UNIVERSIDAD NACIONAL DE CÓRDOBA
Facultad de Ciencias Exactas, Físicas y Naturales

Cátedra de Arquitectura de Computadoras

Trabajo Práctico Nro. 2: UART

Fernández Oria, Luciano
luchoof212@gmail.com

Sieber, Braian
braiansieber@gmail.com

10/2/2019

CONSIGNA:

En este trabajo práctico se nos solicitó desarrollar un sistema de comunicaion UART que reciba instrucciones para la ejecucion de operaciones en la ALU que desarrollamos en el TP1.

Requerimientos:

- Implementar en FPGA una comunicacion UART.

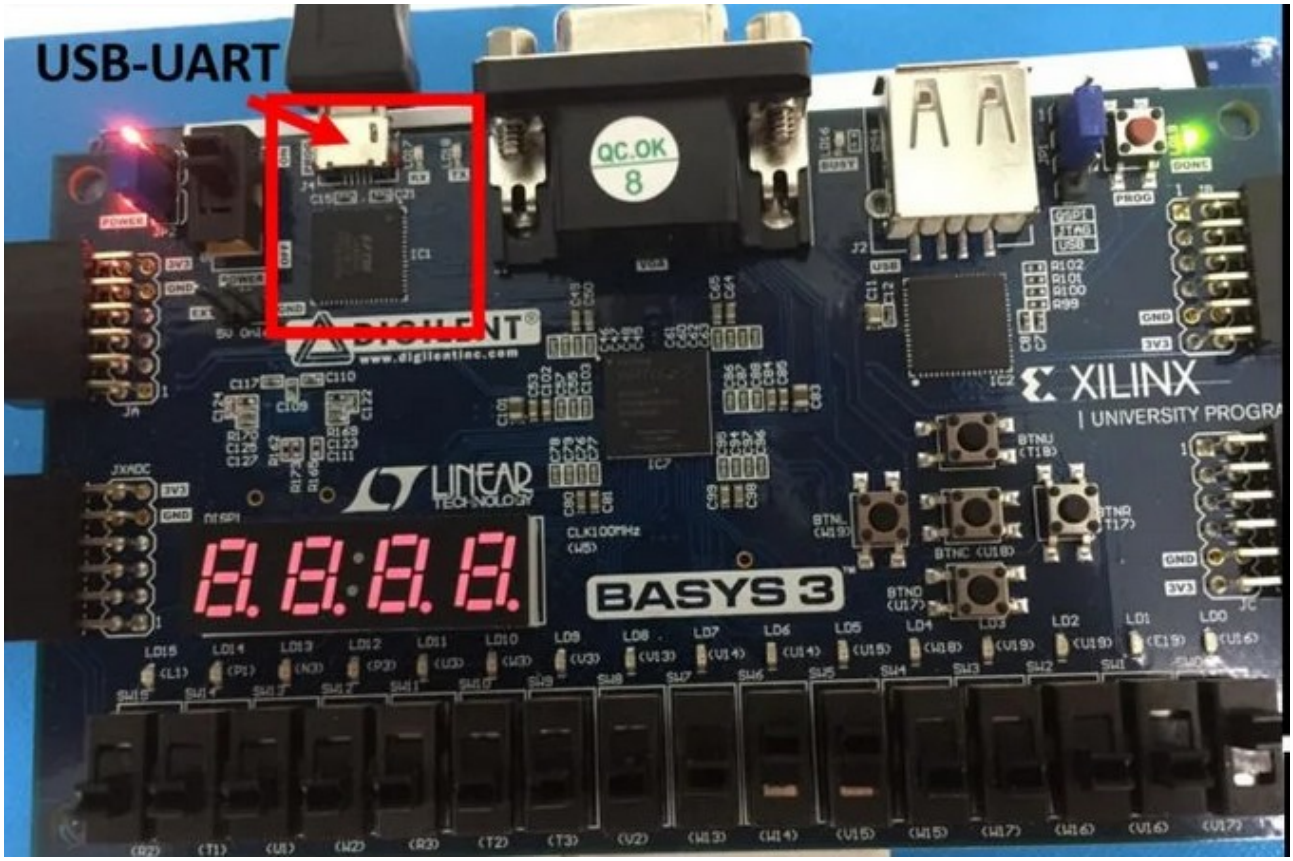
-Utilizar las placas de desarrollo Basys 3.

- Desarrollar una interfaz que comunique el transmisor y el receptor con la ALU.

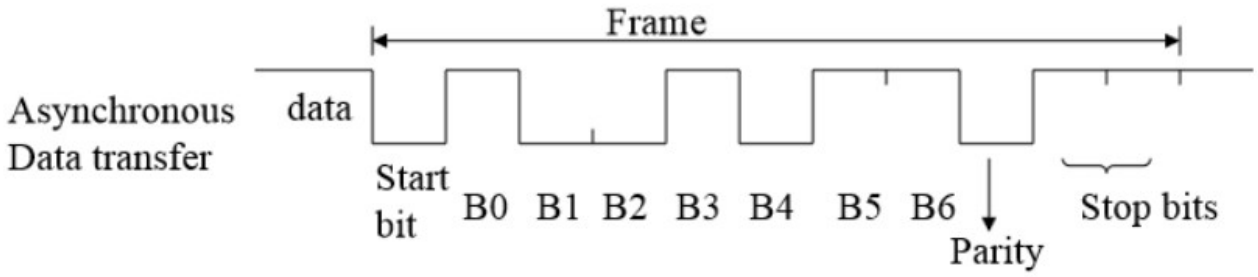
- Recibir y transmitir datos desde una PC.

