

INSTITUTO TECNOLÓGICO DE COSTA RICA

ÁREA ACADÉMICA DE INGENIERÍA MECATRÓNICA

**MT 4001
Electrónica Digital**

Tarea de diseño 1

David Rodríguez Camacho – 2019024546
Andrew Eliécer Vargas Puffenberger – 2020425926
Emmanuel Naranjo Blanco – 2019053605

Profesor: Ing. Carlos Salazar García, MSc.

Semestre II - 2021

I. Desarrollo del diseño

Para la solución del problema propuesto, se dividió en distintas etapas más simples para posteriormente realizar un ensamble que cumpla con las especificaciones de diseño. Se presenta a continuación el procedimiento para llegar al circuito diseñado.


i. Requerimientos y restricciones del problema.

Se desea sumar dos números en código ASCII A y B, donde cada uno tiene unidades y decenas, para obtener como resultado de la suma corresponde un número ASCII de tres dígitos, unidades, decenas y centenas.

Para esto se sabe que en la entrada ingresan dos dígitos codificados en ASCII para A (AD y AU) para formar el número ADAU y dos dígitos codificados en ASCII para B (BD y BU). Como se muestra en la Figura 1, los dígitos del 0 al 9 tienen su respectivo código de 7 bits, donde los primeros 3 bits más significativos corresponden a 011 y los 4 bits restantes equivalen al número particular en binario puro.

Además, como restricción se tiene que realizar la suma de A y B en binario natural y posteriormente realizar la conversión a ASCII estándar; y los números siempre serán positivos.

USASCII code chart



Column Row	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	\	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(8	H	X	h	x
9	HT	EM)	9	I	Y	i	y
10	LF	SUB	*	:	J	Z	j	z
11	VT	ESC	+	;	K	[k	{
12	FF	FS	,	<	L	\	l	
13	CR	GS	-	=	M]	m	}
14	SO	RS	.	>	N	^	n	~
15	SI	US	/	?	O	_	o	DEL

numbers 0-9
their binary
codification

Figura 1. Código ASCII para los números 0 – 9.

ii. Conversión de dígitos ASCII a binario natural.

Dado que se reciben 4 números en código ASCII estándar de 7 bits (dos para A y dos para B), es necesario realizar una conversión hacia binario natural antes de realizar la suma. Para esto, es necesario colocar cada dígito en su respectiva posición de decimal o de unidad.

De este modo, se propuso multiplicar por 10 a los dígitos correspondientes a las decenas (AD y AB) y luego sumarlos su respectiva unidad (AU y BU), esto en binario natural. Entones, como se presentó anteriormente, los 4 bits menos significativos del código ASCII representan a cada número entre 0 y 9, por lo que se decidió tomar estos bits como entrada del bloque multiplicador (Figura 2). Dado que se requiere mantener la palabra constante durante la operación, se extendió agregando tres 0 al número de 4 bits obtenido y mantener las operaciones en 7 bits ya que la mayor decena es 90 y requiere esta cantidad de bits para representarla. En la siguiente sección se detallará el proceso de multiplicación, de momento basta saber que en ingresa un número binario de 7 bits entre 0 y 9, y sale un número de 7 bits multiplicado por 10.

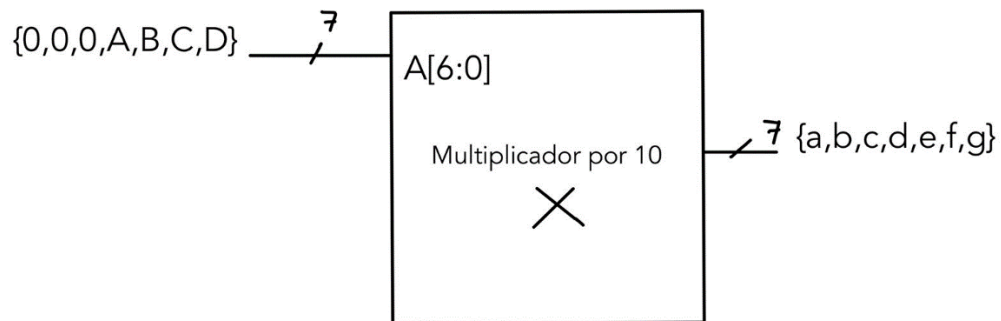


Figura 2. Bloque de multiplicación por 10.

Una vez obtenidas las decenas, se procede a sumar las unidades para así obtener el número final en binario. Se realizó el mismo procedimiento de utilizar los 4 bits menos significativos del código ASCII, se ajustó la palabra a 7 bits agregando tres 0 y luego se realizó la suma mediante un sumador de 7 bits.

Este procedimiento es necesario hacerlo para A y para B como se muestra en la Figura 3. De este modo, se logró la conversión a binario natural para cada número A y B. En este caso, no se requiere el uso del acarreo dado que se mantiene la palabra constante y no hay desbordamiento.

