EA773 Laboratório de Circuitos Lógicos
Relatório Roteiro 1

Autor: Mateus Henrique Silva Araújo

RA: 184940

Data: 27/08/2022

1. Projeto do módulo “complementa”:

1.1. Escopo:

Projeto de um circuito combinacional capaz de realizar o complemento bit-a-bit de uma entrada de 4 bits, mediante a permissão de uma entrada de controle (enable).

1.2. Especificação de alto-nível:

Entradas:

$\underline{i} = (i_3, i_2, i_1, i_0)$ com $i_k \in \{0,1\}$ e $k = 0,1,2,3$;
$cmp \in \{0,1\}$ (entrada enable)

Saídas:

$\underline{o} = (o_3, o_2, o_1, o_0)$ com $o_k \in \{0,1\}$ e $k = 0,1,2,3$;

Função:

$$o_k = \begin{cases} \overline{i_k} & se\ cpm = 1 \\ i_k & se\ não \end{cases}; \text{ com } k = 0,1,2,3$$

1.3. Especificação binária:

Entradas:

$\underline{i} = (i_3, i_2, i_1, i_0)$ com $i_k \in \{0,1\}$ e $k = 0,1,2,3$;
$cmp \in \{0,1\}$ (entrada enable)

Saídas:

$\underline{o} = (o_3, o_2, o_1, o_0)$ com $o_k \in \{0,1\}$ e $k = 0,1,2,3$

Função de saída: Tabela verdade

i_3	i_2	i_1	i_0	o_3	o_2	o_1	o_0
0	0	0	0	cmp	cmp	cmp	cmp
0	0	0	1	cmp	cmp	cmp	cmp'
0	0	1	0	cmp	cmp	cmp'	cmp
0	0	1	1	cmp	cmp	cmp'	cmp'
0	1	0	0	cmp	cmp'	cmp	cmp
0	1	0	1	cmp	cmp'	cmp	cmp'
0	1	1	0	cmp	cmp'	cmp'	cmp
0	1	1	1	cmp	cmp'	cmp'	cmp'
1	0	0	0	cmp'	cmp	cmp	cmp
1	0	0	1	cmp'	cmp	cmp	cmp'
1	0	1	0	cmp'	cmp	cmp'	cmp
1	0	1	1	cmp'	cmp	cmp'	cmp'
1	1	0	0	cmp'	cmp'	cmp	cmp

1	1	0	1	cmp'	cmp'	cmp	cmp'
1	1	1	0	cmp'	cmp'	cmp'	cmp
1	1	1	1	cmp'	cmp'	cmp'	cmp'

1.4. Minimizações:

o_3:	$i_1'i_0'$	$i_1'i_0$	i_1i_0	i_1i_0'
$i_3'i_2'$	cmp	cmp	cmp	cmp
$i_3'i_2$	cmp	cmp	cmp	cmp
i_3i_2	cmp'	cmp'	cmp'	cmp'
i_3i_2'	cmp'	cmp'	cmp'	cmp'

$o_3 = i_3' \cdot cmp + i_3 \cdot cmp' = i_3 \oplus cmp$

o_2:	$i_1'i_0'$	$i_1'i_0$	i_1i_0	i_1i_0'
$i_3'i_2'$	cmp	cmp	cmp	cmp
$i_3'i_2$	cmp'	cmp'	cmp'	cmp'
i_3i_2	cmp'	cmp'	cmp'	cmp'
i_3i_2'	cmp	cmp	cmp	cmp

$o_2 = i_2' \cdot cmp + i_2 \cdot cmp' = i_2 \oplus cmp$

o_1:	$i_1'i_0'$	$i_1'i_0$	i_1i_0	i_1i_0'
$i_3'i_2'$	cmp	cmp	cmp'	cmp'
$i_3'i_2$	cmp	cmp	cmp'	cmp'
i_3i_2	cmp	cmp	cmp'	cmp'
i_3i_2'	cmp	cmp	cmp'	cmp'

$o_1 = i_1' \cdot cmp + i_1 \cdot cmp' = i_1 \oplus cmp$

o_0:	$i_1'i_0'$	$i_1'i_0$	i_1i_0	i_1i_0'
$i_3'i_2'$	cmp	cmp'	cmp'	cmp
$i_3'i_2$	cmp	cmp'	cmp'	cmp
i_3i_2	cmp	cmp'	cmp'	cmp
i_3i_2'	cmp	cmp'	cmp'	cmp

$o_0 = i_0' \cdot cmp + i_0 \cdot cmp' = i_0 \oplus cmp$

1.5. Esquemático do circuito:

A figura 1 apresenta o esquemático do módulo complementa, implementado usando as funções minimizadas acima.

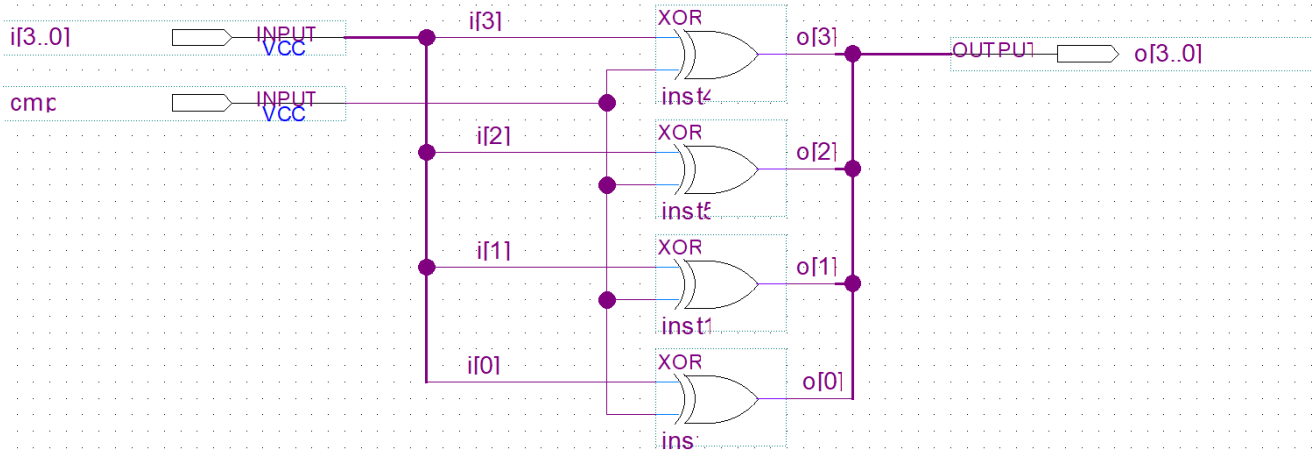

Figura 1: Diagrama esquemático do módulo complementa

1.6. Simulações:

Para as simulações, as entradas foram configuradas de modo a assumirem todas as configurações possíveis. A simulação funcional é apresentada na figura 2 e a simulação temporal, na figura 3.

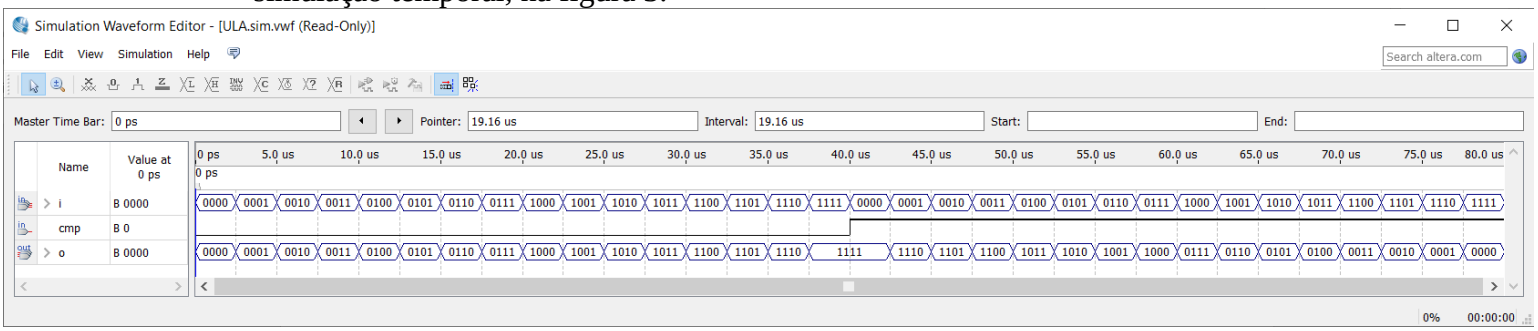

Figura 2: simulação funcional do módulo complementa

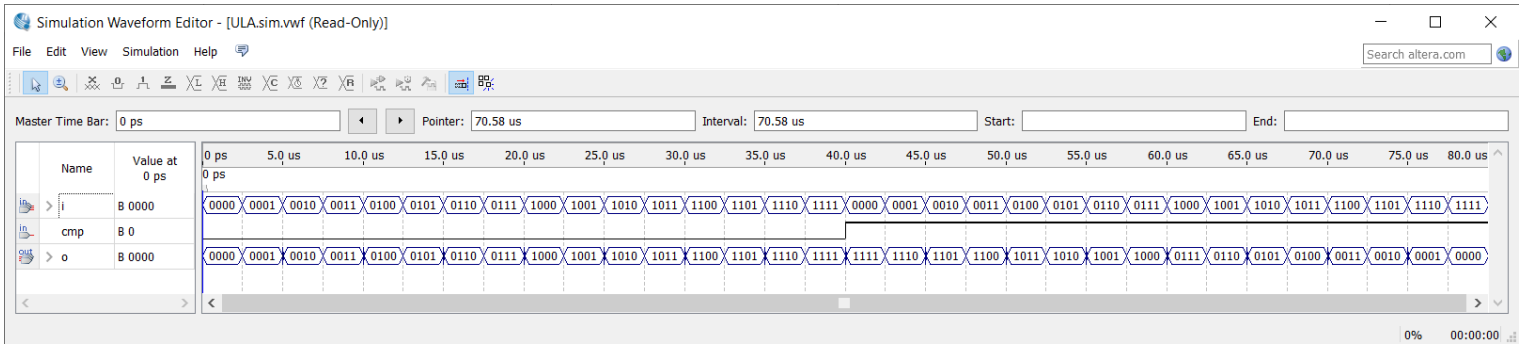

Figura 3: simulação temporal do módulo complementa

2. Projeto do módulo "zera":

2.1. Escopo:

Projeto de um combinacional capaz de receber uma entrada de 4 bits e zerá-la, mediante a permissão de uma entrada de controle (enable).

2.2. Especificação de alto-nível:

Entradas:

$\underline{i} = (i_3, i_2, i_1, i_0)$ com $i_k \in \{0,1\}$ e $k = 0,1,2,3$;
$zera \in \{0,1\}$ (entrada enable);

Saídas:

$\underline{o} = (o_3, o_2, o_1, o_0)$ com $o_k \in \{0,1\}$ e $k = 0,1,2,3$;

Função:

$$o_k = \begin{cases} 0 & se\ zera = 1 \\ i_k & se\ não \end{cases}; \text{ com } k = 0,1,2,3$$

2.3. Especificação binária:

Entradas:

$\underline{i} = (i_3, i_2, i_1, i_0)$ com $i_k \in \{0,1\}$ e $k = 0,1,2,3$;
$cmp \in \{0,1\}$ (entrada enable);

Saídas:

$\underline{o} = (o_3, o_2, o_1, o_0)$ com $o_k \in \{0,1\}$ e $k = 0,1,2,3$

Função: Tabela verdade

i_3	i_2	i_1	i_0	o_3	o_2	o_1	o_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	$zera'$
0	0	1	0	0	0	$zera'$	0
0	0	1	1	0	0	$zera'$	$zera'$
0	1	0	0	0	$zera'$	0	0
0	1	0	1	0	$zera'$	0	$zera'$
0	1	1	0	0	$zera'$	$zera'$	0
0	1	1	1	0	$zera'$	$zera'$	$zera'$
1	0	0	0	$zera'$	0	0	0
1	0	0	1	$zera'$	0	0	$zera'$
1	0	1	0	$zera'$	0	$zera'$	0
1	0	1	1	$zera'$	0	$zera'$	$zera'$
1	1	0	0	$zera'$	$zera'$	0	0
1	1	0	1	$zera'$	$zera'$	0	$zera'$
1	1	1	0	$zera'$	$zera'$	$zera'$	0
1	1	1	1	$zera'$	$zera'$	$zera'$	

2.4. Minimizações:

o_3:	$i_1'i_0'$	$i_1'i_0$	i_1i_0	i_1i_0'
$i_3'i_2'$	0	0	0	0
$i_3'i_2$	0	0	0	0
i_3i_2	$zera'$	$zera'$	$zera'$	$zera'$
i_3i_2'	$zera'$	$zera'$	$zera'$	$zera'$

$o_3 = i_3 \cdot zera'$

o_2:	$i_1'i_0'$	$i_1'i_0$	i_1i_0	i_1i_0'
$i_3'i_2'$	0	0	0	0
$i_3'i_2$	$zera'$	$zera'$	$zera'$	$zera'$
i_3i_2	$zera'$	$zera'$	$zera'$	$zera'$
i_3i_2'	0	0	0	0

$o_2 = i_2 \cdot zera'$

o_1:	$i_1'i_0'$	$i_1'i_0$	i_1i_0	i_1i_0'
$i_3'i_2'$	0	0	$zera'$	$zera'$
$i_3'i_2$	0	0	$zera'$	$zera'$
i_3i_2	0	0	$zera'$	$zera'$
i_3i_2'	0	0	$zera'$	$zera'$

$o_1 = i_1 \cdot zera'$

o_0:	$i_1'i_0'$	$i_1'i_0$	i_1i_0	i_1i_0'
$i_3'i_2'$	0	$zera'$	$zera'$	0
$i_3'i_2$	0	$zera'$	$zera'$	0
i_3i_2	0	$zera'$	$zera'$	0
i_3i_2'	0	$zera'$	$zera'$	0

$o_0 = i_0 \cdot zera'$

2.5. Esquemático do circuito:

A figura 4 apresenta o esquemático do módulo zera, implementado usando as funções de minimização acima.

