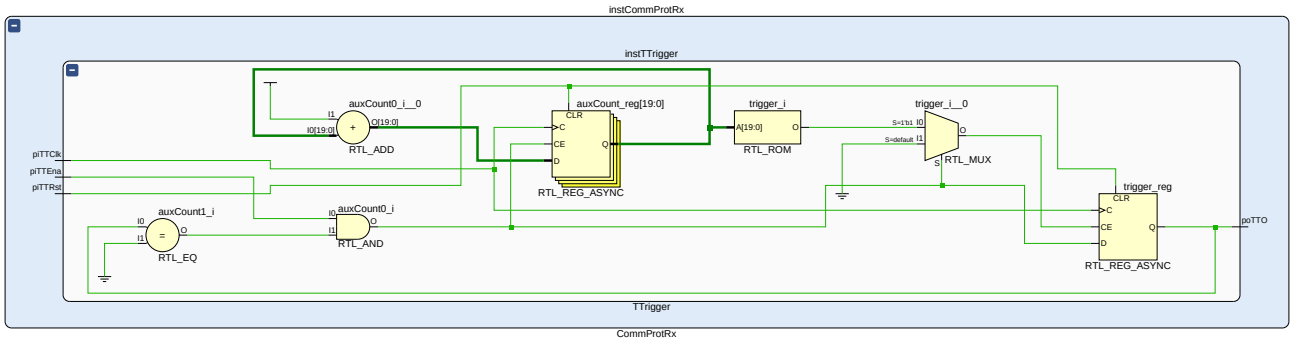


Figura 11: Esquemático del módulo **TTrigger**

Es utilizado por el modulo CommProtRx para detectar timeout.

3.7.5. PwmGen

Genera una salida PWM de periodo y resolución variable.

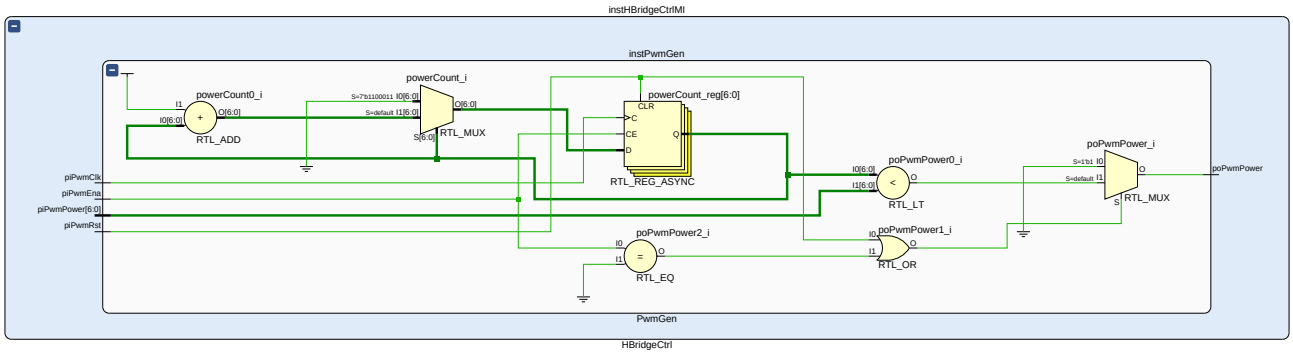


Figura 12: Esquemático del módulo **PwmGen**

3.7.6. HexToSevSeg

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

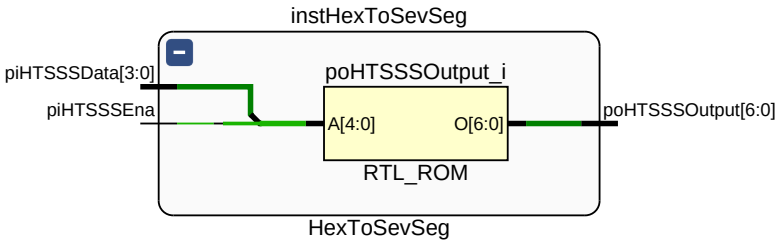


Figura 13: Esquemático del módulo **HexToSevSeg**

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.