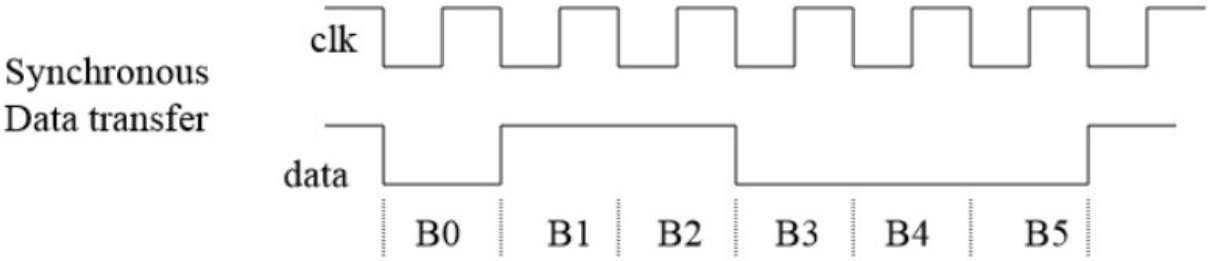
- Validar el desarrollo por medio de Test Bench

Para probar el modulo utilizamos la placa de desarrollo Basys 3 que cuenta con una FPGA Artix 7 de la empresa Xilinx y un conector USB-UART.



La transmision de datos se realiza mediante una comunicion asincrona entre los modulos de recepcion y transmision de datos y la PC; por lo que usamos un bit de start y otro de stop.





Los datos transmitidos son en bloques de 8 bits. NO utilizamos bit de paridad y cada uno representa un valor en ASCII:

Caracteres de control ASCII

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

Caracteres ASCII imprimibles

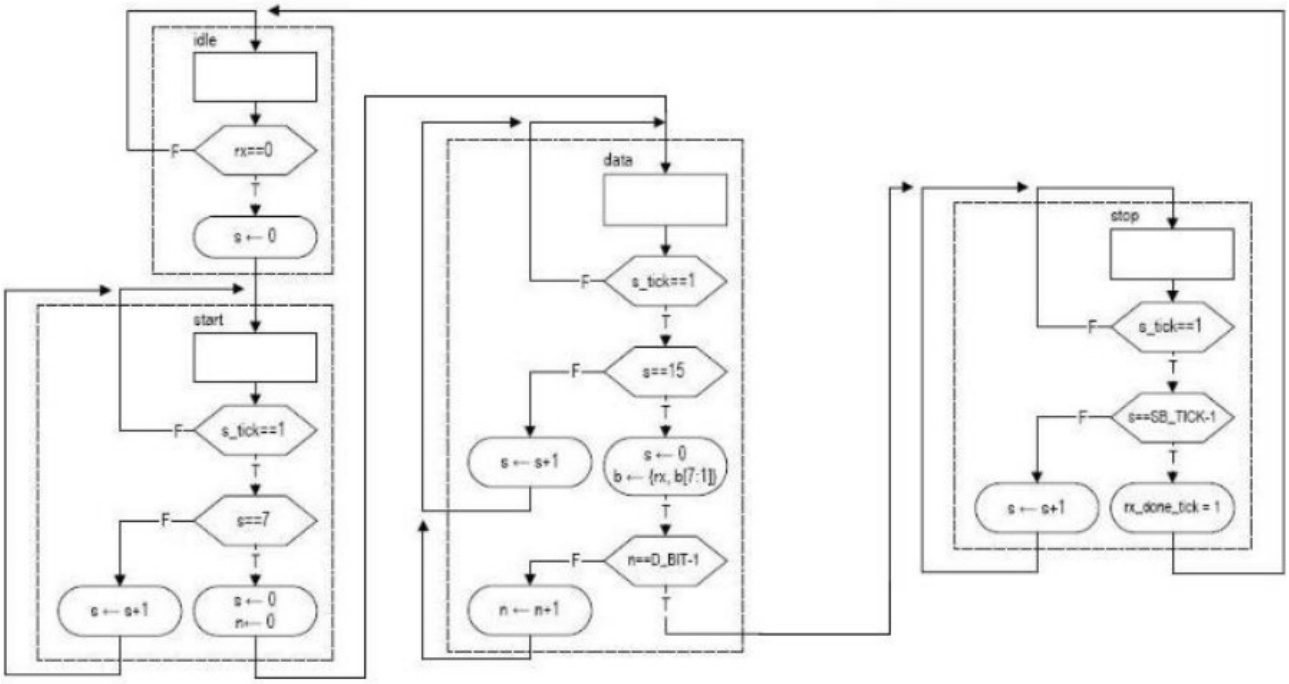
DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_	elCodigoASCII.com.ar		