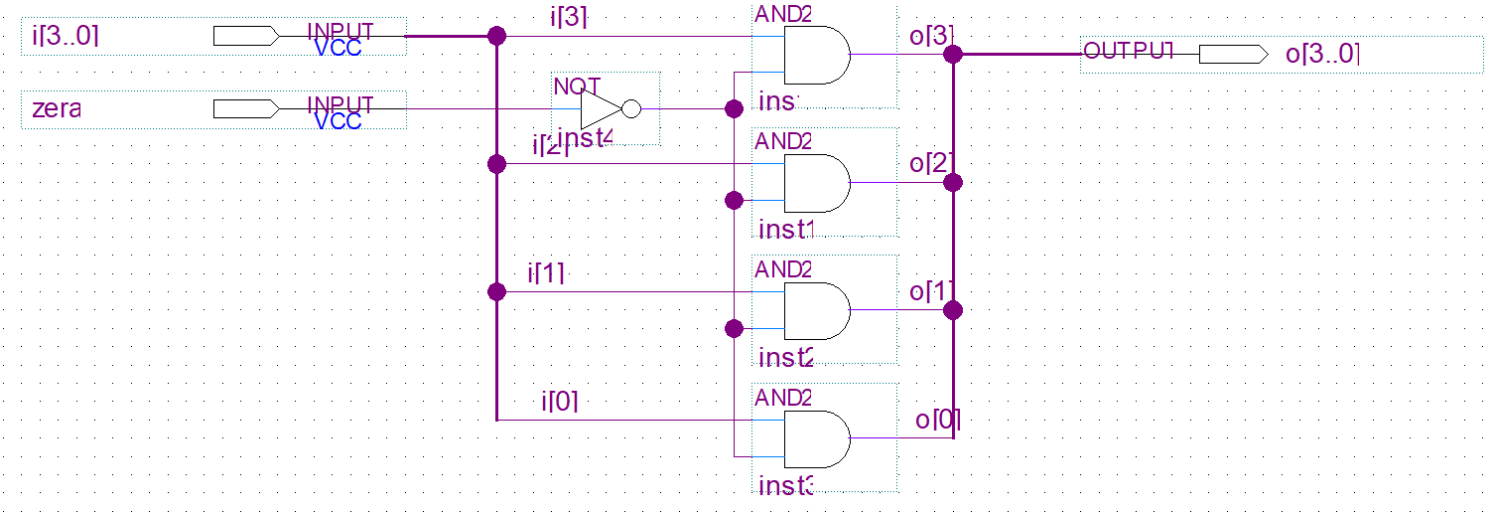

Figura 4: Diagrama esquemático do módulo zera

2.6. Simulações:

Para as simulações, as entradas foram configuradas de modo a assumirem todas as configurações possíveis. A simulação funcional é apresentada na figura 5 e a simulação temporal, na figura 6. Nota-se que, em ambos os casos, o circuito obedece ao comportamento especificado anteriormente, zerando a entrada *i* quando a entrada *zera* é igual a 1.

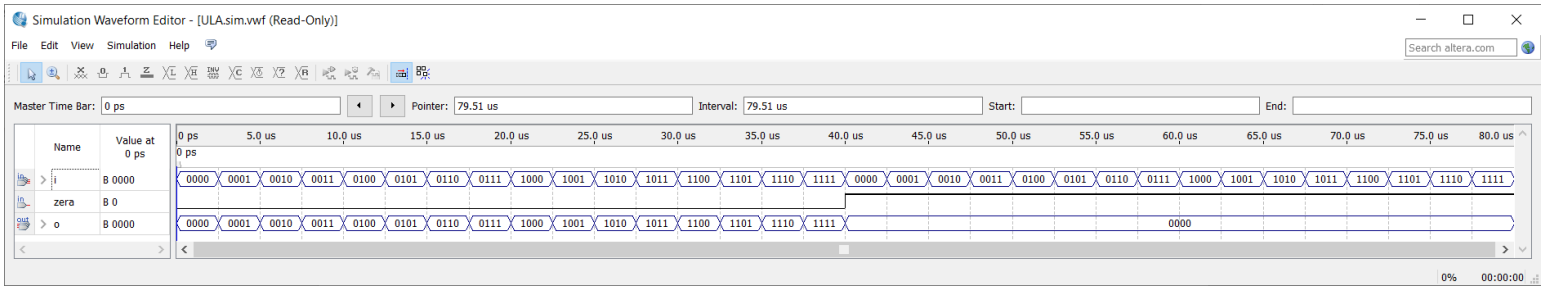

Figura 5: Simulação funcional do módulo zera

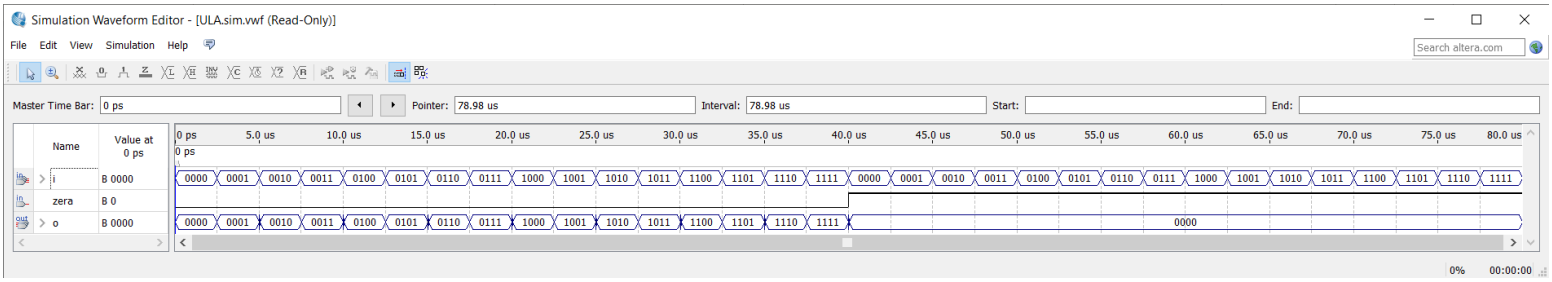

Figura 6: Simulação temporal do módulo zera

3. Projeto do módulo “somador_4_bits”:

3.1. Escopo:

Projeto de um circuito combinacional, que recebe duas entradas de 4 bits, representando dois valores inteiros, e uma entrada de carry, capaz de efetuar a soma entre os dois valores da entrada (considerando o carry fornecido). As saídas devem

ser um número de 4 bits (identificando o resultado da soma), bem como as bandeiras de estado C (carry), V (overflow), N (negativo) e Z (zero).

3.2. Módulos utilizados:

Para a implementação da função de soma, particionaremos o somador em mais dois módulos, o módulo de meio somador e o módulo somador completo (que recebe dois bits e um carry, executa a soma e retorna tanto o resultado quanto o carry).

i) Módulo "meio_somador":

Circuito combinacional que recebe dois bits, executa a soma deles retorna e retorna tanto o resultado quanto o carry.

a. Especificação binária:

Entradas:

$x_i \in \{0,1\}$
$y_i \in \{0,1\}$

Saídas:

$g_i \in \{0,1\}$ (bit de geração)
$p_i \in \{0,1\}$ (bit de propagação)

Função: Tabela verdade

x_i	y_i	g_i	p_i
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

b. Minimizações:

g_i:	y_i'	y_i
x_i'	0	0
x_i	0	1

$g_i = x_i y_i$

p_i:	y_i'	y_i
x_i'	0	1
x_i	1	0

$p_i = x_i y_i' + x_i' y_i = x_i \oplus y_i$

c. Esquemático:

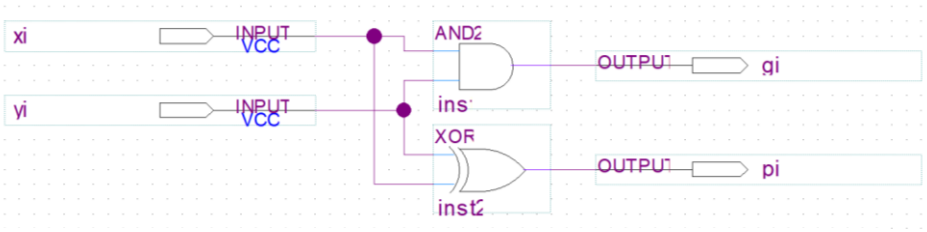

Figura 7: Diagrama esquemático do módulo meio_somador

d. Simulações:

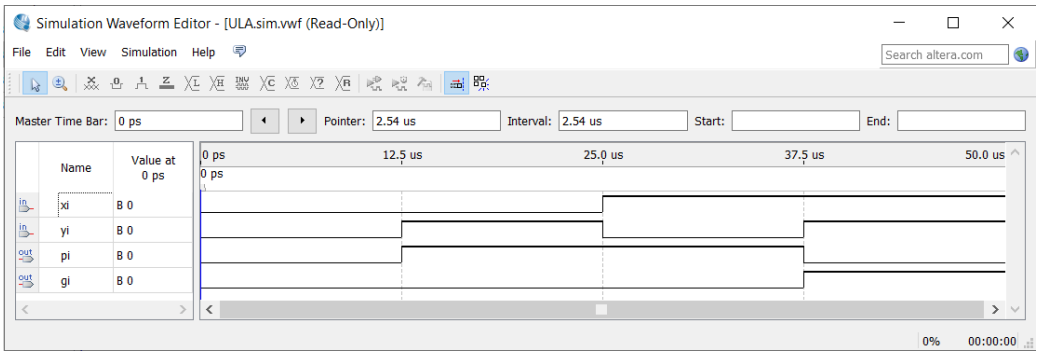

Figura 8: Simulação funcional do módulo meio_somador

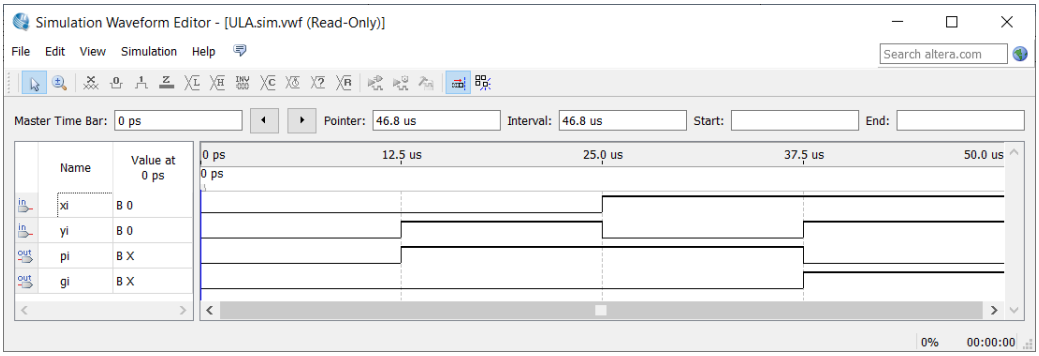

Figura 9: Simulação temporal do módulo meio_somador

ii) Módulo "somador_completo":

Circuito combinacional que recebe dois bits e um carry, executa a soma deles retorna e retorna tanto o resultado quanto o carry. Para tal, empregaremos o módulo meio somador especificado anteriormente.

a. Especificação binária:

Entradas:

$x_i \in \{0,1\}$
$y_i \in \{0,1\}$
$c_{in} \in \{0,1\}$

Saídas:

$z_i \in \{0,1\}$ (resultado da soma)
$c_{out} \in \{0,1\}$ (carry out)

Função: Tabela verdade

x_i	y_i	c_{in}	z_i	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

b. Minimização:

Usando o módulo meio somador, podemos implementar a tabela verdade acima. Para facilitar a compreensão das equações, vamos definir $\mathbb{m}_{s_{[g_i]}}(x_i, y_i)$ como a saída g_i do circuito meio somador tendo como entradas x_i e y_i, bem como $\mathbb{m}_{s_{[p_i]}}(x_i, y_i)$ como a saída p_i do circuito meio somador tendo como entradas x_i e y_i. Assim:

$$z_i = \mathbb{m}_{s[p_i]}\left(\mathbb{m}_{s[p_i]}(x_i, y_i), c_{in}\right)$$

$$c_{out} = \mathbb{m}_{s[g_i]}\left(\mathbb{m}_{s[p_i]}(x_i, y_i), c_{in}\right) + \mathbb{m}_{s[g_i]}(x_i, y_i)$$

c. Esquemático:

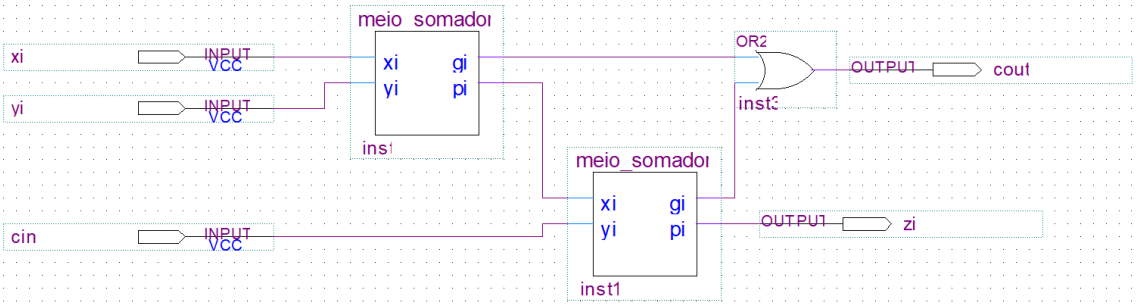

Figura 10: Diagrama esquemático do módulo somador_completo

d. Simulações:

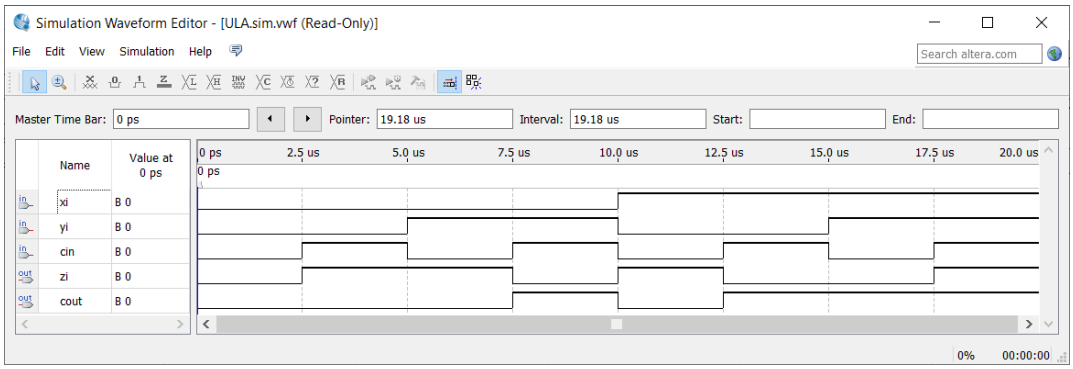

Figura 11: Simulação funcional do módulo somador_completo

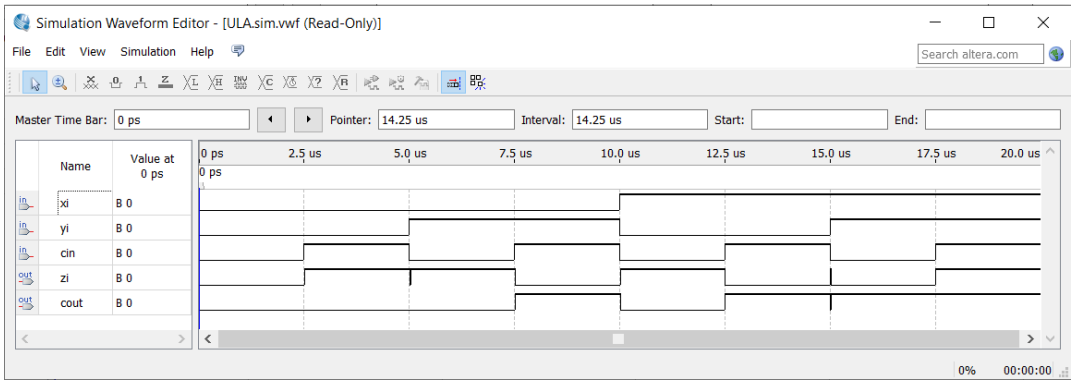

Figura 12: Simulação temporal do módulo somador_completo

3.3. Especificação de alto-nível:

Entradas:

$\underline{x} = (x_3, x_2, x_1, x_0)$ com $x_k \in \{0,1\}$ e $k = 0,1,2,3$;
$\underline{x} = (y_3, y_2, y_1, y_0)$ com $i_k \in \{0,1\}$ e $k = 0,1,2,3$;
$c_i \in \{0,1\}$ (carry in)

Saídas:

$\underline{s} = (s_3, s_2, s_1, s_0)$ com $s_k \in \{0,1\}$ e $k = 0,1,2,3$;

$C \in \{0,1\}$; $V \in \{0,1\}$; $N \in \{0,1\}$; $Z \in \{0,1\}$

Função:

$\underline{s} = \underline{x} + \underline{y}$;

$C = \begin{cases} 1 & se\ \underline{x} + \underline{y}\ gera\ carry \\ 0 & se\ não \end{cases}$; $\quad V = \begin{cases} 1 & se\ \underline{x} + \underline{y}\ gera\ overflow \\ 0 & se\ não \end{cases}$;

$N = \begin{cases} 1 & se\ \underline{s} < 0 \\ 0 & se\ não \end{cases}$; $\quad Z = \begin{cases} 1 & se\ \underline{s} = 0 \\ 0 & se\ não \end{cases}$

3.4. Especificação binária:

Entradas:

$\underline{x} = (x_3, x_2, x_1, x_0)$ com $x_k \in \{0,1\}$ e $k = 0,1,2,3$;
$\underline{x} = (y_3, y_2, y_1, y_0)$ com $i_k \in \{0,1\}$ e $k = 0,1,2,3$;
$c_i \in \{0,1\}$ (carry in)

Saídas:

$\underline{s} = (s_3, s_2, s_1, s_0)$ com $s_k \in \{0,1\}$ e $k = 0,1,2,3$;

$C \in \{0,1\}$; $V \in \{0,1\}$; $N \in \{0,1\}$; $Z \in \{0,1\}$

Função:

Usando o módulo de somador completo especificado acima, podemos realizar a soma completa de cada bit considerando os dois bits de entrada e o carry da soma do bit menos significativo. Dessa forma, primeiramente definimos $\mathbb{S}_{[z_i]}(x, y, c)$ como a saída z_i do somador completo tendo as entradas $x_i = x, y_i = y$ e $c_{in} = c$, $\mathbb{S}_{[c_{out}]}(x, y, c)$ como a saída c_{out} do somador completo tendo as entradas $x_i = x, y_i = y$ e $c_{in} = c$ e c_{out_k} dado por :

$$c_{out_k} = \begin{cases} \mathbb{S}_{[c_{out}]}(x_0, y_0, c_i) & se\ k = 0 \\ \mathbb{S}_{[c_{out}]}(x_k, y_k, c_{out_{k-1}}) & se\ k = 1,2,3 \end{cases}$$

Teremos que a saída s_k , com $k = 0, 1, 2, 3$ será dada por:

$s_k = \mathbb{S}_{[z_i]}(x_k, y_k, c_{out_k})$

Para as demais saídas, teremos:

$C = \mathbb{S}_{[c_{out}]}(x_3, y_3, c_{out_2})$, pois o carry produzido pela soma completa é igual ao carry do bit mais significativo;

$N = s_3$, pois o bit mais significativo do resultado define se o resultado é negativo (caso $s_3 = 1$) ou positivo (caso $s_3 = 0$);

$Z = s_3' s_2' s_1' s_0'$, pois o resultado só é zero se todos os bits do resultado forem zero;

Para a bandeira de estado V, como ela depende somente dos bits x_3, y_3 e s_3, tem-se a seguinte tabela verdade:

x_3	y_3	s_3	V
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

3.5. Minimizações:

As saídas $\underline{s}$, C, Z e N já tiveram suas equações de saída reduzidas a partir de sua definição. Para a flag V, fazemos:

V:	$y_3's_3'$	$y_3's_3$	y_3s_3	y_3s_3'
x_3'	0	1	0	0
x_3	0	0	0	1

$$V = x_3'y_3's_3 + x_3y_3s_3'$$

3.6. Esquemático do circuito:

Assim como descrito anteriormente, o módulo do circuito "somador_4_bits" foi implementado utilizando o módulo "somador_completo". O circuito resultante está disposto na figura 13.

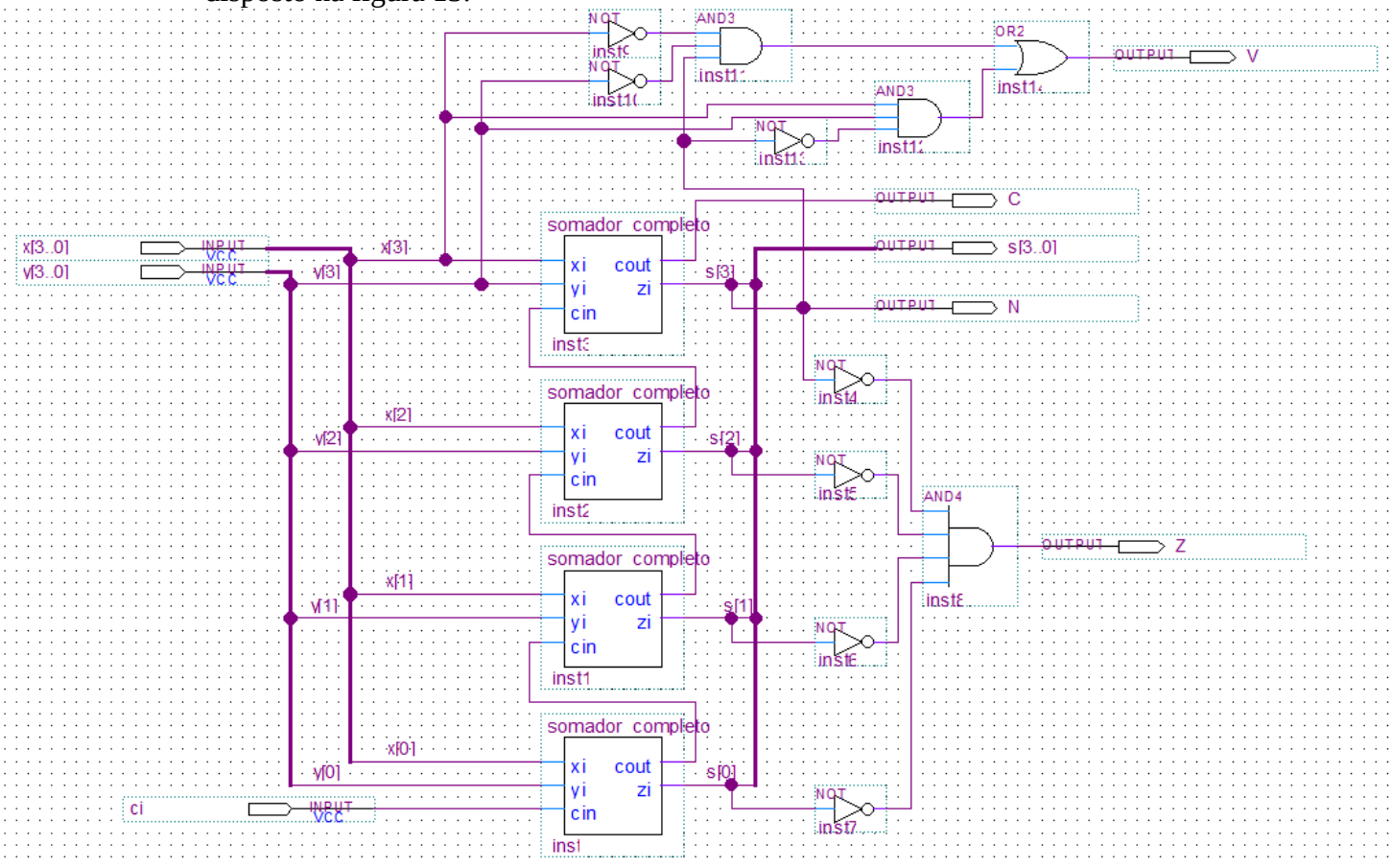

Figura 13: Diagrama esquemático do módulo somador_4_bits

3.7. Simulações:

Como o circuito deste módulo apresenta 9 entradas e, portanto, 2^9 possíveis combinações destas, não seria viável analisar todas as saídas geradas em tais casos. Portanto, como medida para contornar esse problema, uma tabela de valores de interesse foi montada e usada como entrada para a simulação (tabela 1). Os resultados estão expostos nas figuras 14 e 15.

Entradas			Saídas esperadas				
$\underline{x}$	$\underline{y}$	c_i	$\underline{s}$	C	V	N	Z
0100	0010	1	0111	0	0	0	0
0000	0000	0	0000	0	0	0	1
0011	1010	0	1101	0	0	1	0
0110	0010	1	1001	0	1	1	0
0011	1110	0	0001	1	0	0	0
0100	1011	1	0000	1	0	0	1
1100	1100	0	1000	1	0	1	0
1110	0000	0	1110	0	0	1	0
0001	0001	1	0011	0	0	0	0
0101	1000	0	1101	0	0	1	0
1001	0101	0	1110	0	0	1	0
0100	0001	1	0110	0	0	0	0
1001	1010	0	0011	1	1	0	0
1111	1111	1	1111	1	0	1	0
1000	0110	1	1111	0	0	1	0
0110	1010	1	0001	1	0	0	0

Tabela 1: combinações de interesse das entradas do somador e saídas esperadas para cada caso. As primeiras 8 combinações de entradas foram escolhidas de modo a contemplar todos os casos possíveis de ocorrências das flags. As demais foram tomadas aleatoriamente.

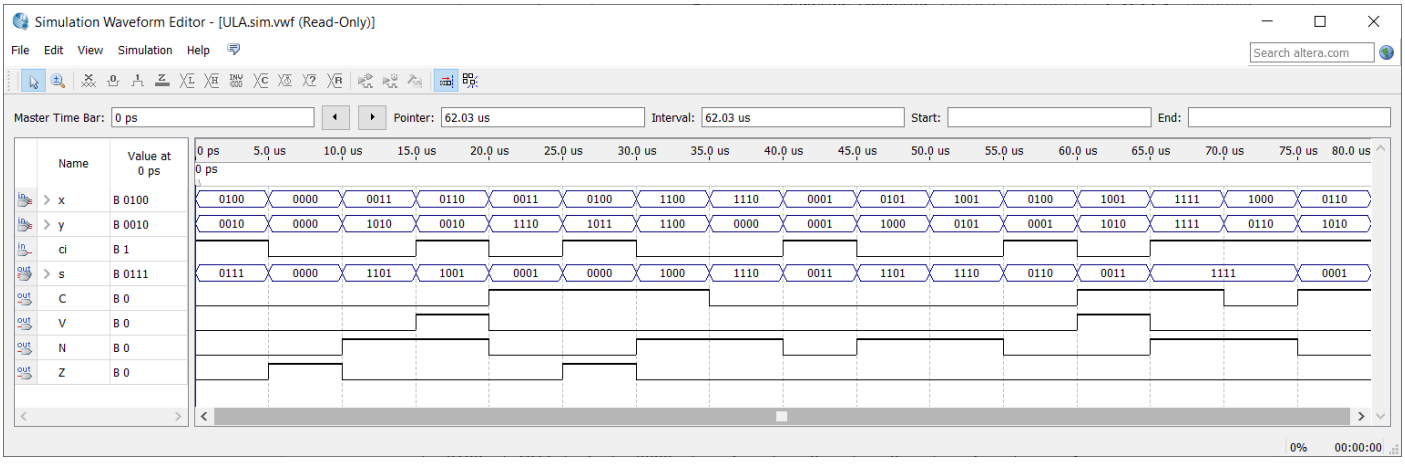

Figura 14: simulação funcional do módulo somador_4_bits

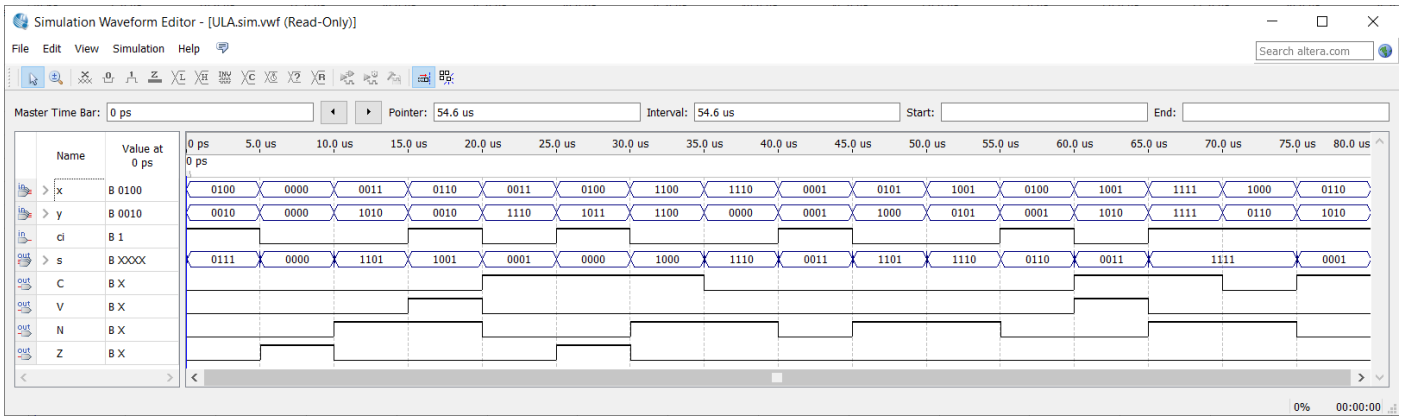

Figura 15: simulação temporal do módulo somador_4_bits

4. Projeto do módulo “decodificador_ULA”:

4.1. Escopo:

Projeto de um circuito combinacional capaz de atuar como decodificador da ULA construída. O circuito deve receber uma entrada seletora de 4 bits e um carry, enviando 4 sinais de controle como saída.