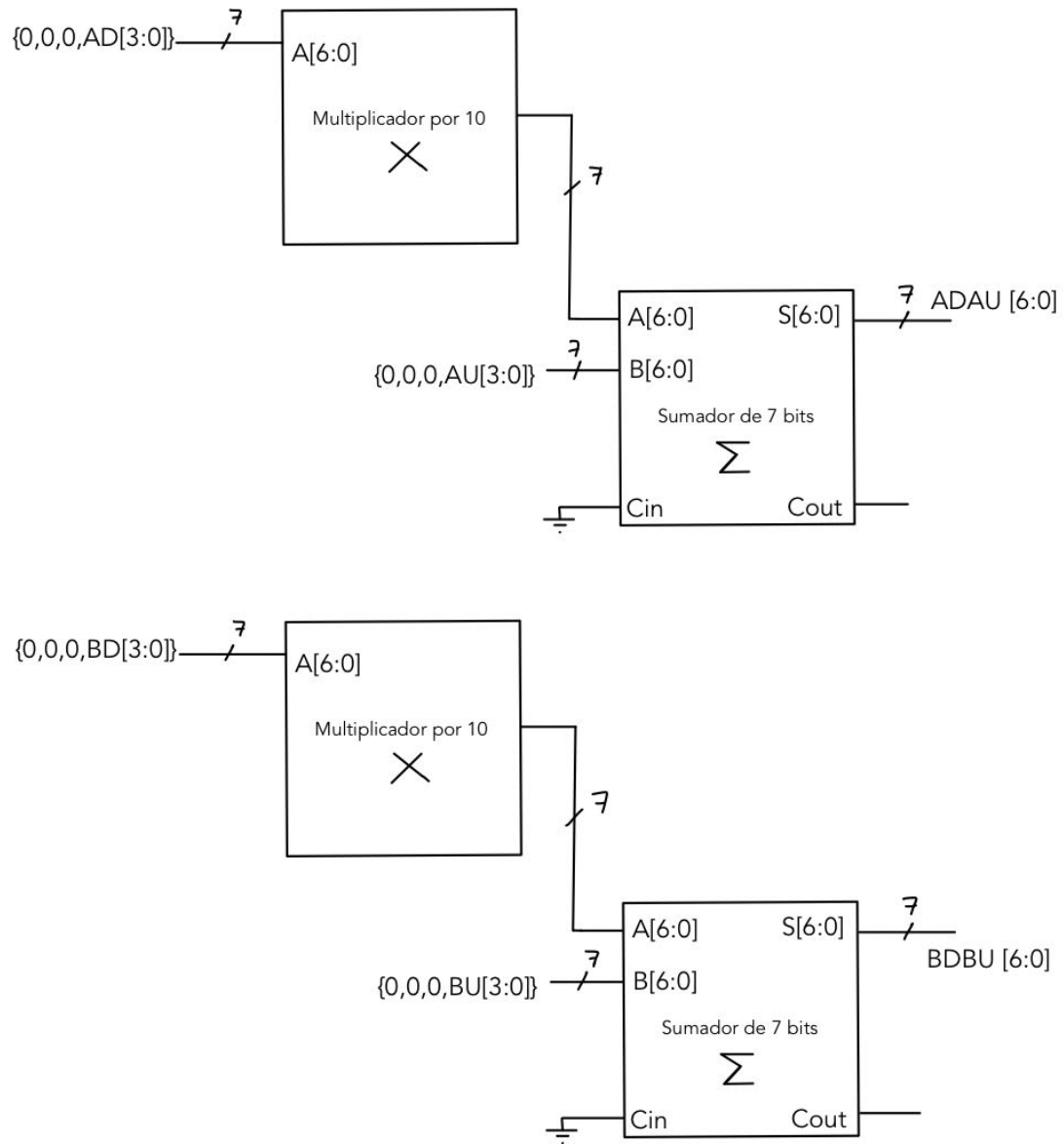


Figura 3. Conversión de A y B a binario natural.

iii. Multiplicador.

El bloque que realiza la multiplicación por 10 se indica en la Figura 4. Para obtener dicha configuración se realizó una tabla de verdad para analizar el comportamiento de cada salida al realizar la multiplicación, esto es necesario únicamente para los números binarios de 0 a 9 (Tabla 1). En esta se tiene que los cuatro bits menos significativos de la entrada son los que tienen relevancia para obtener las funciones ya que los tres MSB son ceros. Además, el bloque consiste en una operación generalizada, donde A corresponde al MSB y D al LSB para los 4 bits obtenidos del código ASCII, y las salidas son un número {a, b, c, d, e, f, g}.

Tabla 1. Tabla de verdad para obtener el bloque de multiplicación.