ASCII extendido

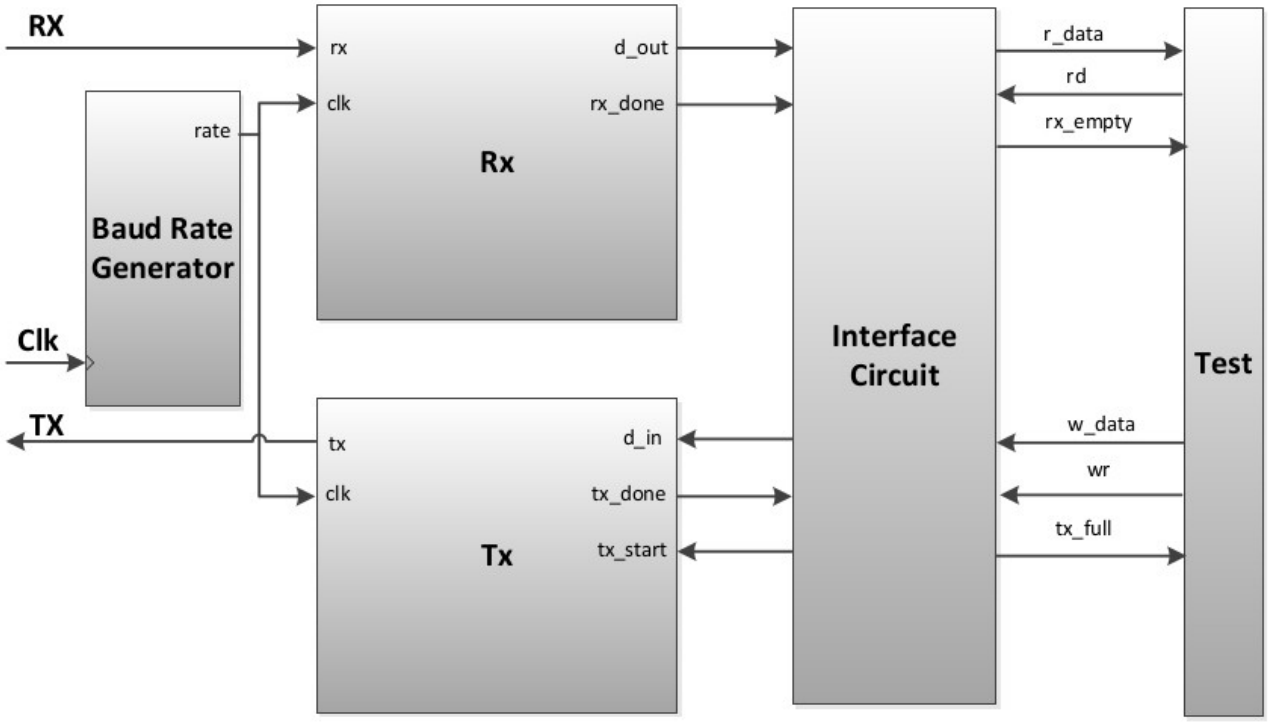
DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ö	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

La recepcion de datos comienza cuando se envia una señal de start al modulo de recepcion y termina con un bit de finalizacion.

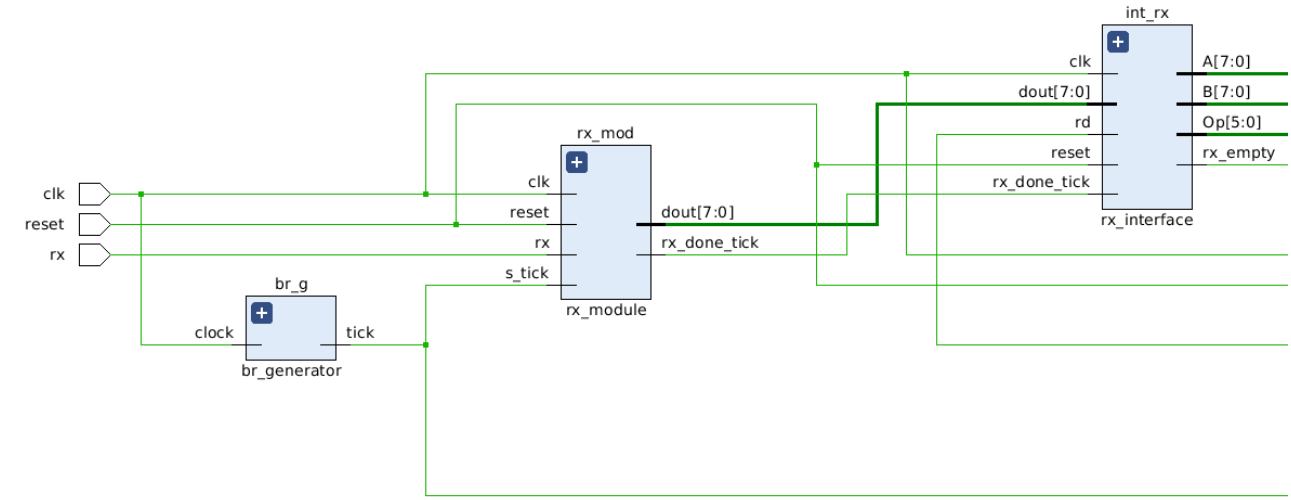
Diagrama de estados del modulo de recepcion extraido del libro:

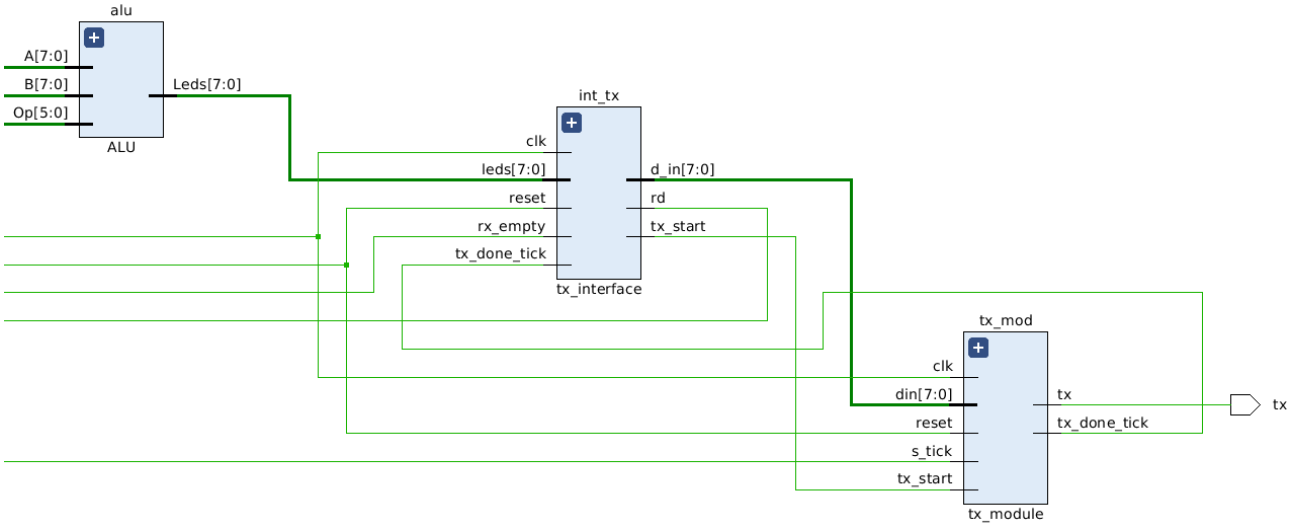


El muestreo de la señal de entrada se realiza mediante un modulo llamado baudrate generator, que cuenta la cantidad de ciclos de clock para realizar el muestreo del simbolo transmitido. Genera un Tick 16 veces por Baud Rate. Si baud rate es 19.200 ciclos por segundo, la frecuencia de muestreo debe ser 38.400 *16= 614.400 ticks por segundo. Si el clock de la placa es 100 Mhz, hay que generar un tick cada 163 ciclos de reloj. El Baud Rate Generator es un contador módulo 163.



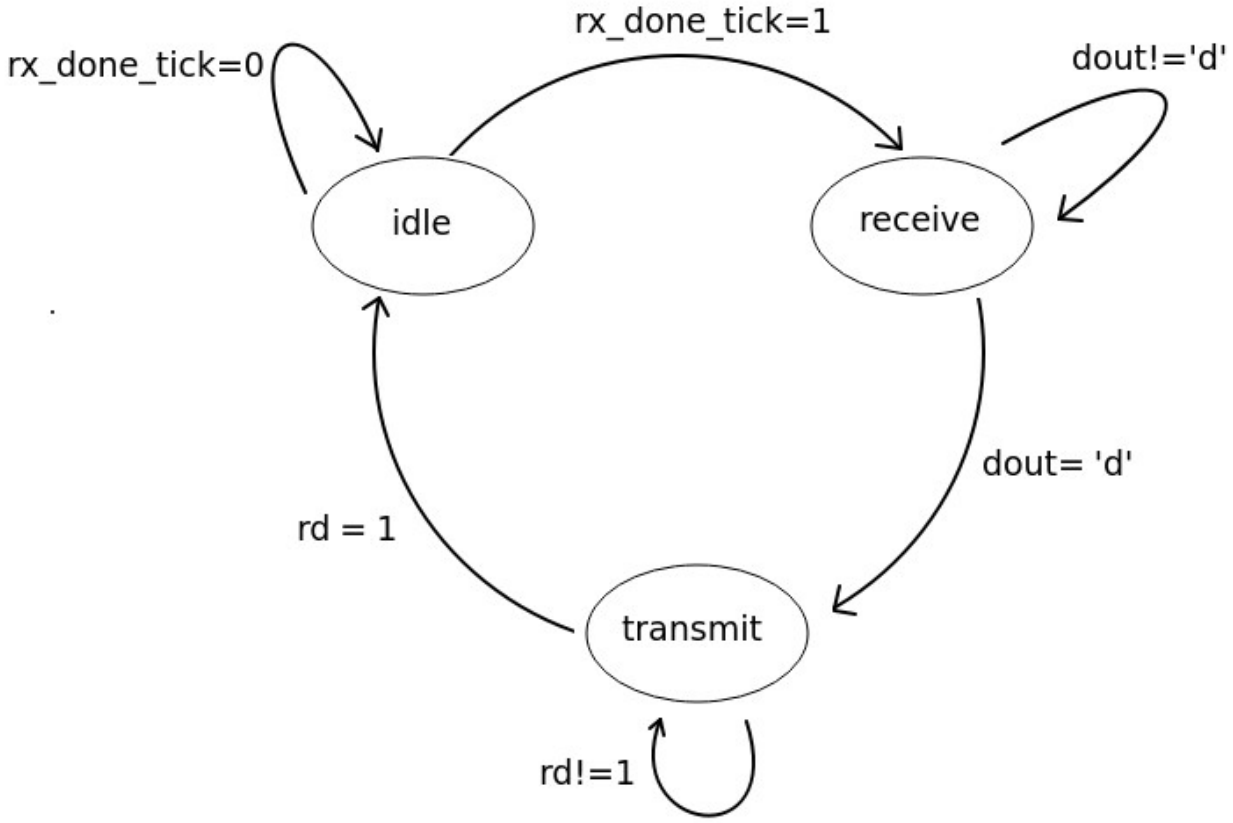
Los modulos Rx y Tx los extrajimos del libro; y desarrollamos una interfaz para comunicar con la ALU. La solucion que implementamos utiliza dos interfaces, una para la recepcion de datos y otra para la transmision.