4.2. Especificação de alto-nível:

Entradas:

$\underline{f} = (f_3, f_2, f_1, f_0)$ com $f_k \in \{0,1\}$ e $k = 0,1,2,3$ (entrada seletora);

$c_i \in \{0,1\}$ (carry in);

Saídas:

$cmpx \in \{0,1\}$; $cmpy \in \{0,1\}$; $zry \in \{0,1\}$; $cci \in \{0,1\}$

Função:

Código ($f_3 f_2 f_1 f_0$)	Nome	$cmpx$	$cmpy$	zry	cci
000x	ADD	0	0	0	0
001x	ADDC	0	0	0	c_i
010x	SUB	0	1	0	1
0110	INC	0	0	1	1
0111	DEC	0	1	1	0
1000	NEG	1	0	1	1
1001	CMPL	1	0	1	0

4.3. Especificação binária:

Entradas:

$\underline{f} = (f_3, f_2, f_1, f_0)$ com $f_k \in \{0,1\}$ e $k = 0,1,2,3$ (entrada seletora);

$c_i \in \{0,1\}$ (carry in);

Saídas:

$cmpx \in \{0,1\}$; $cmpy \in \{0,1\}$; $zry \in \{0,1\}$; $cci \in \{0,1\}$

Função: Tabela verdade

f_3	f_2	f_1	f_0	$cmpx$	$cmpy$	zry	cci
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	c_i
0	0	1	1	0	0	0	c_i
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	1
0	1	1	0	0	0	1	1
0	1	1	1	0	1	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	0	1	0
1	0	1	0	x	x	x	x

1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

4.4. Minimizações:

$cmpx$:	$f_1'f_0'$	$f_1'f_0$	f_1f_0	f_1f_0'
$f_3'f_2'$	0	0	0	0
$f_3'f_2$	0	0	0	0
f_3f_2	x	x	x	x
f_3f_2'	1	1	x	x

$$cmpx = f_3$$

$cmpy$:	$f_1'f_0'$	$f_1'f_0$	f_1f_0	f_1f_0'
$f_3'f_2'$	0	0	0	0
$f_3'f_2$	1	1	1	0
f_3f_2	x	x	x	x
f_3f_2'	0	0	x	x

$$cmpy = f_2f_1' + f_2f_0 = f_2(f_1' + f_0)$$

zry:	$f_1'f_0'$	$f_1'f_0$	f_1f_0	f_1f_0'
$f_3'f_2'$	0	0	0	0
$f_3'f_2$	0	0	1	1
f_3f_2	x	x	x	x
f_3f_2'	1	1	x	x

$$zry = f_3 + f_2f_1$$

cci:	$f_1'f_0'$	$f_1'f_0$	f_1f_0	f_1f_0'
$f_3'f_2'$	0	0	c_i	c_i
$f_3'f_2$	1	1	0	1
f_3f_2	x	x	x	x
f_3f_2'	1	0	x	x

$$cci = f_3f_0' + f_2f_0' + f_2f_1' + c_if_3'f_2'f_1$$

4.5. Esquemático do circuito:

O circuito do módulo “decodificador_ULA” foi implementado utilizando as funções minimizadas acima. Seu esquemático está exposto na figura 16.

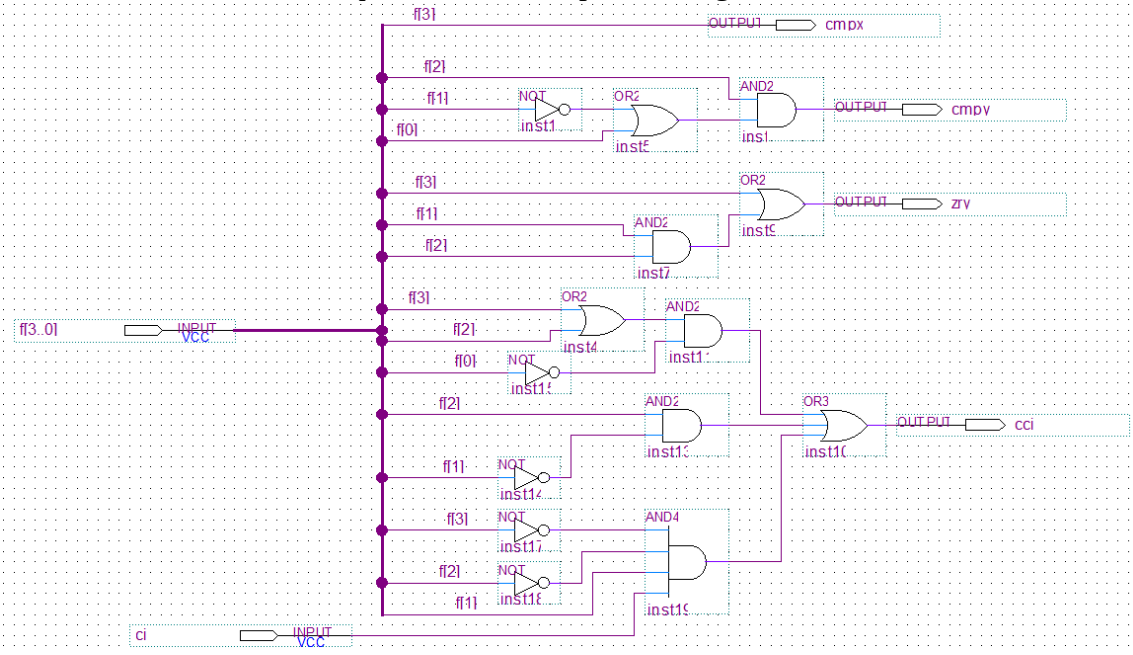

Figura 16: Diagrama esquemático do módulo somador_4_bits

4.6. Simulações:

Para as simulações, as entradas foram configuradas de modo a assumirem todas as configurações possíveis, até mesmo as consideradas inválidas para o circuito. A simulação funcional é apresentada na figura 17 e a simulação temporal, na figura 18. A figura 19 é uma simulação funcional adicional feita somente com as combinações das entradas de seleção válidas para o circuito.

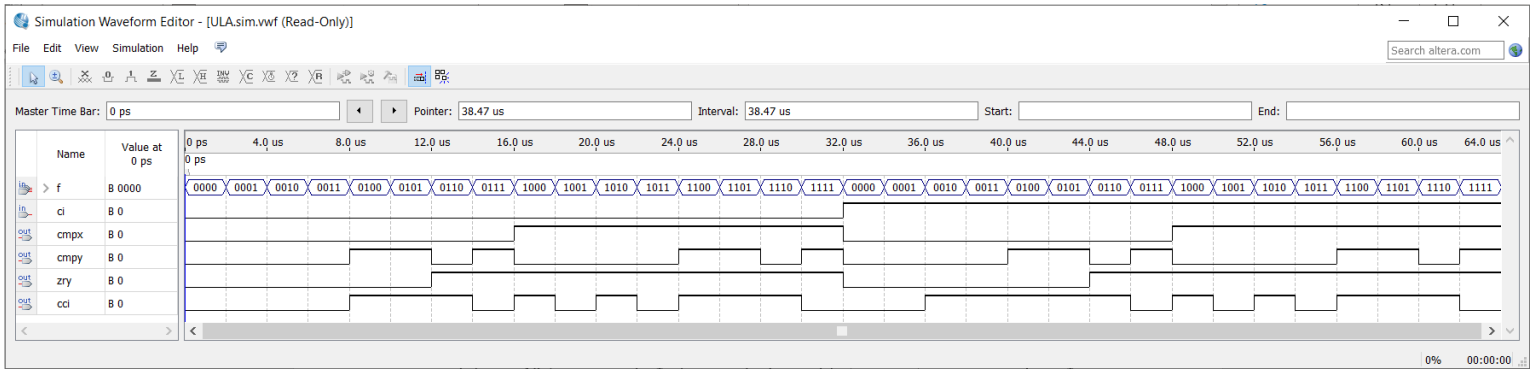

Figura 17: Simulação funcional do módulo "decodificador_ULA", considerando até combinações de entradas inválidas

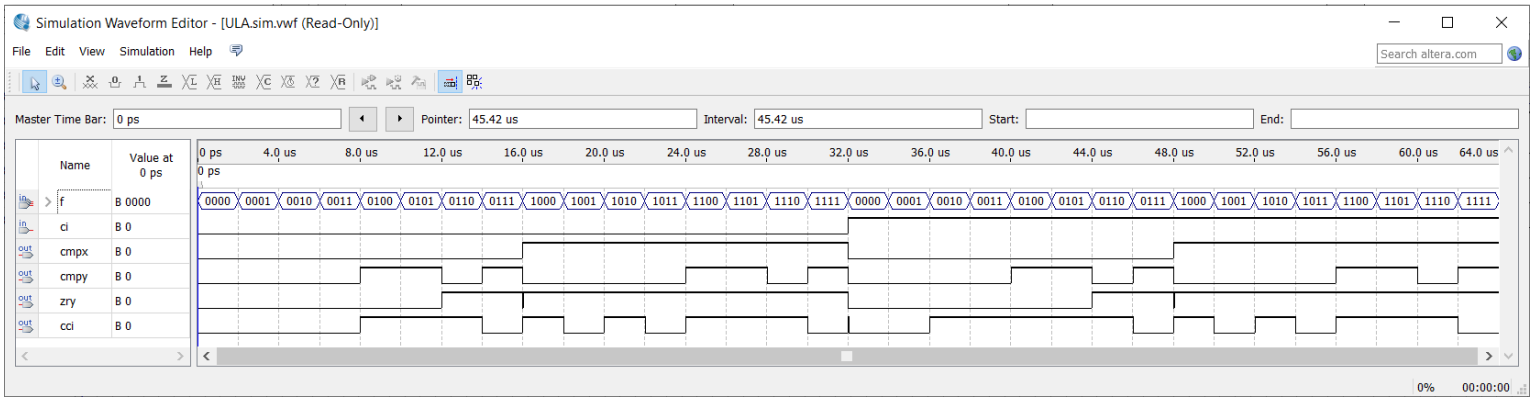

Figura 18: Simulação temporal do módulo "decodificador_ULA", considerando até combinações de entradas inválidas

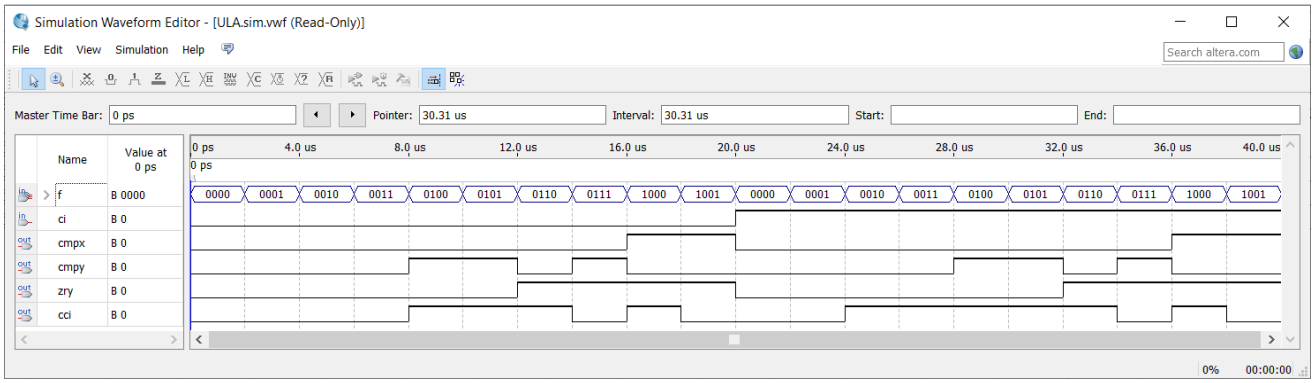

Figura 19: Simulação funcional do módulo "decodificador_ULA", considerando somente combinações de entradas válidas

5. Projeto do circuito ULA:

5.1. Escopo:

O circuito ULA é o projeto final deste relatório, pelo qual todos os demais módulos foram construídos. Esse circuito combinacional empregará instâncias de cada símbolo definido até então, devendo ser capaz de, a partir de uma entrada seletora de

4 bits, realizar uma determinada operação sobre duas entradas de 4 bits de operandos e uma entrada de 1 bit de carry.

5.2. Especificação de alto-nível:

Entradas:

$\underline{x} = (x_3, x_2, x_1, x_0)$ com $x_k \in \{0,1\}$ e $k = 0,1,2,3$;
$\underline{y} = (y_3, y_2, y_1, y_0)$ com $y_k \in \{0,1\}$ e $k = 0,1,2,3$;
$c_i \in \{0,1\}$ (carry in);

Saídas:

$\underline{s} = (s_3, s_2, s_1, s_0)$ com $s_k \in \{0,1\}$ e $k = 0,1,2,3$;
$C \in \{0,1\}$; $V \in \{0,1\}$; $N \in \{0,1\}$; $Z \in \{0,1\}$

Função:

Código ($f_3 f_2 f_1 f_0$)	Nome	Operação	Descrição
000x	ADD	$\underline{s} = \underline{x} + \underline{y}$	Adição de x com y
001x	ADDC	$\underline{s} = \underline{x} + \underline{y} + c_i$	Adição de x com y com carry
010x	SUB	$\underline{s} = \underline{x} - \underline{y}$	Subtração y de x
0110	INC	$\underline{s} = \underline{x} + 1$	Incremente unitário de x
0111	DEC	$\underline{s} = \underline{x} - 1$	Decremento unitário de x
1000	NEG	$\underline{s} = -\underline{x}$	Negação aritmética de x em complemento de 2
1001	CMPL	$\underline{s} = \sim\underline{x}$	Complemento bit-a-bit de x

5.3. Especificação binária e Minimizações:

Entradas:

$\underline{x} = (x_3, x_2, x_1, x_0)$ com $x_k \in \{0,1\}$ e $k = 0,1,2,3$;
$\underline{y} = (y_3, y_2, y_1, y_0)$ com $y_k \in \{0,1\}$ e $k = 0,1,2,3$;
$c_i \in \{0,1\}$ (carry in);

Saídas:

$\underline{s} = (s_3, s_2, s_1, s_0)$ com $s_k \in \{0,1\}$ e $k = 0,1,2,3$;
$C \in \{0,1\}$; $V \in \{0,1\}$; $N \in \{0,1\}$; $Z \in \{0,1\}$

Função:

Para implementar as saídas desejadas do circuito ULA, serão empregues os módulos “complementa”, “zera”, “somador_4_bits” e “decodificador_ULA”. Portanto, como medida de facilitar o entendimento de cada função, vamos definir primeiramente que:

- $\mathbb{c}_{[\underline{o}]}(\underline{i}, cmp)$ representa o vetor de 4 bits de saída do módulo “complementa”, tendo como entrada o vetor de 4 bits i e o sinal de controle cmp;
- $\mathbb{z}_{[\underline{o}]}(\underline{i}, zera)$ representa o vetor de 4 bits de saída do módulo “zera”, tendo como entrada o vetor de 4 bits i e o sinal de controle cmp;
- $\mathbb{S}_{[\underline{s}]}(x, y, c_i)$ representa o vetor de 4 bits de saída do módulo “somador_4_bits”, tendo como entrada os vetores de 4 bits x e y, bem como o bit de carry c_i;

- $\mathbb{S}_{[C]}(x, y, c_i)$, $\mathbb{S}_{[V]}(x, y, c_i)$, $\mathbb{S}_{[N]}(x, y, c_i)$ ou $\mathbb{S}_{[z]}(x, y, c_i)$ representam as flags de saída do módulo "somador_4_bits", tendo como entrada os vetores de 4 bits x e y, bem como o bit de carry c_i;
- $\mathbb{d}_{[cmpx]}(f, c_i)$ representa a saída $cmpx$ do módulo "decodificador_ULA", tendo como entrada o vetor de 4 bits f e o bit de carry c_i;
- $\mathbb{d}_{[cmpy]}(f, c_i)$ representa a saída $cmpy$ do módulo "decodificador_ULA", tendo como entrada o vetor de 4 bits f e o bit de carry c_i;
- $\mathbb{d}_{[zry]}(f, c_i)$ representa a saída zry do módulo "decodificador_ULA", tendo como entrada o vetor de 4 bits f e o bit de carry c_i;
- $\mathbb{d}_{[cci]}(f, c_i)$ representa a saída cci do módulo "decodificador_ULA", tendo como entrada o vetor de 4 bits f e o bit de carry c_i.