Entradas				Multiplicación por 10				Salidas							Resultado decimal
A	B	C	D	W	X	Y	Z	a	b	c	d	e	f	g	
0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	0	0	0	1	0	1	0	10
0	0	1	0	1	0	1	0	0	0	1	0	1	0	0	20
0	0	1	1	1	0	1	0	0	0	1	1	1	1	0	30
0	1	0	0	1	0	1	0	0	1	0	1	0	0	0	40
0	1	0	1	1	0	1	0	0	1	1	0	0	1	0	50
0	1	1	0	1	0	1	0	0	1	1	1	1	0	0	60
0	1	1	1	1	0	1	0	1	0	0	0	1	1	0	70
1	0	0	0	1	0	1	0	1	0	1	0	0	0	0	80
1	0	0	1	1	0	1	0	1	0	1	1	0	1	0	90

Una vez realizada la tabla de verdad, se realizó suma de minterminos para obtener las ecuaciones para cada salida y se simplificó utilizando un software de álgebra de Boole.

$$a = A + BCD$$

$$b = A'BC' + A'BD'$$

$$c = B'C + BC'D + CD' + A$$

$$d = A'B'D + A'BD' + B'C'D$$

$$e = C$$

$$f = D$$

$$g = 0$$

En resumen, en la Tabla 2 se muestra la tabla de verdad y en la Figura 4 la configuración de las compuertas lógicas.

Tabla 2. Resumen para el circuito multiplicador.

#	Entradas				Salidas						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	1	0
2	0	0	1	0	0	0	1	0	1	0	0
3	0	0	1	1	0	0	1	1	1	1	0
4	0	1	0	0	0	1	0	1	0	0	0
5	0	1	0	1	0	1	1	0	0	1	0
6	0	1	1	0	0	1	1	1	1	0	0
7	0	1	1	1	1	0	0	0	1	1	0
8	1	0	0	0	1	0	1	0	0	0	0
9	1	0	0	1	1	0	1	1	0	1	0
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

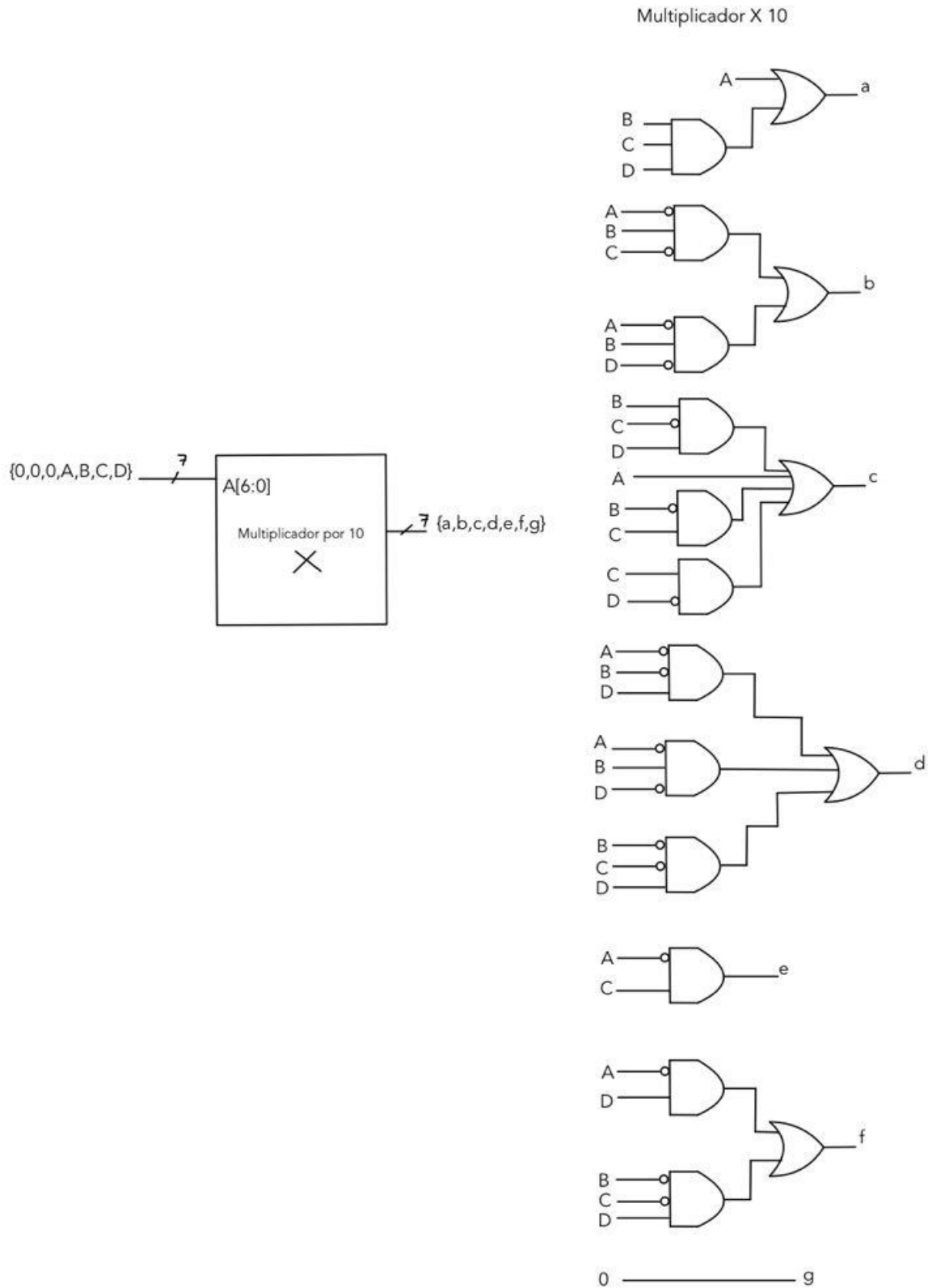


Figura 4. Esquemático del circuito multiplicador propuesto.

iv. Suma de los números convertidos a binario.