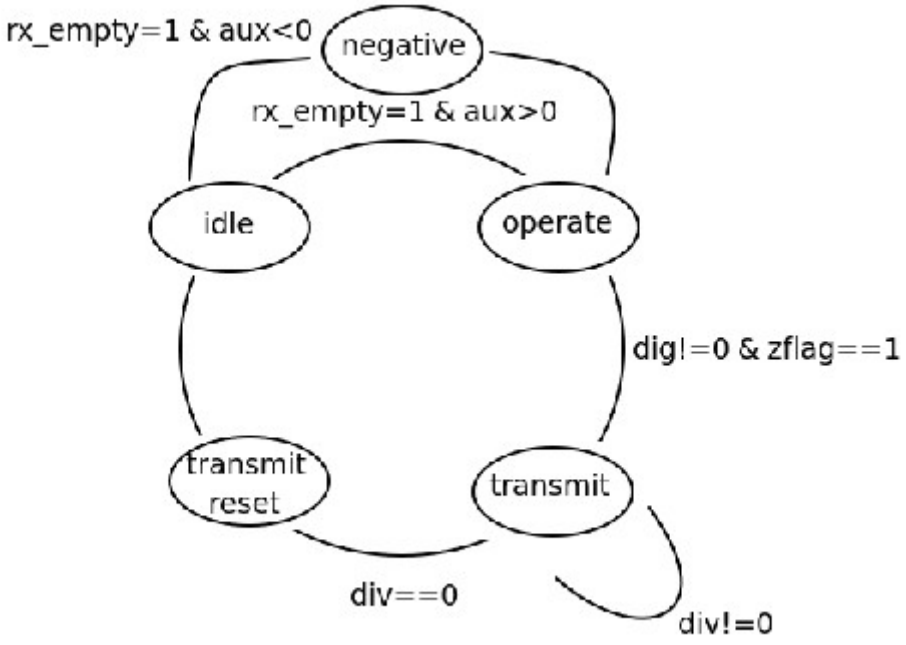


La interfaz de recepcion recibe los caracteres y los envia a la ALU dependiendo de caracteres especiales para la asignacion a las diferentes salidas; y cuando se recibe el caracter de finalizacion avisa a la interfaz de transmision para que comienze a enviar el resultado por el modulo tx.

Diagrama de estados de interfax rx:



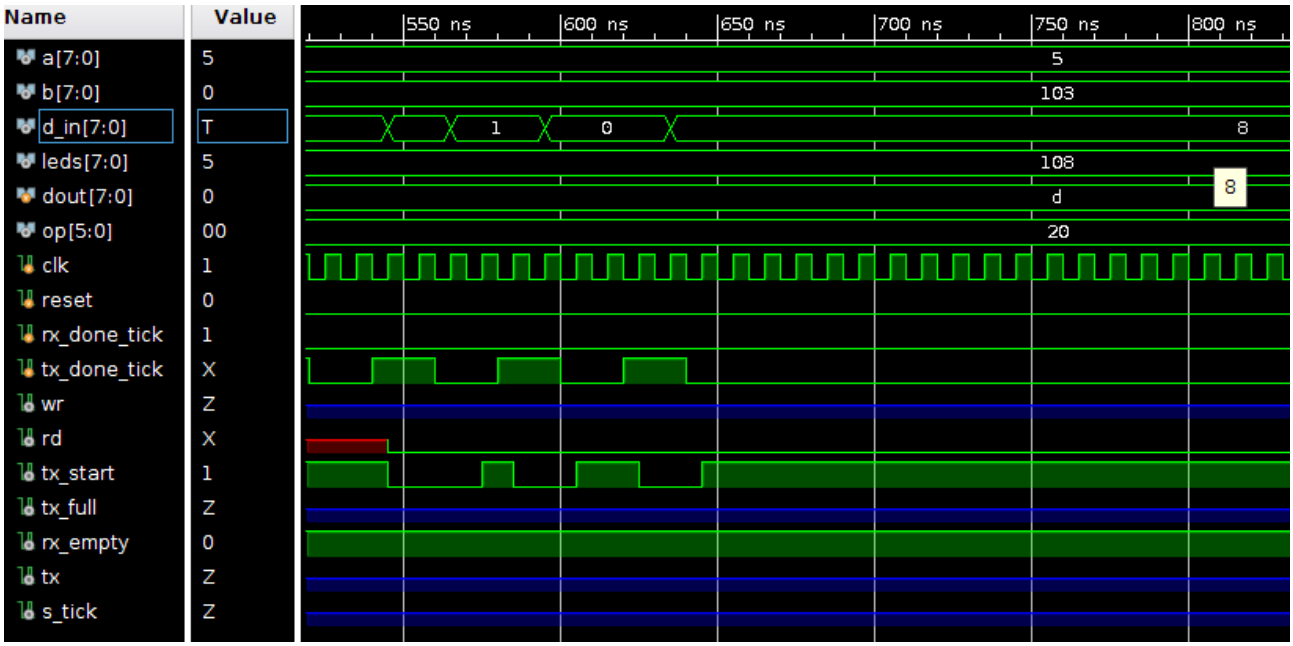
La interfaz de transmision una vez que recibe aviso de la interfaz de recepcion, envia el valor recibido de la ALU separando cada digito de centena, decena y unidad, convirtiendolos a sus valores respectivos en ASCII. Diagrama de estado de la interfaz de transmision:



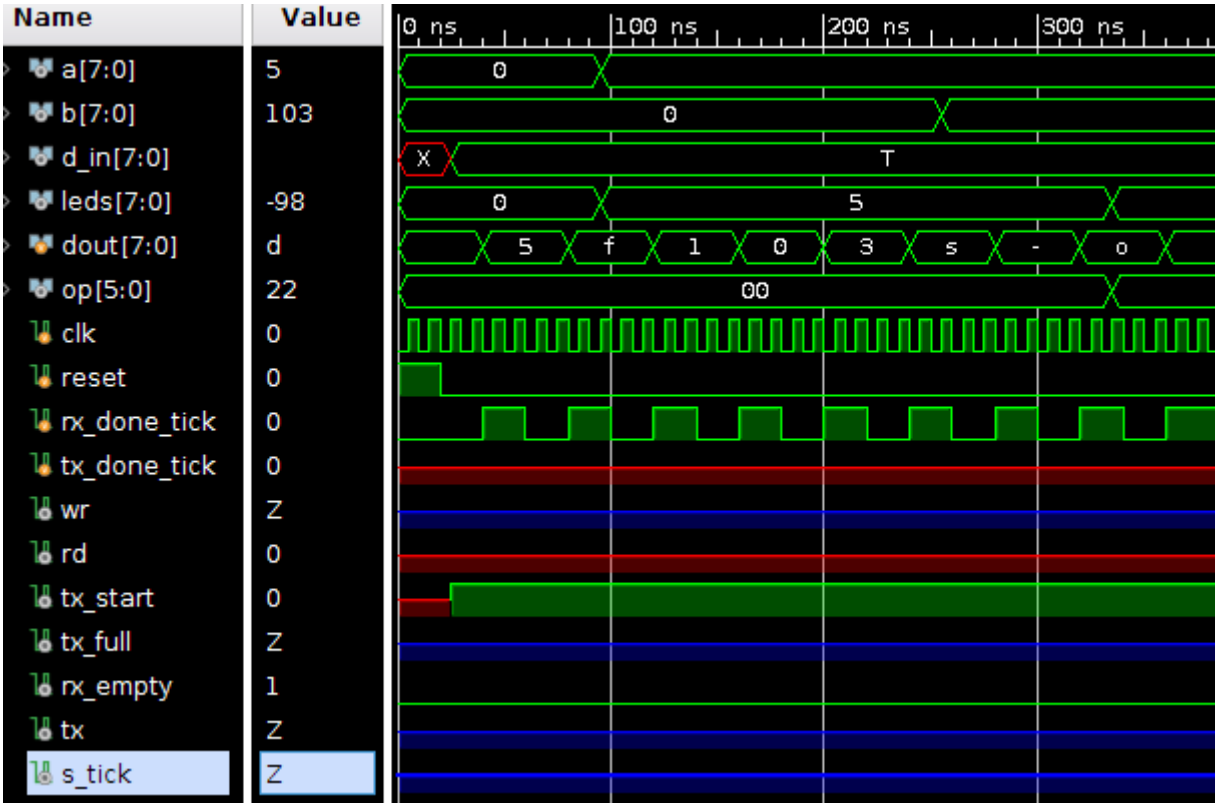
Simulaciones por medio de testbench:

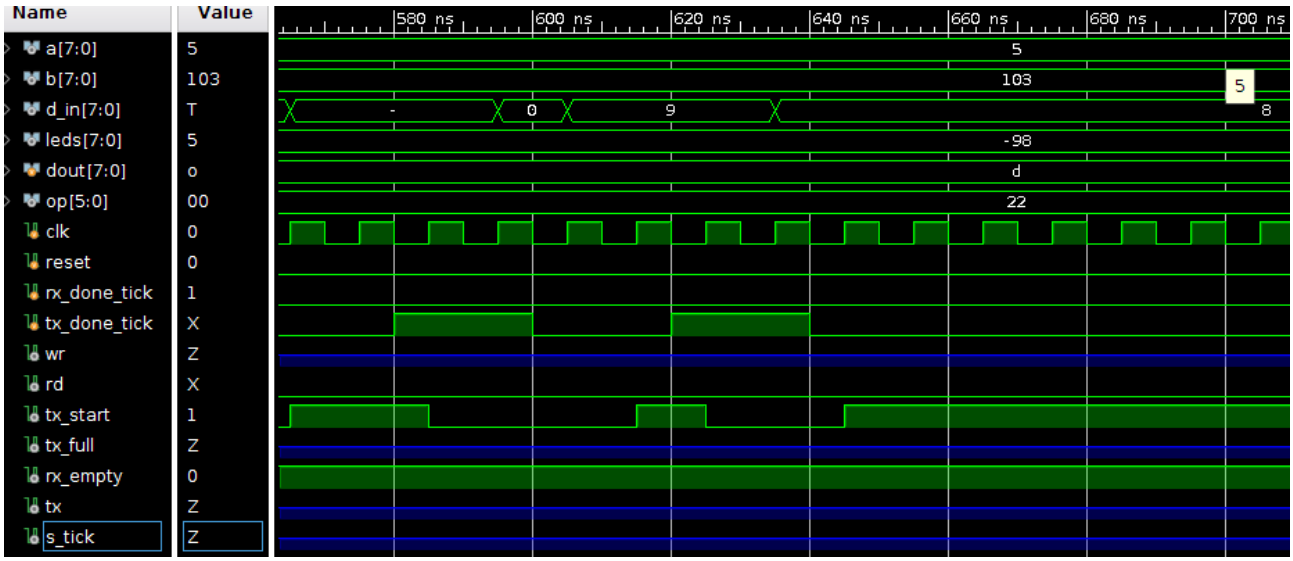
```
reset = 1;
rx_done_tick = 0;
dout = 0;
#20 reset = 0;                    // #20

#20 dout = 53;           // 5     // #40
rx_done_tick = 1;
#20 rx_done_tick = 0;             // #60
#20 dout = 102;          // f     // #80
rx_done_tick = 1;
#20 rx_done_tick = 0;             // #100
#20 dout = 49;           // 1     // #120
rx_done_tick = 1;
#20 rx_done_tick = 0;             // #140
#20 dout = 48;           // 0     // #160
rx_done_tick = 1;
#20 rx_done_tick = 0;             // #180
#20 dout = 51;           // 3     // #200
rx_done_tick = 1;
#20 rx_done_tick = 0;             // #220
#20 dout = 115;          // s     // #240
rx_done_tick = 1;
#20 rx_done_tick = 0;             // #260
#20 dout = "+";           // +     // #280
rx_done_tick = 1;
#20 rx_done_tick = 0;             // #300
#20 dout = 111;          // o     // #320
rx_done_tick = 1;
#20 rx_done_tick = 0;             // #340
#20 dout = 100;          // d     // #360
rx_done_tick = 1;
#20 rx_done_tick = 0;             // #380
rx_done_tick = 1;
#20 rx_done_tick = 0;             // #430
//#20 rx_empty = 1;
#20 tx_done_tick = 1;
#20 tx_done_tick = 0;
```

Ejecutamos una suma para comprobar el funcionamiento:



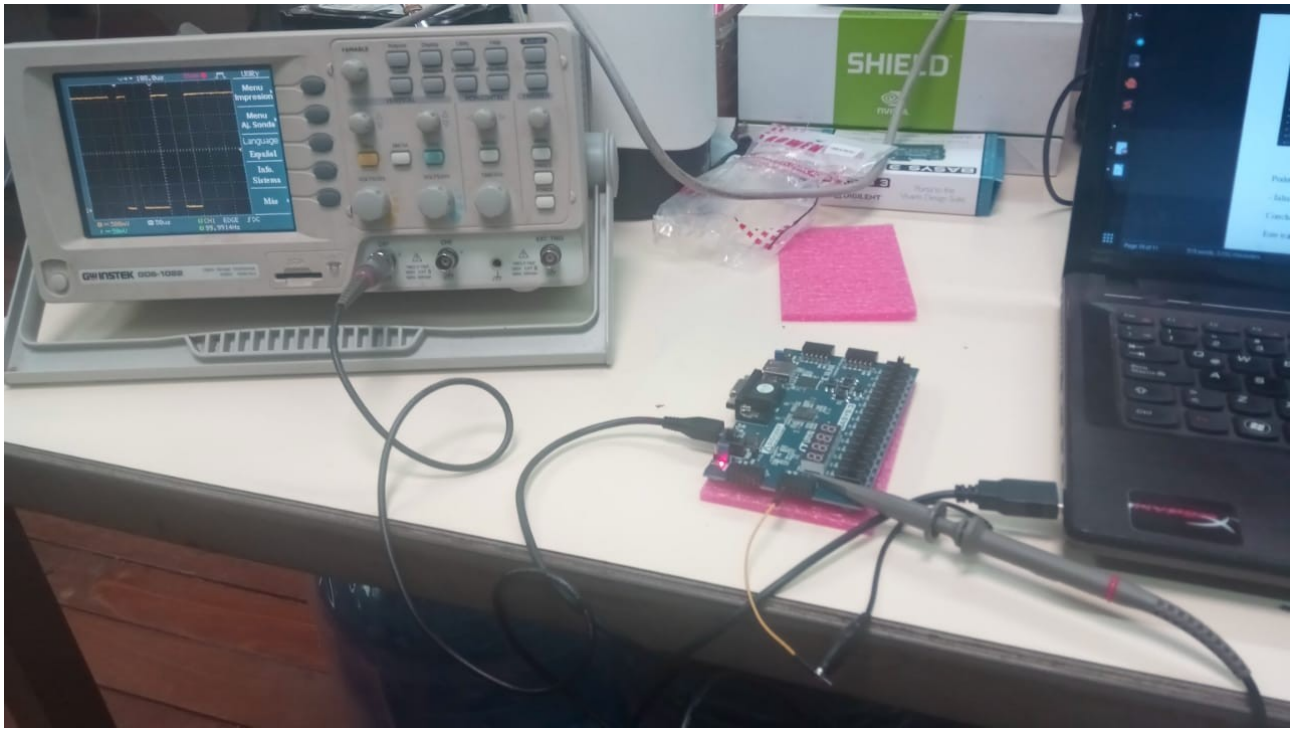
Realizamos una resta para comprobar si envia correctamente los numeros negativos:

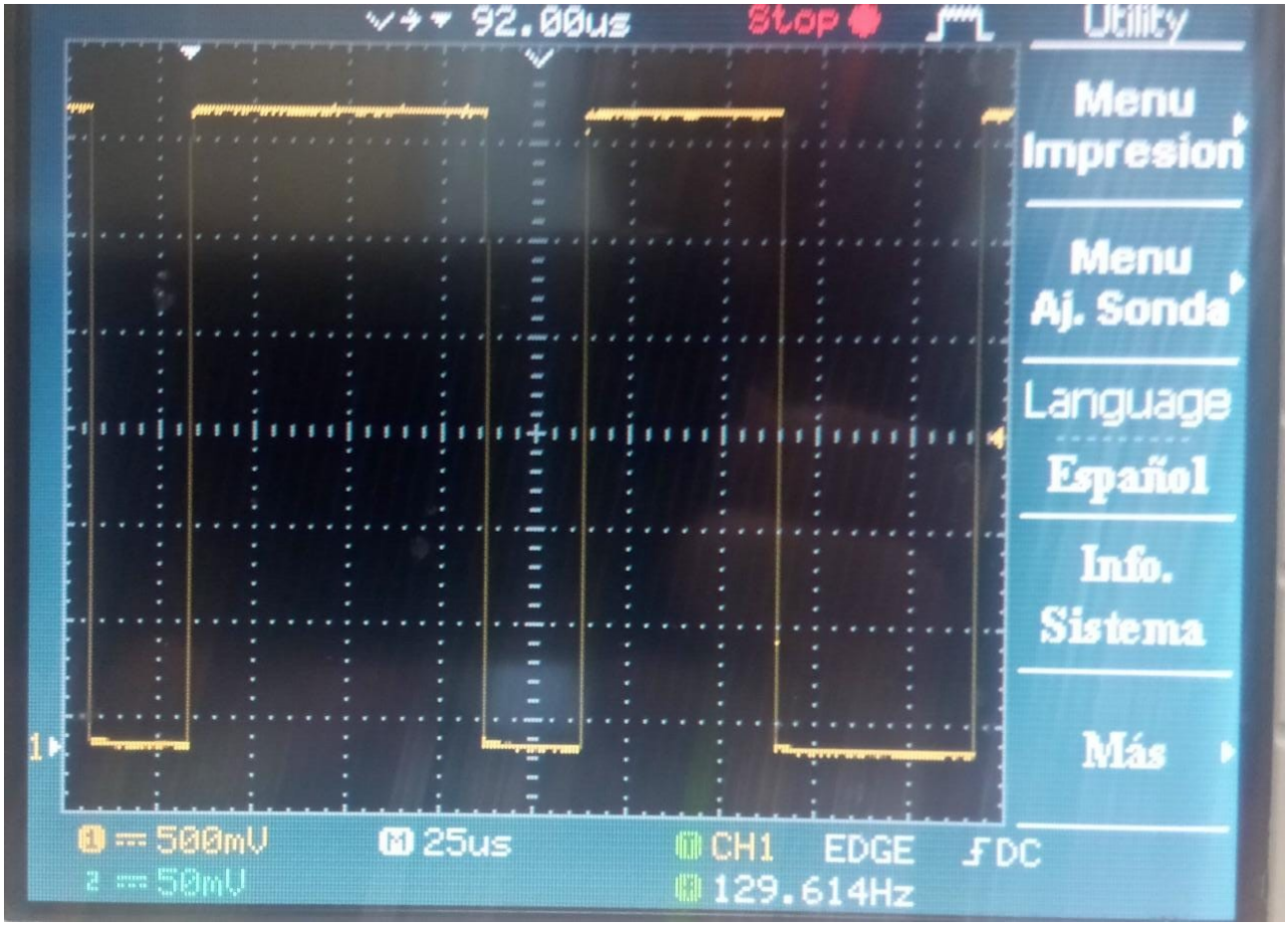




Podemos comprobar que los datos se envian correctamente.

Conectamos la placa al osciloscopio para observar el envio de datos:





Enviamos un 7 por serial. Ponemos la escala en 25 micro segundo ya que la frecuencia de comunicacion es de 38400 bits por segundo, osea 1 bit cada 26 micro sec. Podemos ver que llega el bit de start, seguido de 1110 1100 y el bit de stop. El valor en binario lo vemos invertido por el orden de llegada.

Conclusion:

Este trabajo nos permitio desarrollar maquinas de estado en verilog y comprender mejor el funcionamiento de la comunicacion serial. Pudimos obterner el resultado esperado implementando ambas interfaces de comunicación y probar su funcionamiento sobre la FPGA.