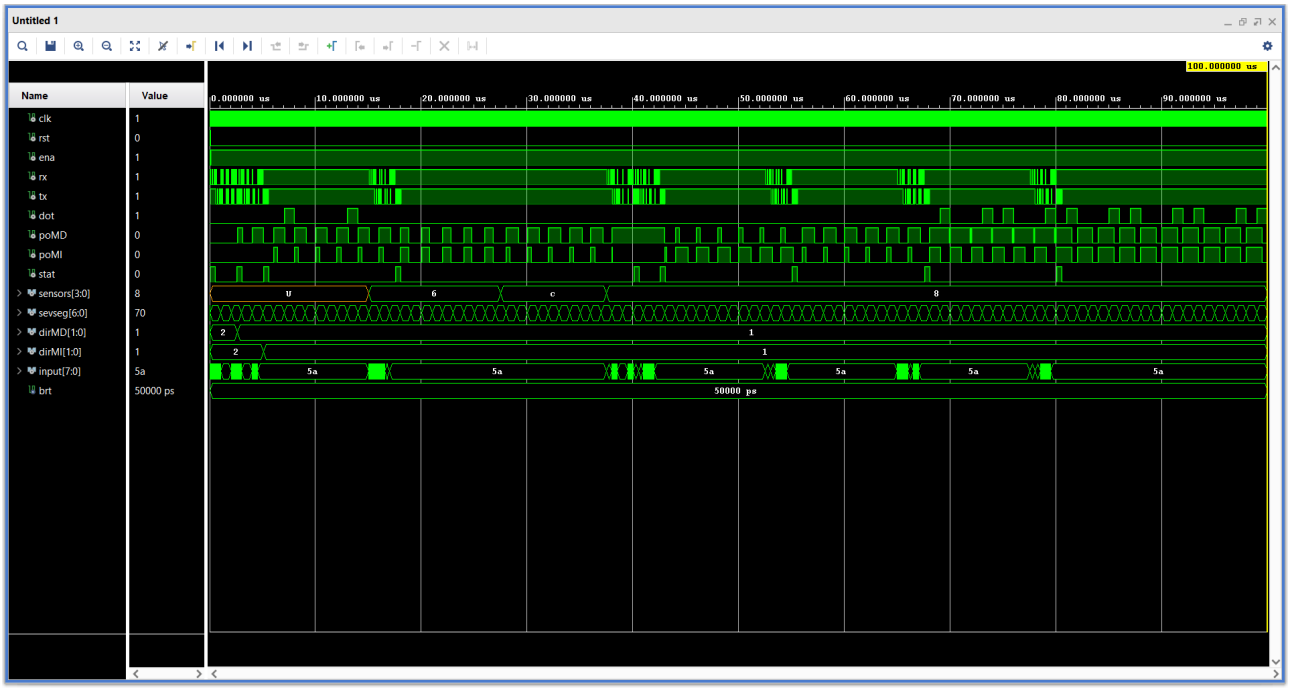


Figura 14: Simulación del sistema

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

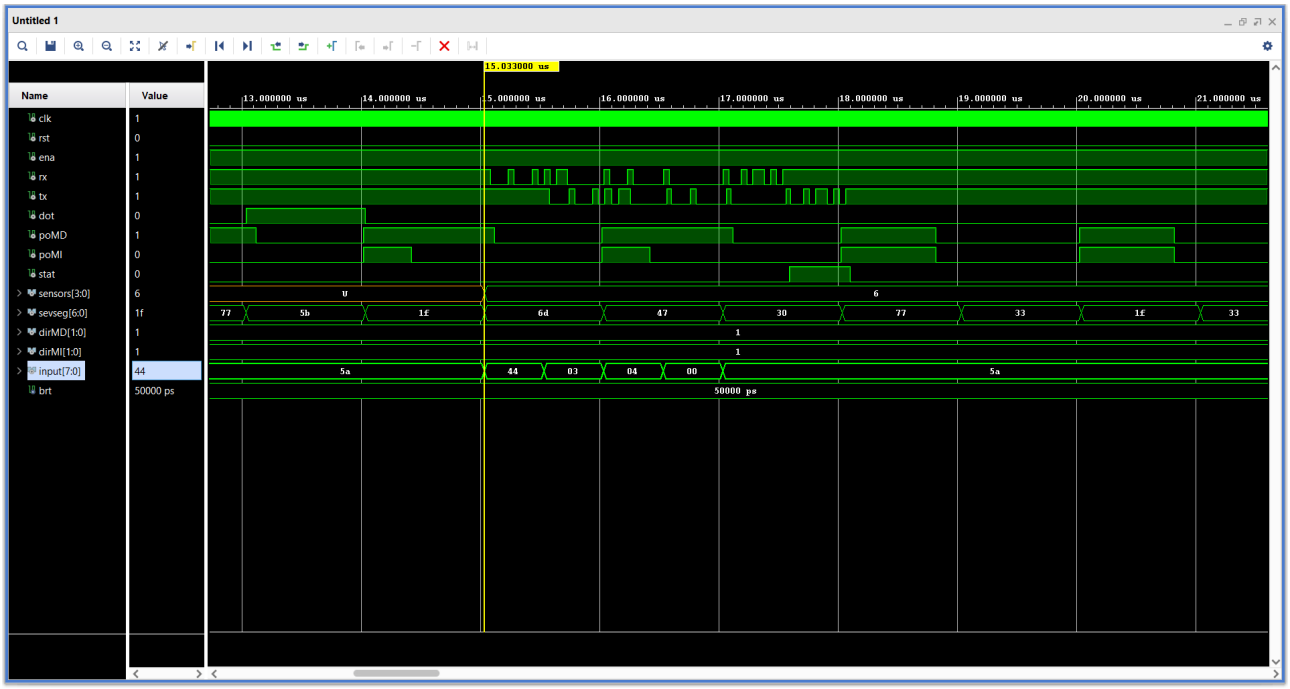


Figura 15: Simulación del sistema - Detalle de la recepción de un comando

5. Conclusiones