Una vez obtenidos los números A y B en su correspondiente representación binaria de 7 bits (ADAU y BDBU respectivamente) se procedió a sumar los resultados mediante un sumador de 7 bits como indica la Figura 5. En este bloque, se tiene como resultado un número de 7 bits para todo aquel valor entre 0 y 127 decimal y un resultado de 8 bits para aquellos números desde 128 hasta 198 decimal. Por lo que para este sumador sí se requiere el uso del acarreo de salida.

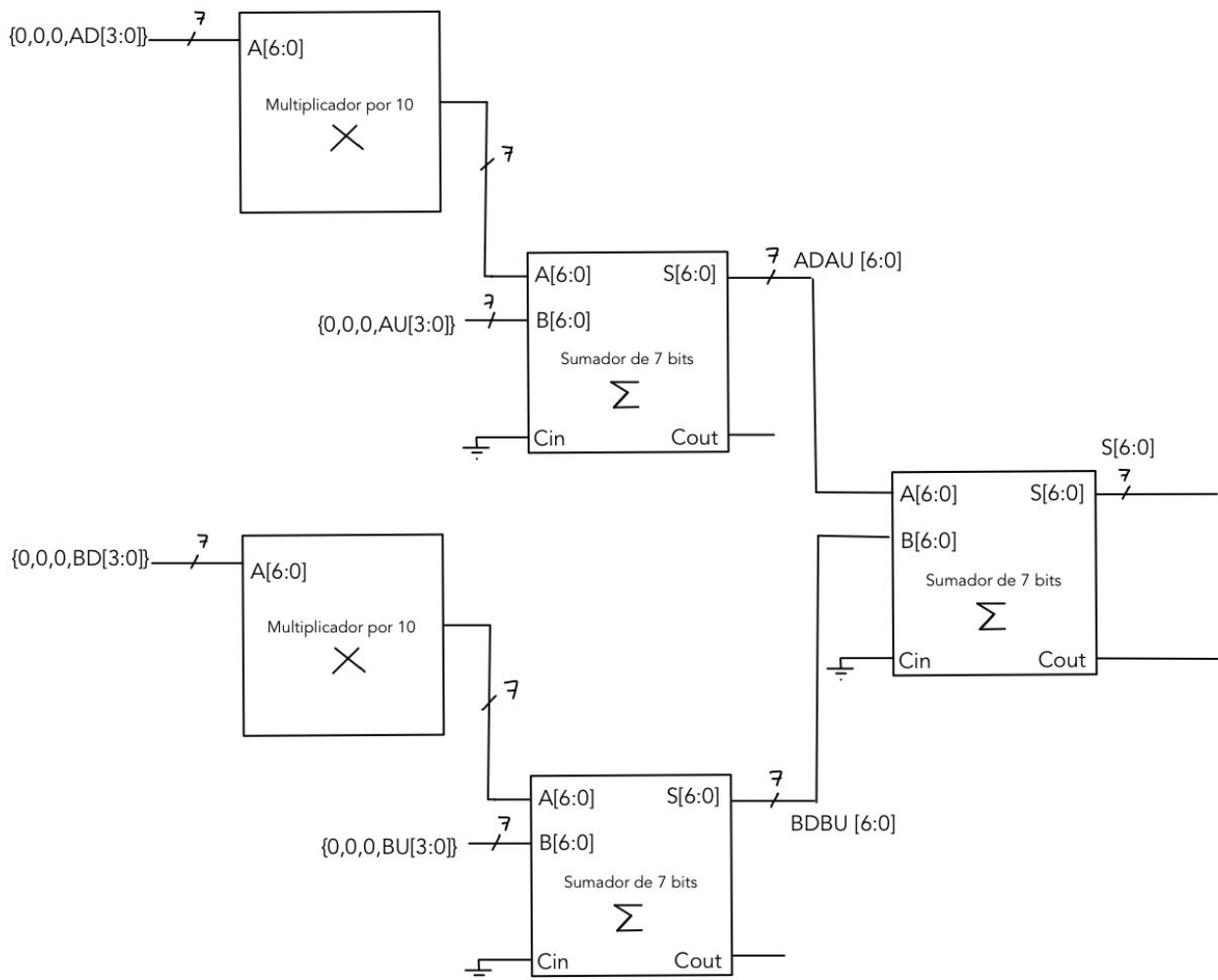


Figura 5. Operación suma en binario natural.

v. Convertidor binario natural a ASCII.