Dessa forma, as funções de cada circuito do projeto ULA podem ser descritos da seguinte maneira:

$$\underline{s} = \mathbb{S}_{[\underline{s}]}\left(\mathbb{c}_{[\underline{o}]}\left(\underline{x}, \mathbb{d}_{[cmpx]}(f, c_i)\right), \mathbb{c}_{[\underline{o}]}\left(\mathbb{z}_{[\underline{o}]}\left(\underline{y}, \mathbb{d}_{[zry]}(f, c_i)\right), \mathbb{d}_{[cmpy]}(f, c_i)\right), \mathbb{d}_{[cci]}(f, c_i)\right)$$

$$C = \mathbb{S}_{[C]}\left(\mathbb{c}_{[\underline{o}]}\left(\underline{x}, \mathbb{d}_{[cmpx]}(f, c_i)\right), \mathbb{c}_{[\underline{o}]}\left(\mathbb{z}_{[\underline{o}]}\left(\underline{y}, \mathbb{d}_{[zry]}(f, c_i)\right), \mathbb{d}_{[cmpy]}(f, c_i)\right), \mathbb{d}_{[cci]}(f, c_i)\right)$$

$$V = \mathbb{S}_{[V]}\left(\mathbb{c}_{[\underline{o}]}\left(\underline{x}, \mathbb{d}_{[cmpx]}(f, c_i)\right), \mathbb{c}_{[\underline{o}]}\left(\mathbb{z}_{[\underline{o}]}\left(\underline{y}, \mathbb{d}_{[zry]}(f, c_i)\right), \mathbb{d}_{[cmpy]}(f, c_i)\right), \mathbb{d}_{[cci]}(f, c_i)\right)$$

$$N = \mathbb{S}_{[N]}\left(\mathbb{c}_{[\underline{o}]}\left(\underline{x}, \mathbb{d}_{[cmpx]}(f, c_i)\right), \mathbb{c}_{[\underline{o}]}\left(\mathbb{z}_{[\underline{o}]}\left(\underline{y}, \mathbb{d}_{[zry]}(f, c_i)\right), \mathbb{d}_{[cmpy]}(f, c_i)\right), \mathbb{d}_{[cci]}(f, c_i)\right)$$

$$Z = \mathbb{S}_{[Z]}\left(\mathbb{c}_{[\underline{o}]}\left(\underline{x}, \mathbb{d}_{[cmpx]}(f, c_i)\right), \mathbb{c}_{[\underline{o}]}\left(\mathbb{z}_{[\underline{o}]}\left(\underline{y}, \mathbb{d}_{[zry]}(f, c_i)\right), \mathbb{d}_{[cmpy]}(f, c_i)\right), \mathbb{d}_{[cci]}(f, c_i)\right)$$

5.4. Esquemático do circuito:

O circuito foi implementado seguindo as funções descritas acima, o resultado está disposto na figura 20.

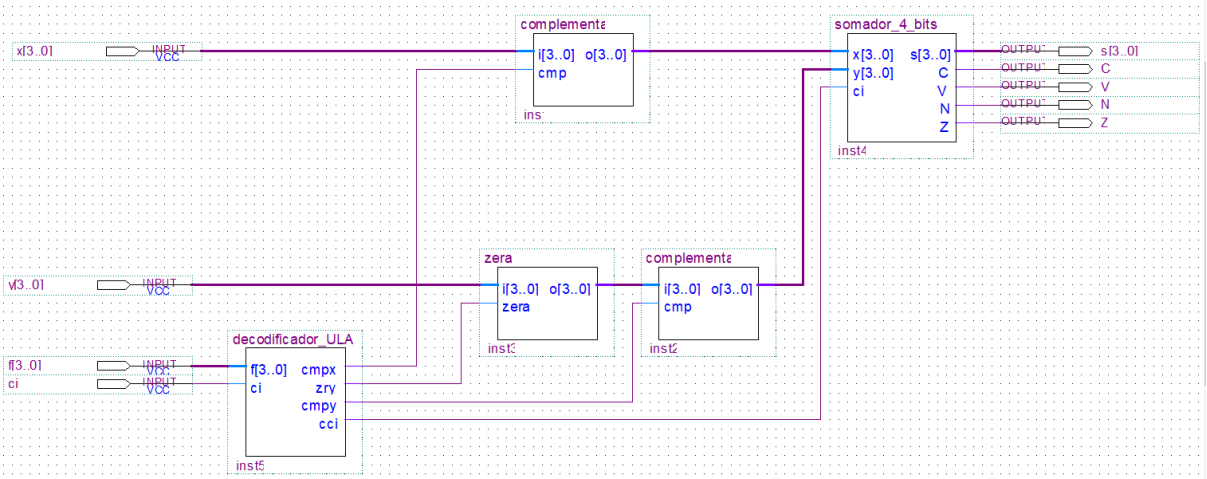

Figura 20: Diagrama esquemático do circuito ULA

5.5. Simulações:

Como o circuito recebe 13 entradas e, portanto, apresenta 2^{13} possíveis combinações destas, não seria viável analisar todas as saídas geradas em tais casos. Portanto, cada funcionalidade implementada foi testada separadamente com um conjunto de valores de entrada de interesse. Os conjuntos de valores testados e resultados esperados estão dispostos nas tabelas 2 a 8. Os resultados das simulações estão expostos nas figuras 21 a 36.

Teste da função ADD							
Entradas			Saídas esperadas				
$\underline{x}$	$\underline{y}$	c_i	$\underline{s}$	C	V	N	Z
0100	0010	x	0110	0	0	0	0
0000	0000	x	0000	0	0	0	1
0011	1010	x	1101	0	0	1	0
0110	0010	x	1000	0	1	1	0
0011	1110	x	0001	1	0	0	0
0100	1011	x	1111	0	0	1	0
1100	1100	x	1000	1	0	1	0
1110	0000	x	1110	0	0	1	0
0001	0001	x	0010	0	0	0	0
0101	1000	x	1101	0	0	1	0
1001	0101	x	1110	0	0	1	0
0100	0001	x	0101	0	0	0	0
1001	1010	x	0011	1	1	0	0
1111	1111	x	1110	1	0	1	0
1000	0110	x	1110	0	0	1	0
0110	1010	x	0000	1	0	0	1

Tabela 2: combinações de interesse das entradas da ULA para o teste da funcionalidade ADD (entrada $\underline{f} = 000x$).

Teste da função ADDC							
Entradas			Saídas esperadas				
$\underline{x}$	$\underline{y}$	c_i	$\underline{s}$	C	V	N	Z
0100	0010	1	0111	0	0	0	0
0000	0000	0	0000	0	0	0	1
0011	1010	0	1101	0	0	1	0
0110	0010	1	1001	0	1	1	0
0011	1110	0	0001	1	0	0	0
0100	1011	1	0000	1	0	0	1
1100	1100	0	1000	1	0	1	0
1110	0000	0	1110	0	0	1	0
0001	0001	1	0011	0	0	0	0
0101	1000	0	1101	0	0	1	0
1001	0101	0	1110	0	0	1	0
0100	0001	1	0110	0	0	0	0
1001	1010	0	0011	1	1	0	0
1111	1111	1	1111	1	0	1	0
1000	0110	1	1111	0	0	1	0
0110	1010	1	0001	1	0	0	0

Tabela 3: combinações de interesse das entradas da ULA para o teste da funcionalidade ADDC (entrada $\underline{f} = 001x$).

Teste da função SUB							
Entradas			Saídas esperadas				
$\underline{x}$	$\underline{y}$	c_i	$\underline{s}$	C	V	N	Z
0100	0010	x	0010	1	0	0	0
0000	0000	x	0000	1	0	0	1
0011	1010	x	1001	0	1	1	0
0110	0010	x	0100	1	0	0	0
0011	1110	x	0101	0	0	0	0
0100	1011	x	1001	0	1	1	0
1100	1100	x	0000	1	0	0	1
1110	0000	x	1110	1	0	1	0
0001	0001	x	0000	1	0	0	1
0101	1000	x	1101	0	1	1	0
1001	0101	x	0100	1	1	0	0
0100	0001	x	0011	1	0	0	0
1001	1010	x	1111	0	0	1	0
1111	1111	x	0000	1	0	0	1
1000	0110	x	0010	1	1	0	0
0110	1010	x	1100	0	1	1	0

Tabela 4: combinações de interesse das entradas da ULA para o teste da funcionalidade SUB (entrada $\underline{f} = 010x$).

Teste da função INC							
Entradas			Saídas esperadas				
$\underline{x}$	$\underline{y}$	c_i	$\underline{s}$	C	V	N	Z
0000	x	x	0001	0	0	0	0
0001	x	x	0010	0	0	0	0
0010	x	x	0011	0	0	0	0
0011	x	x	0100	0	0	0	0
0100	x	x	0101	0	0	0	0
0101	x	x	0110	0	0	0	0
0110	x	x	0111	0	0	0	0
0111	x	x	1000	0	1	0	0
1000	x	x	1001	0	0	1	0
1001	x	x	1010	0	0	1	0
1010	x	x	1011	0	0	1	0
1011	x	x	1100	0	0	1	0
1100	x	x	1101	0	0	1	0
1101	x	x	1110	0	0	1	0
1110	x	x	1111	0	0	1	0
1111	x	x	0000	1	0	0	1

Tabela 5: combinações de interesse das entradas da ULA para o teste da funcionalidade SUB (entrada $\underline{f} = 0110$).

Teste da função DEC							
Entradas			Saídas esperadas				
$\underline{x}$	$\underline{y}$	c_i	$\underline{s}$	C	V	N	Z
0000	x	x	1111	0	0	1	0
0001	x	x	0000	1	0	0	1
0010	x	x	0001	1	0	0	0
0011	x	x	0010	1	0	0	0
0100	x	x	0011	1	0	0	0

Tabela 6: combinações de interesse das entradas da ULA para o teste da funcionalidade

0101	x	x	0100	1	0	0	0
0110	x	x	0101	1	0	0	0
0111	x	x	0110	1	0	0	0
1000	x	x	0111	1	1	0	0
1001	x	x	1000	1	0	1	0
1010	x	x	1001	1	0	1	0
1011	x	x	1010	1	0	1	0
1100	x	x	1011	1	0	1	0
1101	x	x	1100	1	0	1	0
1110	x	x	1101	1	0	1	0
1111	x	x	1110	1	0	1	0

SUB (entrada $\underline{f} = 0111$).

Teste da função NEG							
Entradas			Saídas esperadas				
$\underline{x}$	$\underline{y}$	c_i	$\underline{s}$	C	V	N	Z
0000	x	x	0000	1	0	0	1
0001	x	x	1111	0	0	1	0
0010	x	x	1110	0	0	1	0
0011	x	x	1101	0	0	1	0
0100	x	x	1100	0	0	1	0
0101	x	x	1011	0	0	1	0
0110	x	x	1010	0	0	1	0
0111	x	x	1001	0	0	1	0
1000	x	x	1000	0	1	1	0
1001	x	x	0111	0	0	0	0
1010	x	x	0110	0	0	0	0
1011	x	x	0101	0	0	0	0
1100	x	x	0100	0	0	0	0
1101	x	x	0011	0	0	0	0
1110	x	x	0010	0	0	0	0
1111	x	x	0001	0	0	0	0

Tabela 7: combinações de interesse das entradas da ULA para o teste da funcionalidade SUB (entrada $\underline{f} = 1000$).

Teste da função CMPL							
Entradas			Saídas esperadas				
$\underline{x}$	$\underline{y}$	c_i	$\underline{s}$	C	V	N	Z
0000	x	x	1111	0	0	1	0
0001	x	x	1110	0	0	1	0
0010	x	x	1101	0	0	1	0
0011	x	x	1100	0	0	1	0
0100	x	x	1011	0	0	1	0
0101	x	x	1010	0	0	1	0
0110	x	x	1001	0	0	1	0
0111	x	x	1000	0	0	1	0
1000	x	x	0111	0	0	0	0
1001	x	x	0110	0	0	0	0
1010	x	x	0101	0	0	0	0
1011	x	x	0100	0	0	0	0
1100	x	x	0011	0	0	0	0
1101	x	x	0010	0	0	0	0

Tabela 8: combinações de interesse das entradas da ULA para o teste da funcionalidade CMPL (entrada $\underline{f} = 1001$).