Ahora, ya que se conoce el resultado de la suma $A + B$ en binario, es necesario convertir este número a su correspondiente codificación ASCII estándar. Para esto se divide la salida Y en YC, YD y YU.

Primeramente se diseñó un convertidor de binario natural de 8 bits a BCD. Esto se realizó al utilizar una serie de bloques como el de la Figura 6, cuya estructura interna se calculó bajo la teoría expuesta por Education @ B.Tech (2016). En esta se indica que la conversión se realiza al desplazar de 1 bit en 1 bit a la derecha y comparando grupos de bits en las secciones *unidad o decena*. Donde si estos superan o igualan a 5_{10} se suma 3_{10} a estos bits, de lo contrario se mantiene todo igual y se sigue realizando el desplazamiento. Para comprender esto se invita a analizar el ejemplo 1.

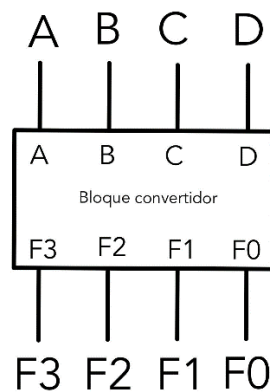


Figura 6. Bloque base para realizar la conversión de binario a BCD.

Ejemplo 1: Conversión del número $1111\ 1111_2$.

	Centena	Decena	Unidad	Binario
				1111 1111
			1	1111 111
			11	1111 11
			111	1111 1
Suma de 3			1010	1111 1
		1	0101	1111
Suma de 3		1	1000	1111
		11	0001	111
		110	0011	11
Suma de 3		1001	0011	11
	1	0010	0111	1
Suma de 3	1	0010	1010	1
	10	0101	0101	
BCD	2	5	5	

De este modo, al saber que en BCD el número más grande permitido es el 9, se realizó la tabla de verdad mostrada en la Tabla 3 y se obtuvieron las ecuaciones respectivas F3, F2, F1 y F0 para las entradas A, B, C y D.

Tabla 3. Tabla de verdad para el módulo de conversión de la Figura 6.

#	Entradas				Salidas			
	A	B	C	D	F3	F2	F1	F0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	X	X	X	X
11	1	0	1	1	X	X	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

Sus respectivas ecuaciones se muestran a continuación:

$$F3 = A + BD + BC$$

$$F2 = AD + BC'D'$$

$$F1 = B'C + CD + AD'$$

$$F0 = AD' + A'B'D + BCD'$$

Así, en la Figura 7, se muestra las compuertas lógicas que componen el bloque de la Figura 6. Además, para el caso de 8 bits, Education @ B.Tech (2016) recomienda realizar la conexión presentada en la Figura 8. En esta ingresan 8 bits, aquellos obtenidos de la suma de A y B (Cout, S[6:0]). Además, la salida Y se subdivide en las unidades (Y[3:0]), decenas (Y[7:4]) y centenas ({0,0,Y[9:8]}), todos números BCD de 4 bits.

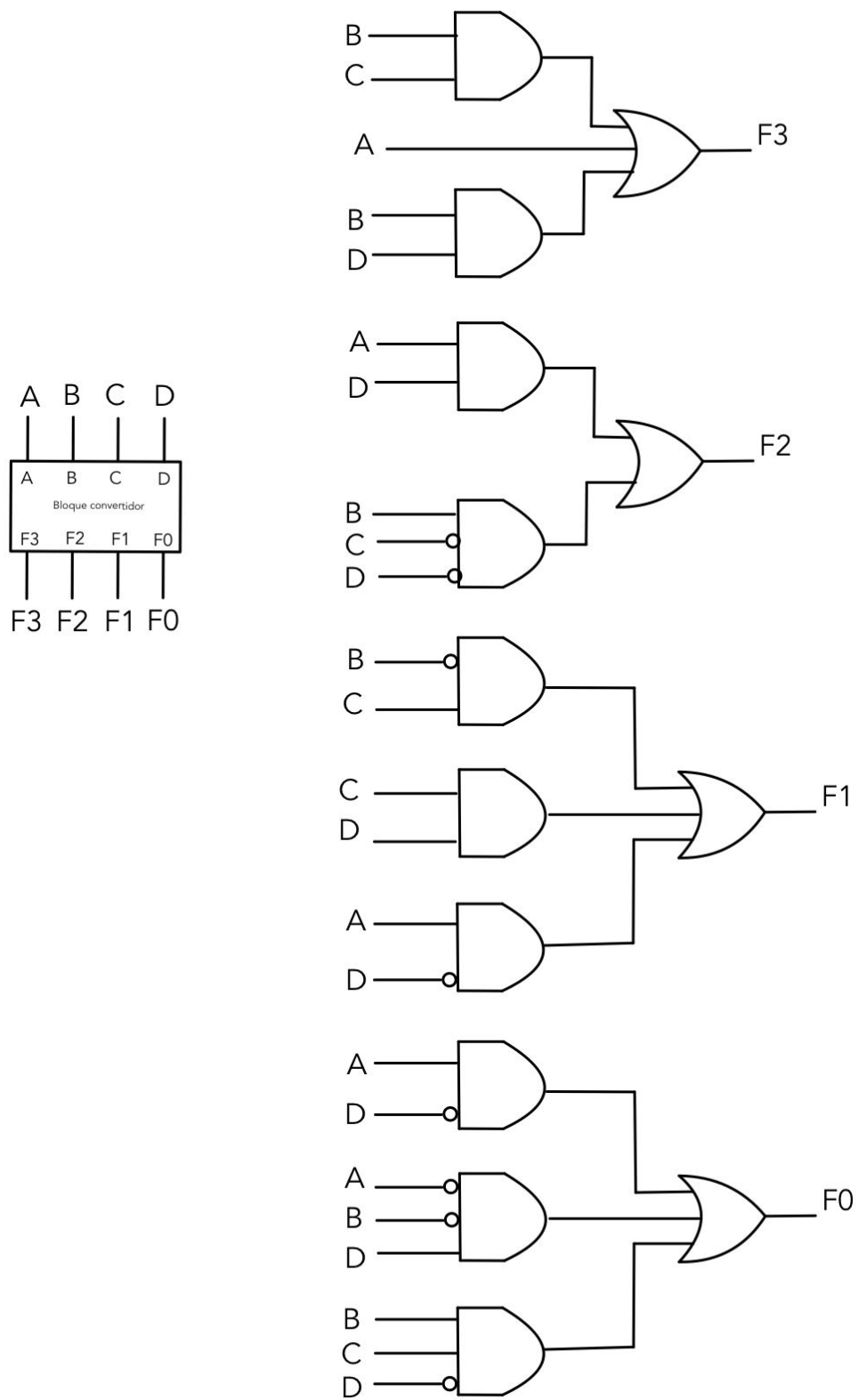


Figura 7. Diseño propuesto para el bloque comparador de la Figura 6.

Convertidor binario de 8 bits a BCD

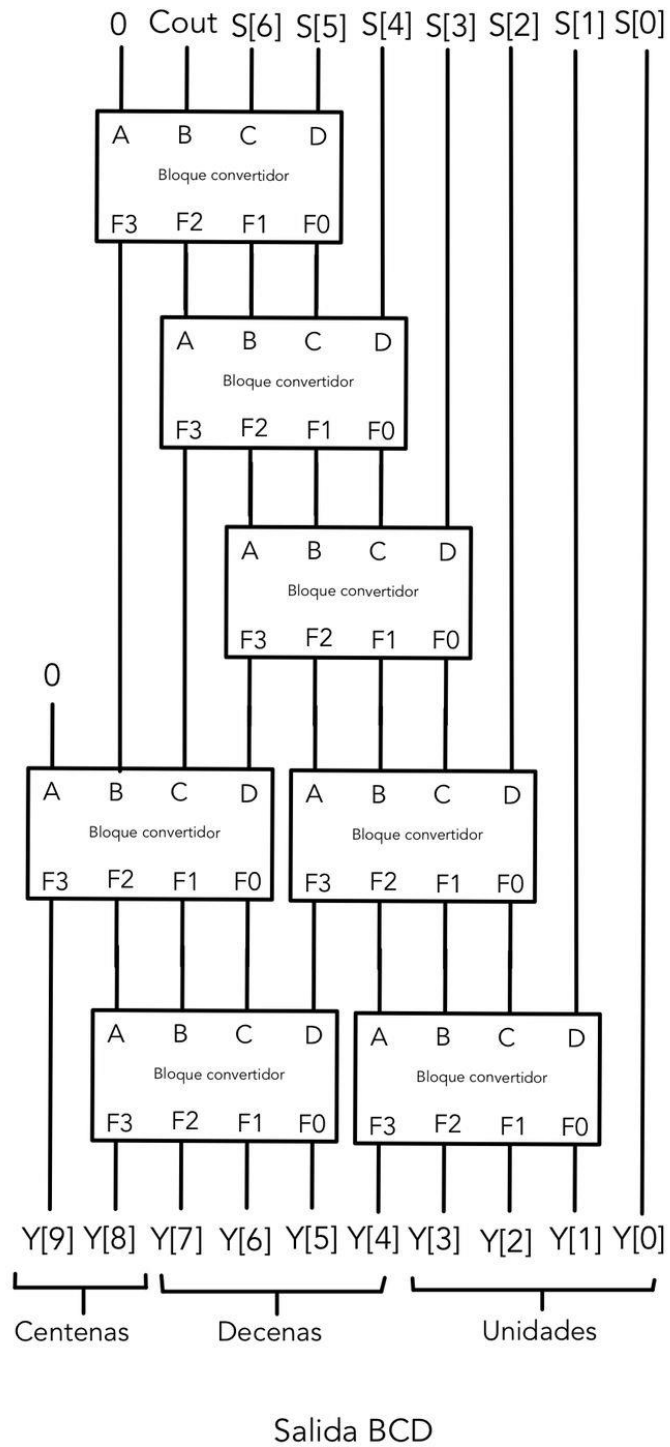


Figura 8. Diseño propuesto para el convertidor de binario 8 bits a BCD.

vi. Obtención del número ASCII.