1110	x	x	0001	0	0	0	0
1111	x	x	0000	0	0	0	1

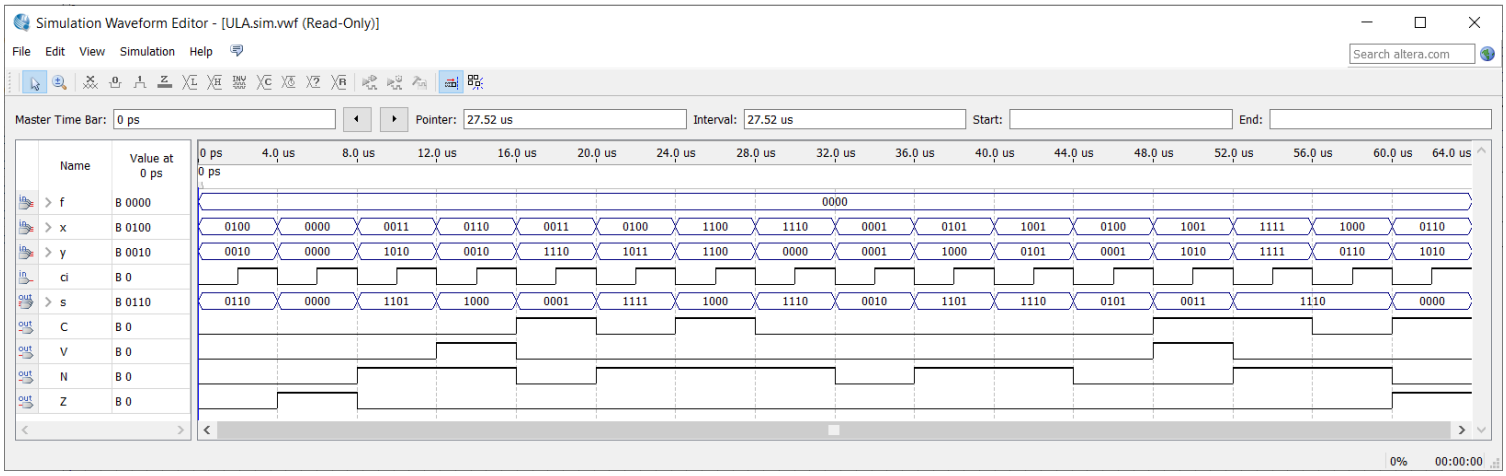

Figura 21: Simulação funcional da função ADD (com a entrada seletora sendo 0000)

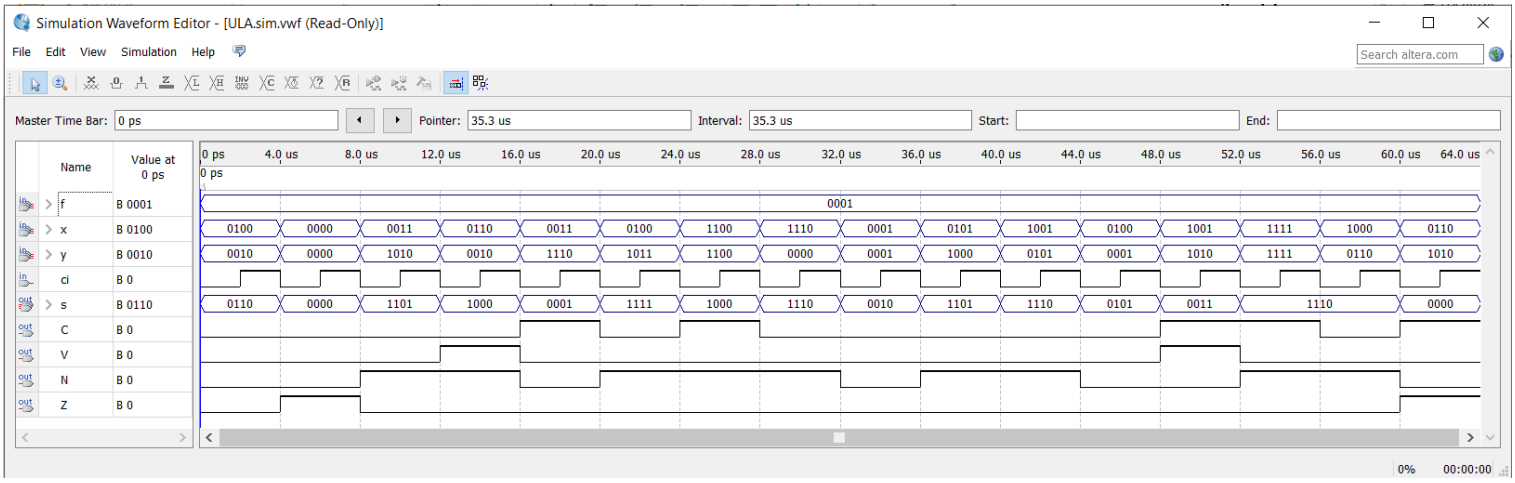

Figura 22: Simulação funcional da função ADD (com a entrada seletora sendo 0001)

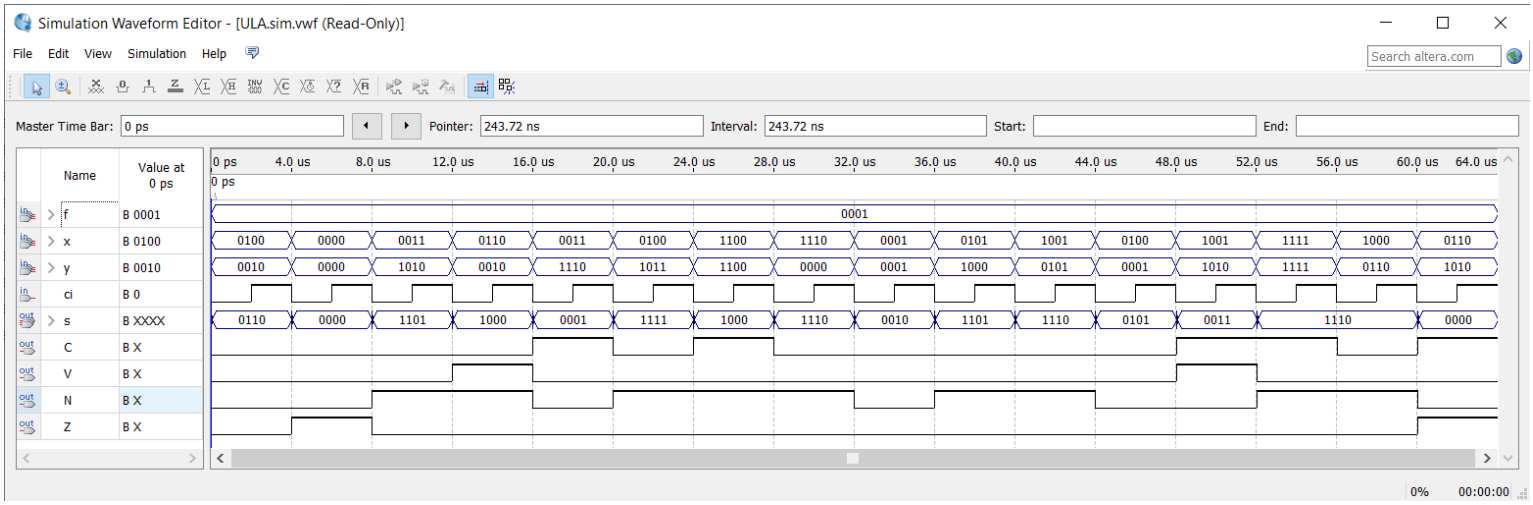

Figura 23: Simulação temporal da função ADD (com a entrada seletora sendo 0001)

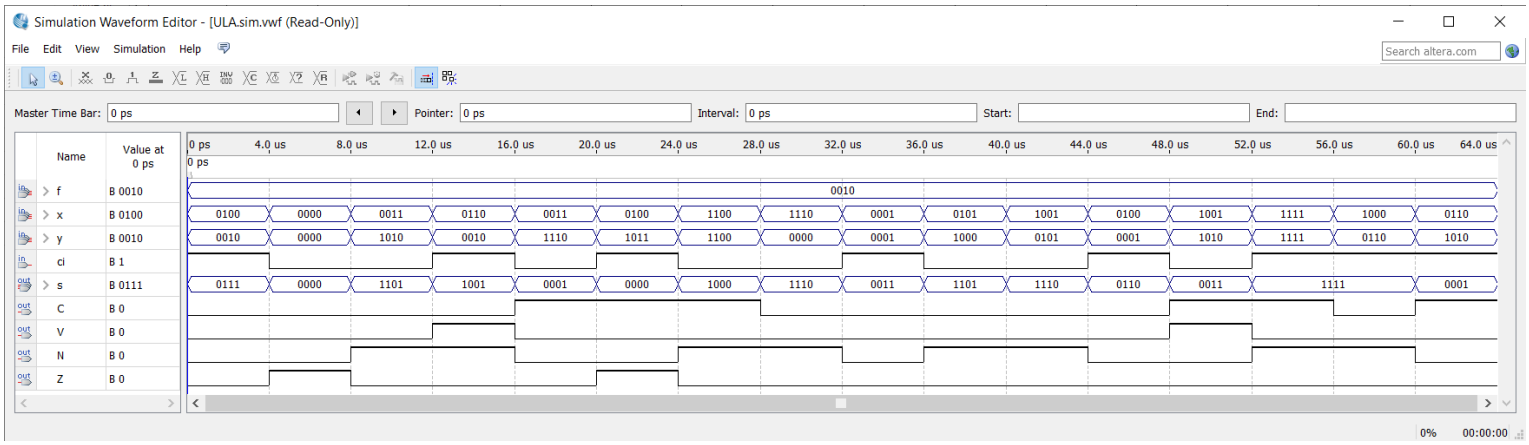

Figura 24: Simulação funcional da função ADDC (com a entrada seletora sendo 0010)

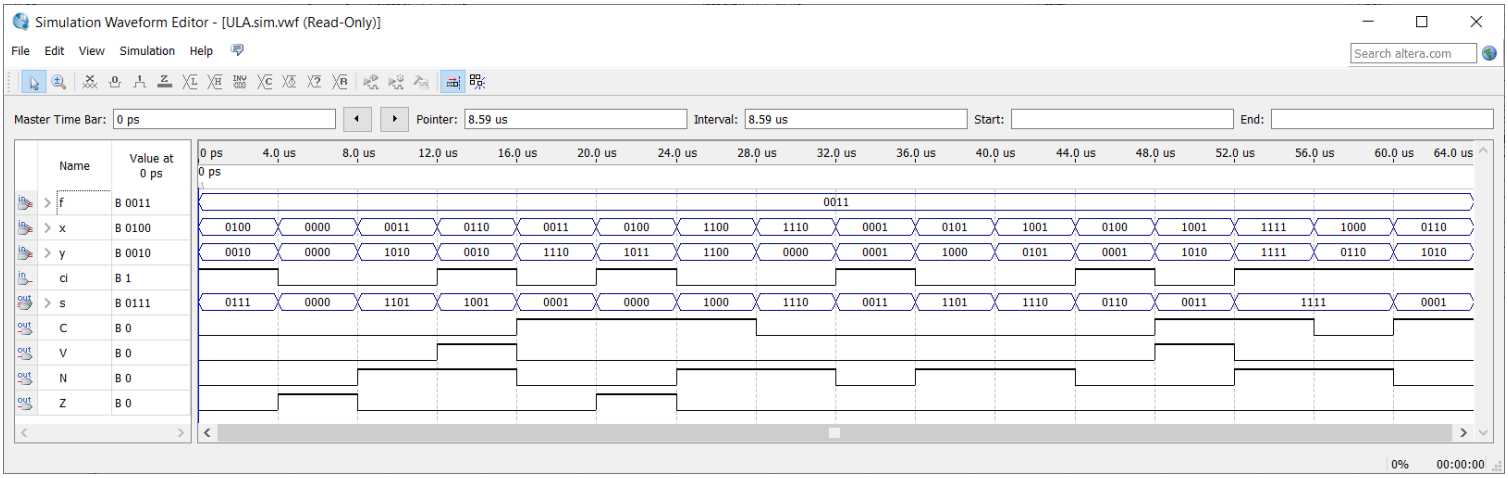

Figura 25: Simulação funcional da função ADDC (com a entrada seletora sendo 0011)

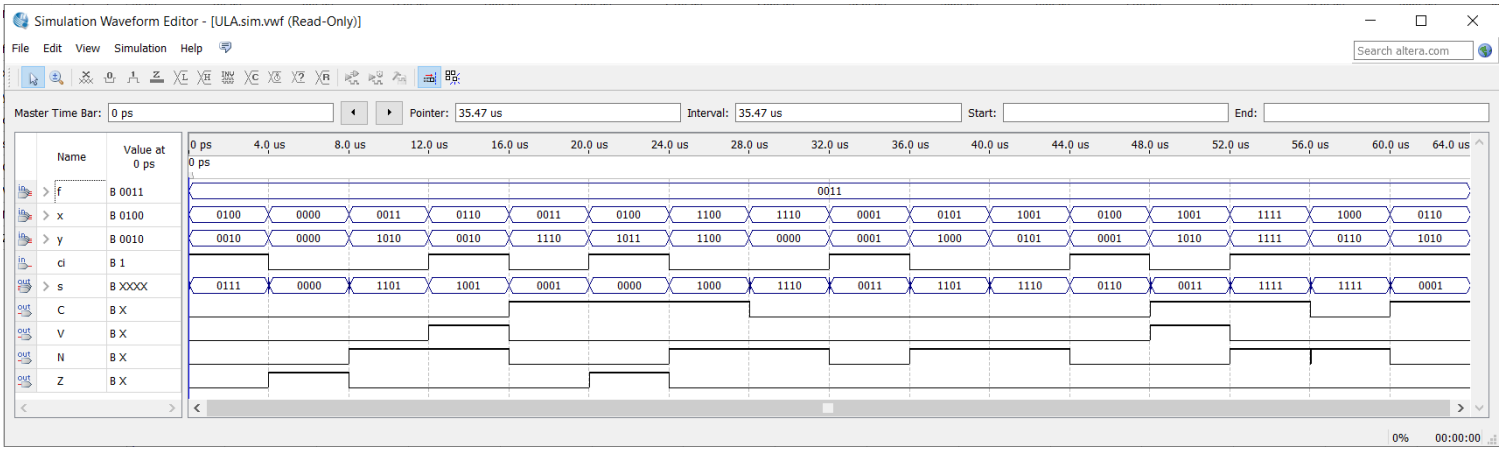

Figura 26: Simulação temporal da função ADDC (com a entrada seletora sendo 0011)

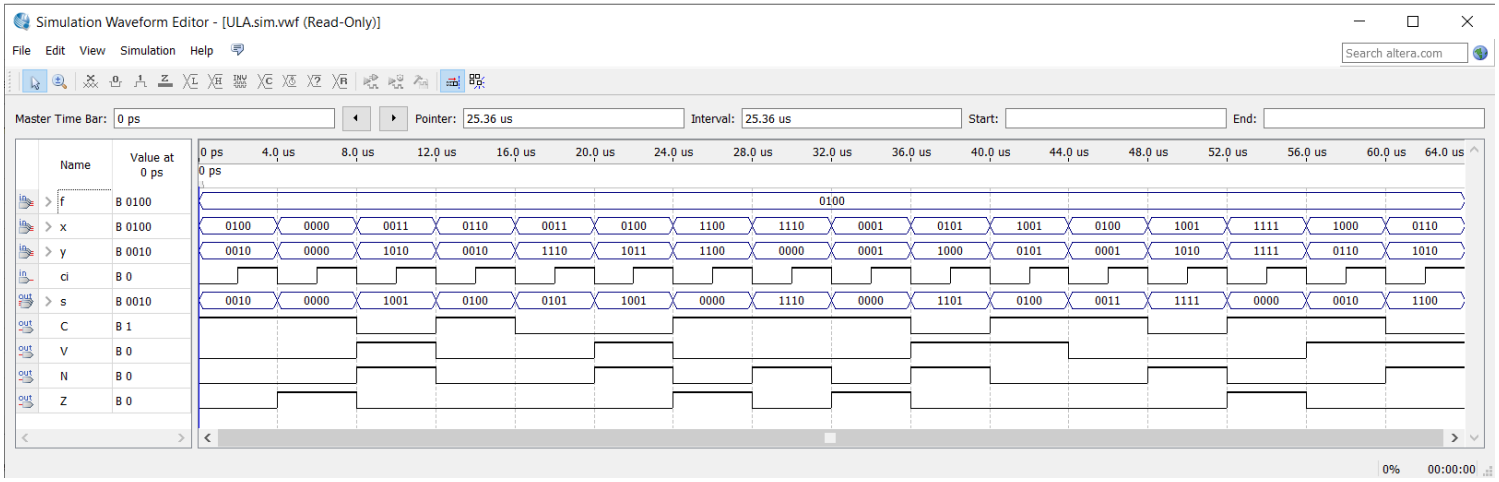

Figura 27: Simulação funcional da função SUB (com a entrada seletora sendo 0100)