Ahora, una vez diseñado el convertidor, se ensambló junto con las demás particiones del problema, y para mejor análisis del modelo propuesto, se decidió colocar el convertidor binario de 8 bits de la Figura 8 en un único bloque cuyas entradas consisten en los 7 bits de la suma S realizada en binario natural y el acarreo de salida Cout (Figura 9).

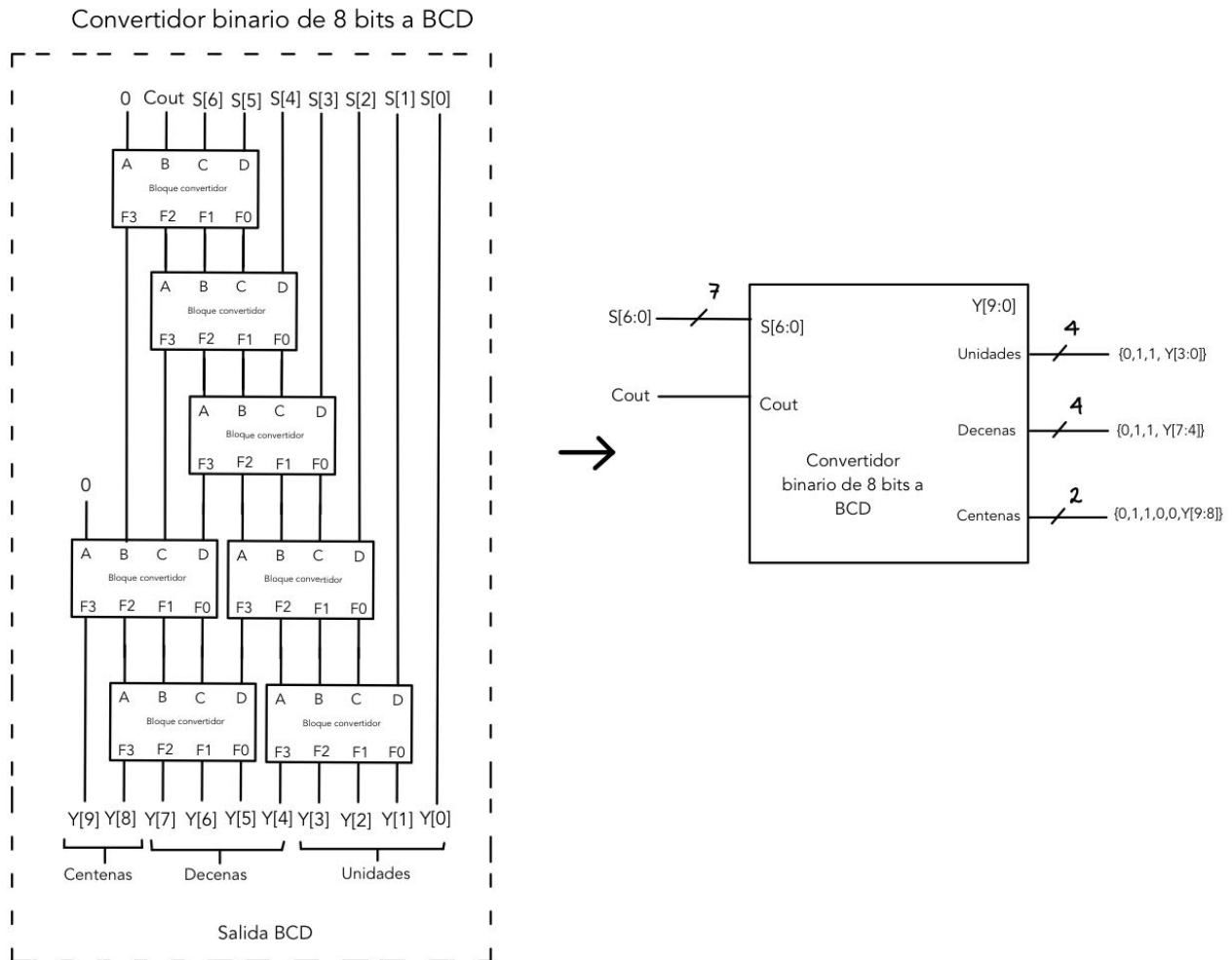


Figura 9. Simplificación del módulo convertidor de binario a BCD.

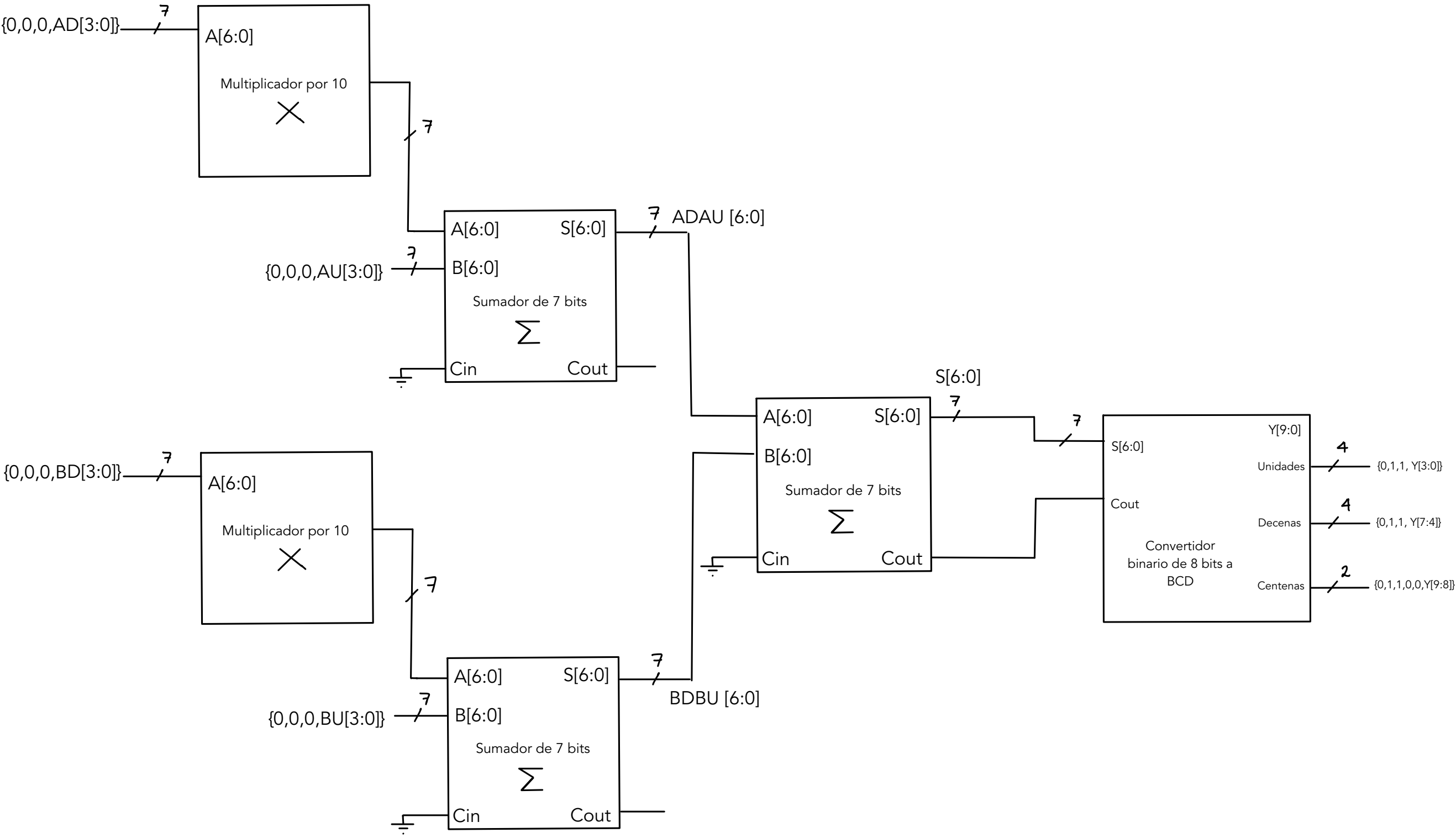
Asimismo, como se indicó en la sección de requerimientos, el código ASCII estándar de 7 bits para los números corresponde al código 011 más el número de 4 bits desde 0 hasta 9. Por lo tanto, a las salidas del convertidor binario – BCD se les concatenó 011 y a cada número correspondiente de unidades, decenas y centenas se le llamó YU, YD y YC respectivamente.

vii. Implementación.

Por último, se implementó cada componente requerido en Verilog, se simularon los componentes para verificar que trabajen de forma esperada y se empleó la simulación del diseño final. Los archivos .v utilizados en la implementación se adjuntan con el presente trabajo, en la carpeta *implementation*.

II. Diseño Final

Se presenta a continuación el circuito final diseñado.



Salida Final
 $\{0,1,1, Y[3:0]\} \Rightarrow YU$
 $\{0,1,1, Y[7:4]\} \Rightarrow YD$
 $\{0,1,1,0,0,Y[9:8]\} \Rightarrow YC$

Referencias

Education @ B.Tech. (2016, 14 de mayo). *Lesson 32 Binary to BCD Converter*. [YouTube].

<https://www.youtube.com/watch?v=sdH2lKiuCjg>