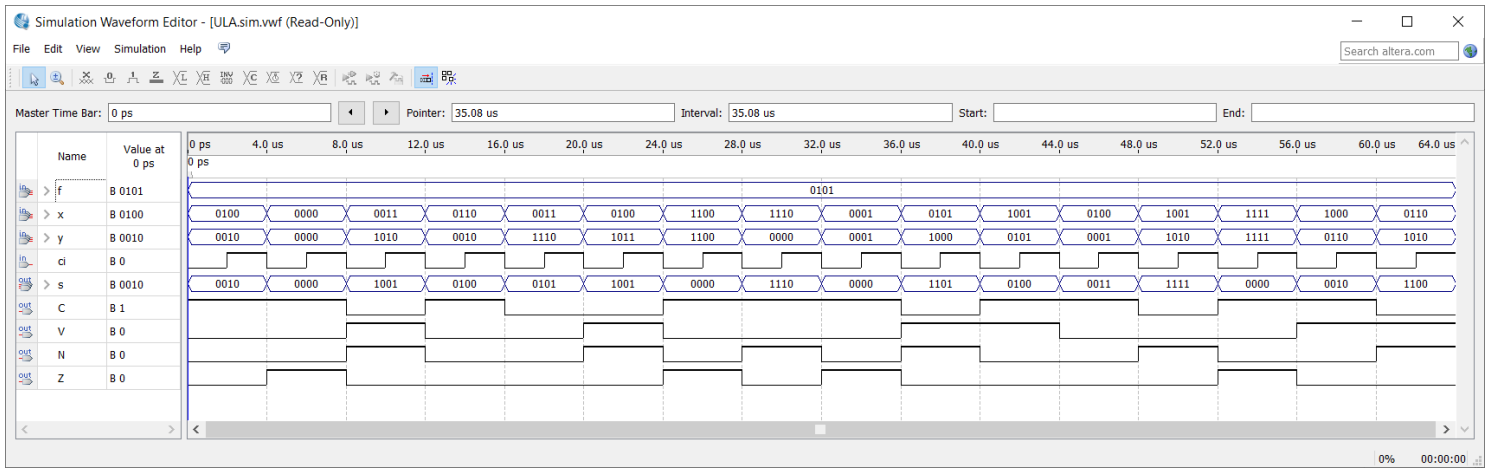

Figura 28: Simulação funcional da função SUB (com a entrada seletora sendo 0101)

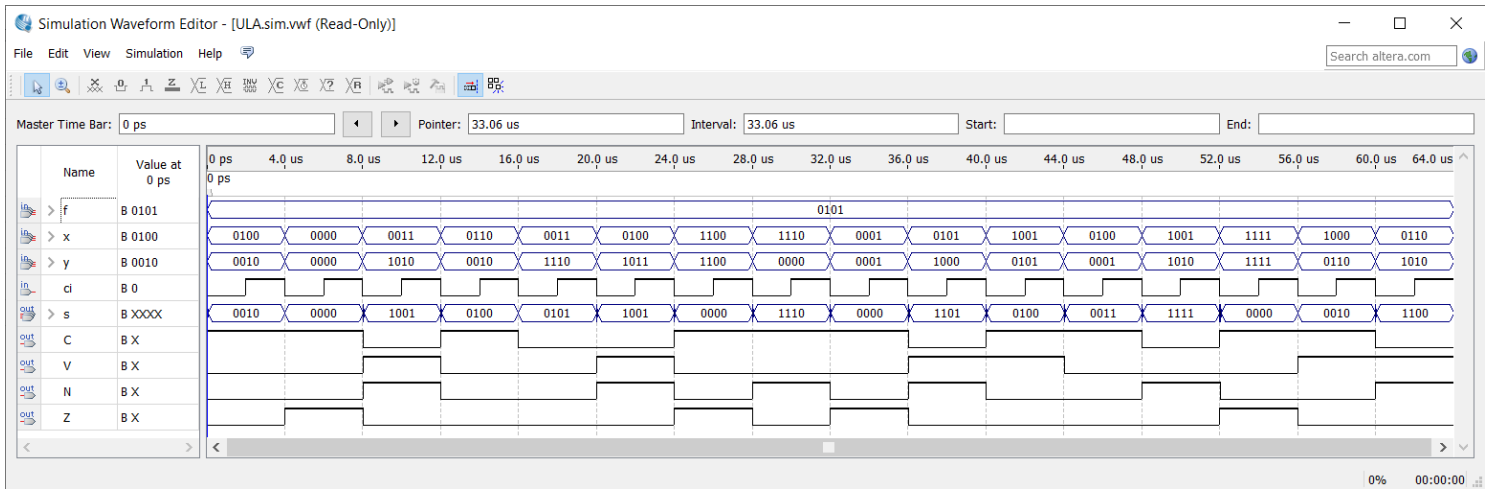

Figura 29: Simulação temporal da função SUB (com a entrada seletora sendo 0101)

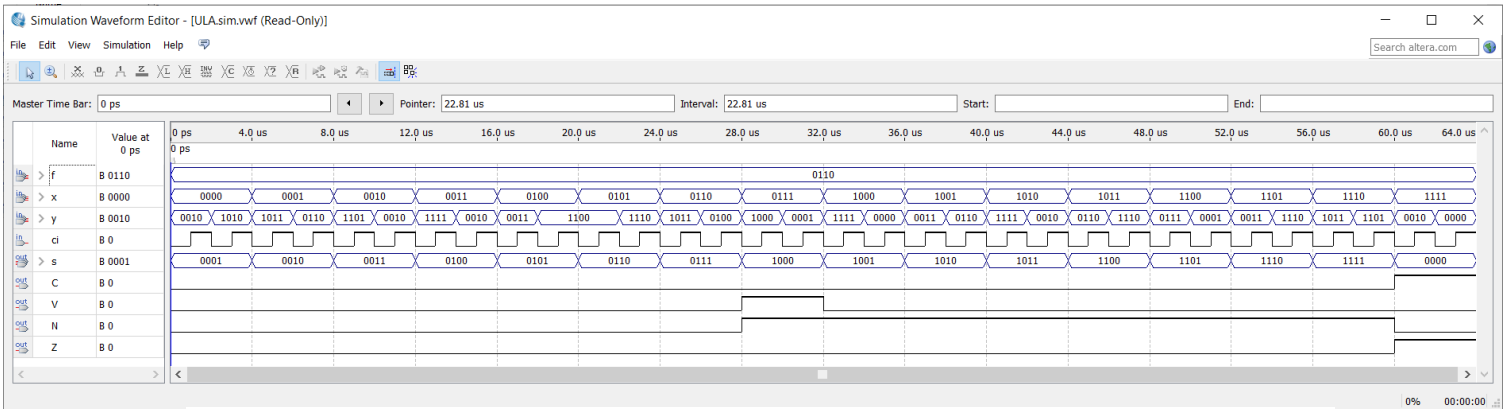

Figura 30: Simulação funcional da função INC

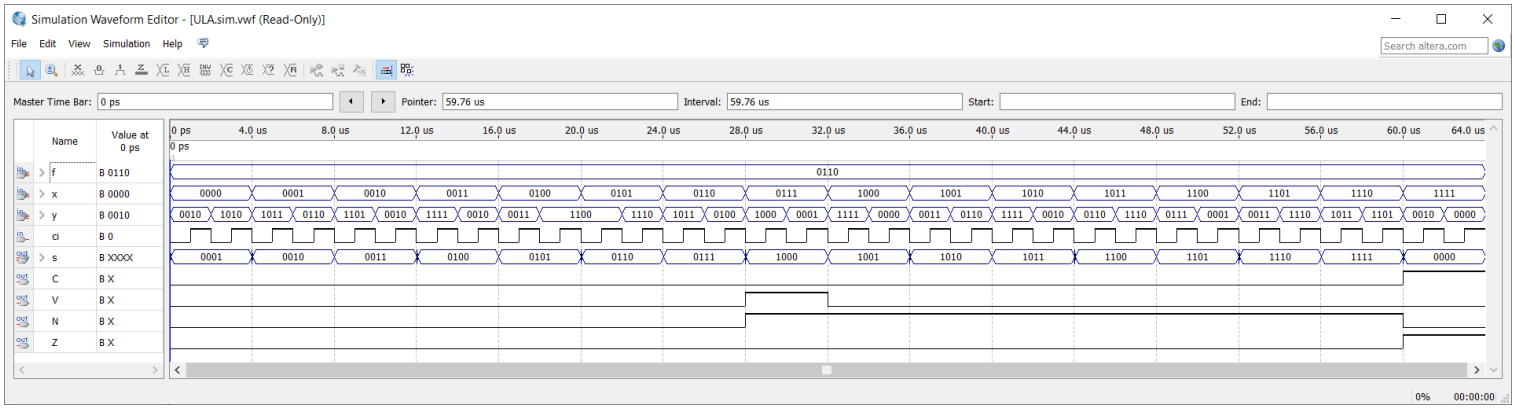

Figura 31: Simulação temporal da função INC

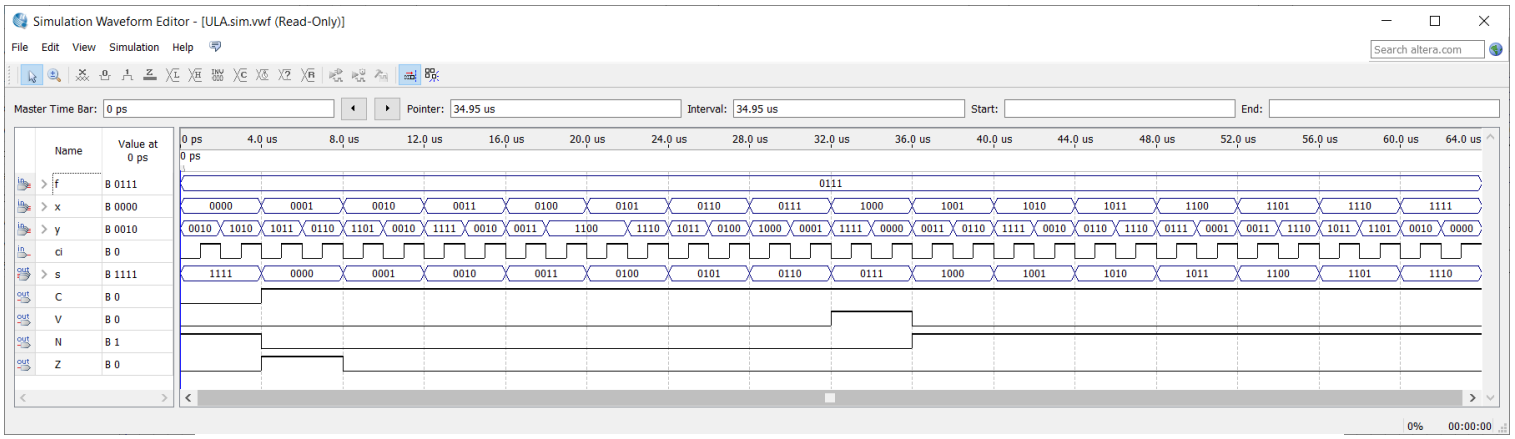

Figura 32: Simulação funcional da função DEC

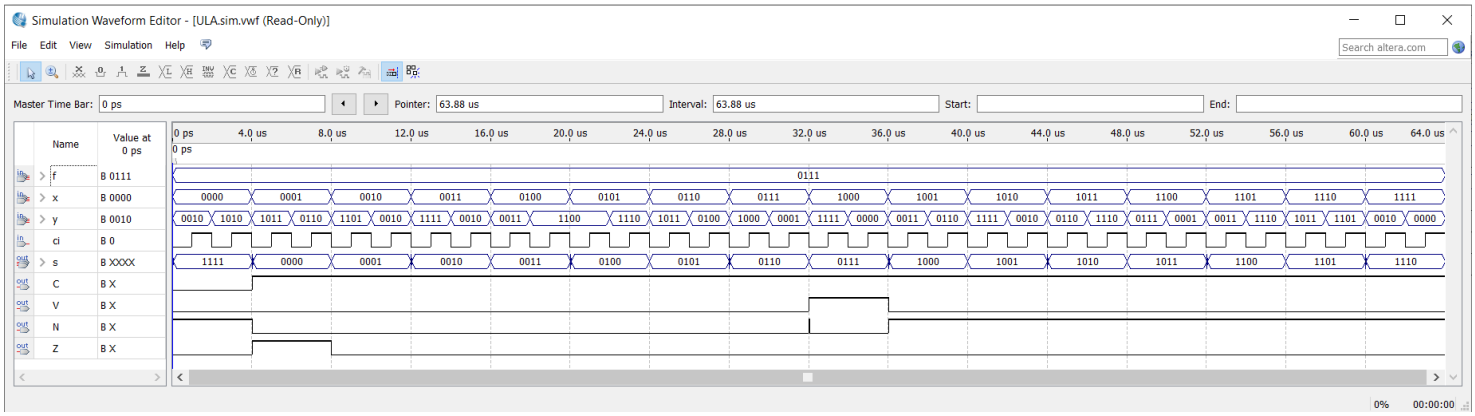

Figura 33: Simulação temporal da função DEC

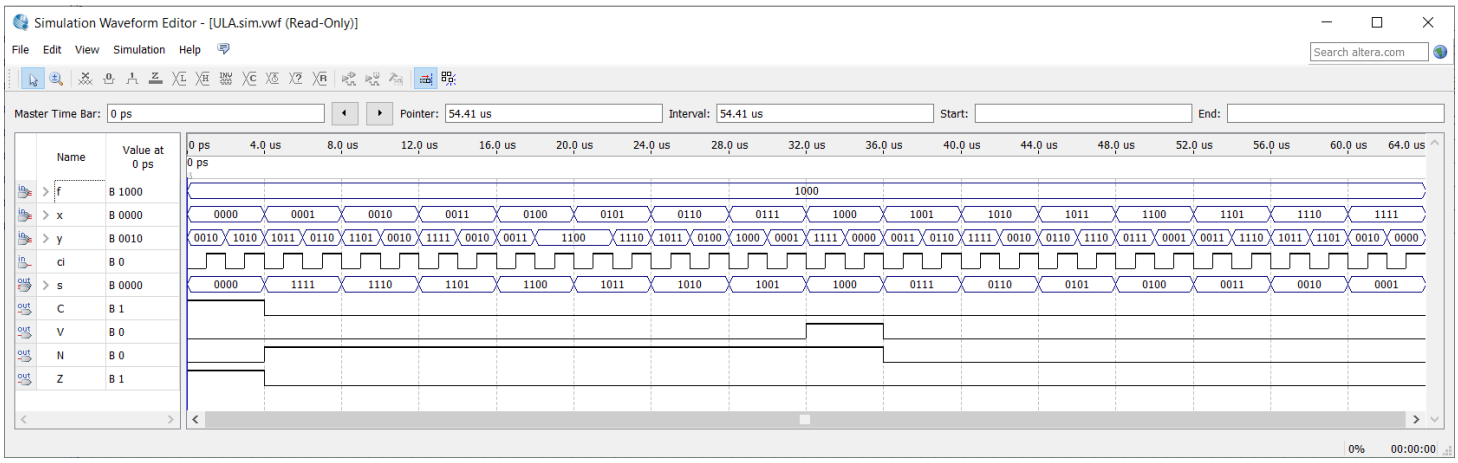

Figura 34: Simulação funcional da função NEG

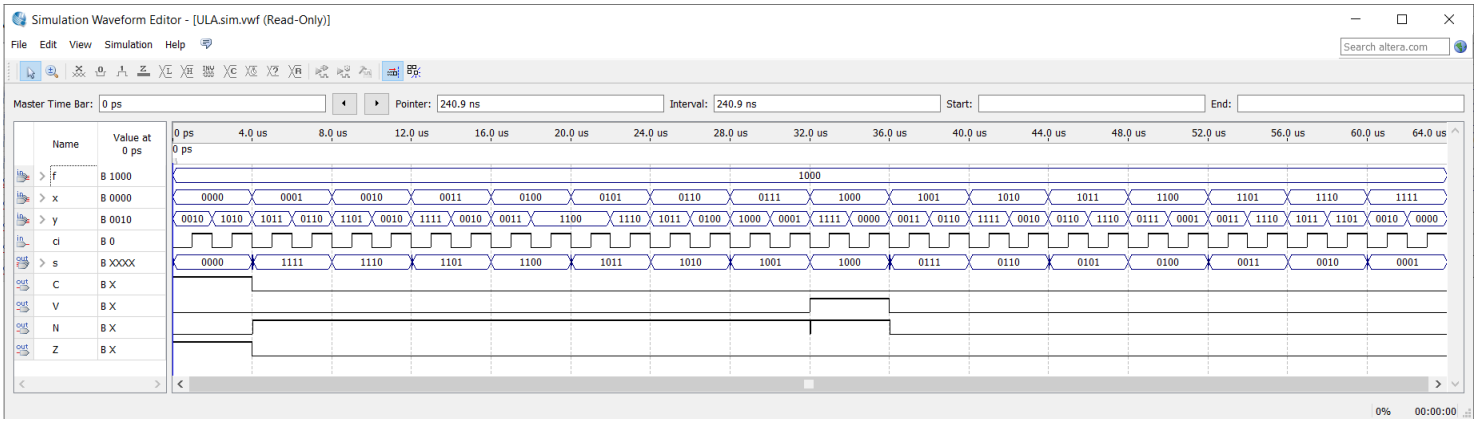

Figura 35: Simulação temporal da função NEG

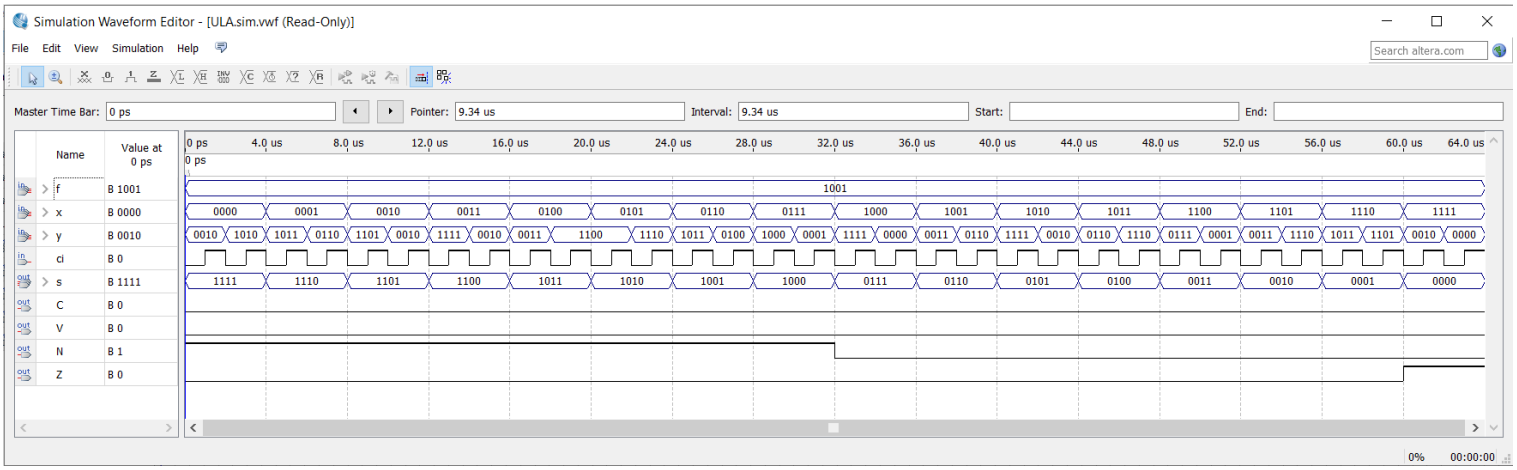

Figura 36: Simulação funcional da função CMPL

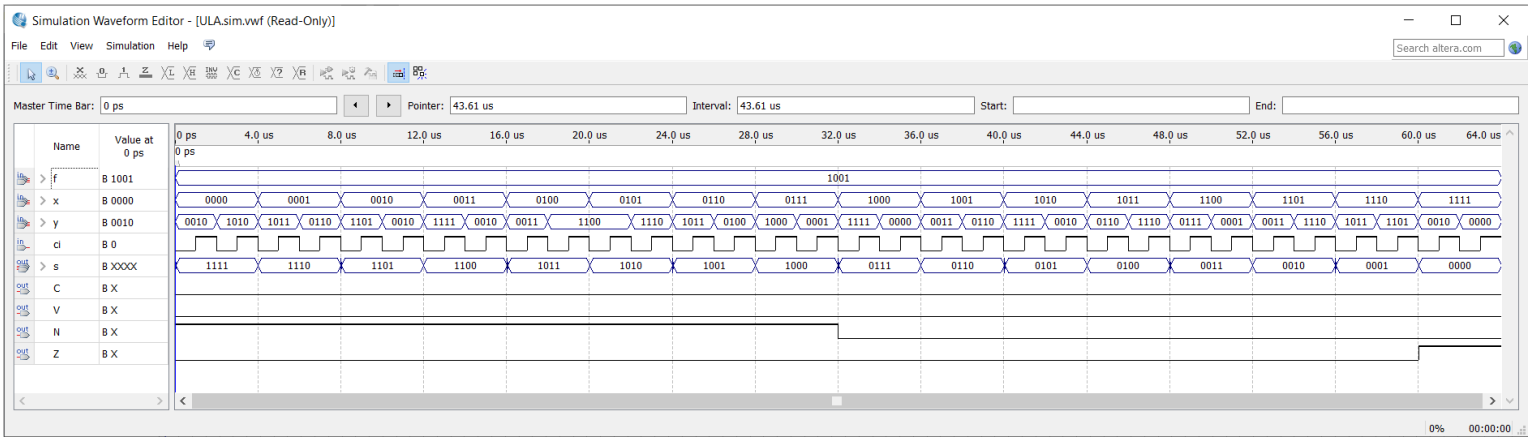

Figura 37: Simulação temporal da função CMPL

5.6. Encapsulamento do módulo ULA e testes finais:

Assim como pedido no roteiro desta atividade, o circuito resultante do projeto ULA foi encapsulado em um módulo e teve suas entradas e saídas associadas aos pinos respectivos. O circuito resultante está disposto no diagrama da figura 38.

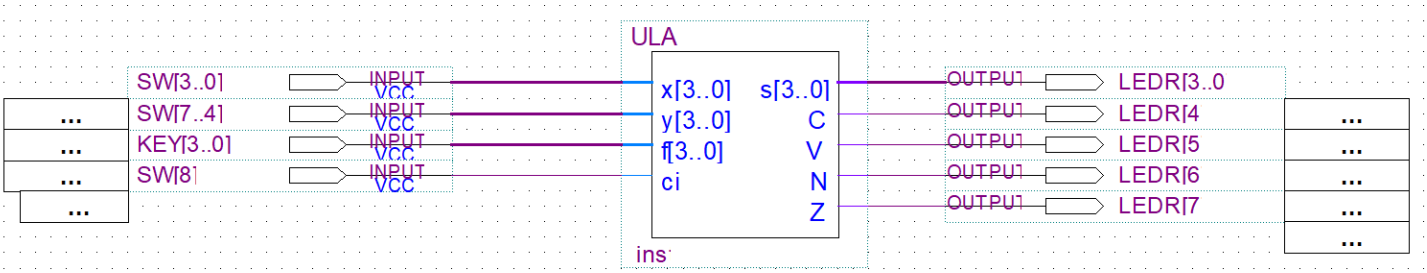

Figura 38: Esquemático do circuito de teste da ULA.

Todos os testes executados anteriormente para o circuito ULA foram refeitos para o circuito de teste da ULA, seus resultados são mostrados nas figuras 39 a 55.

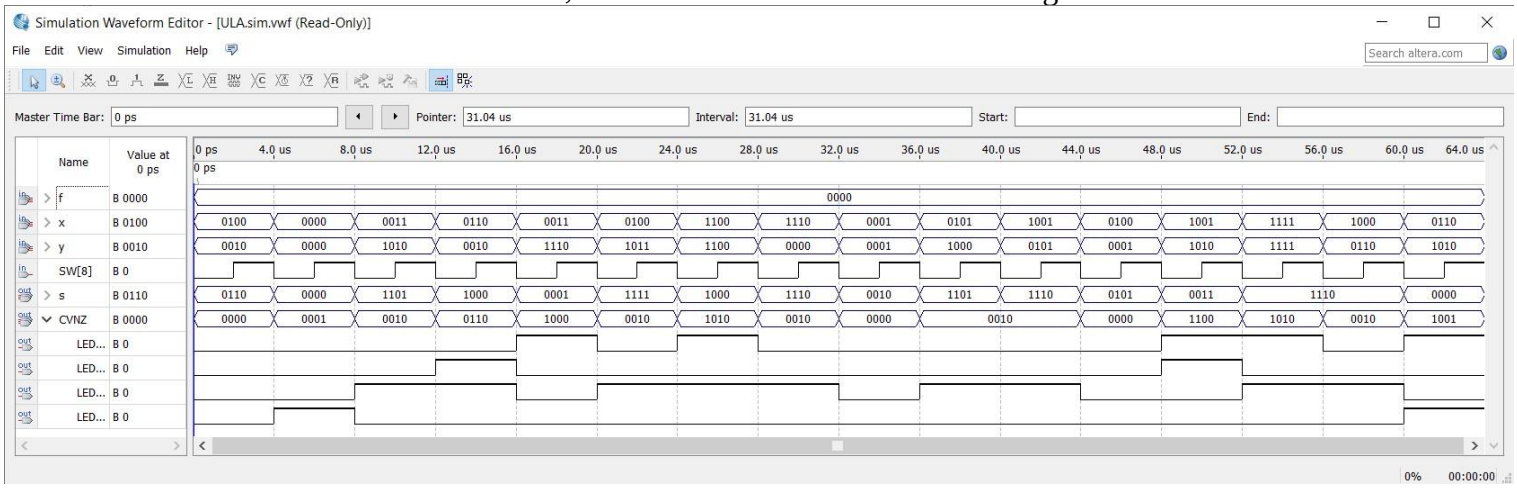

Figura 39: Simulação funcional da função ADD (com entrada seletora 0000)

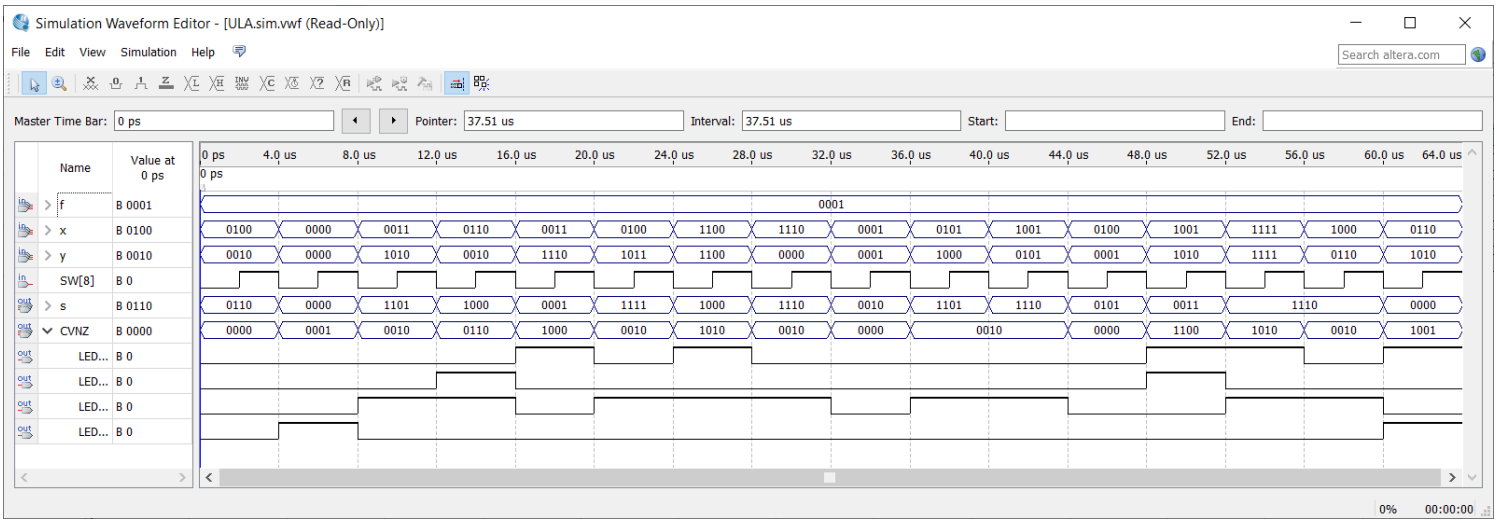

Figura 40: Simulação funcional da função ADD (com entrada seletora 0001)

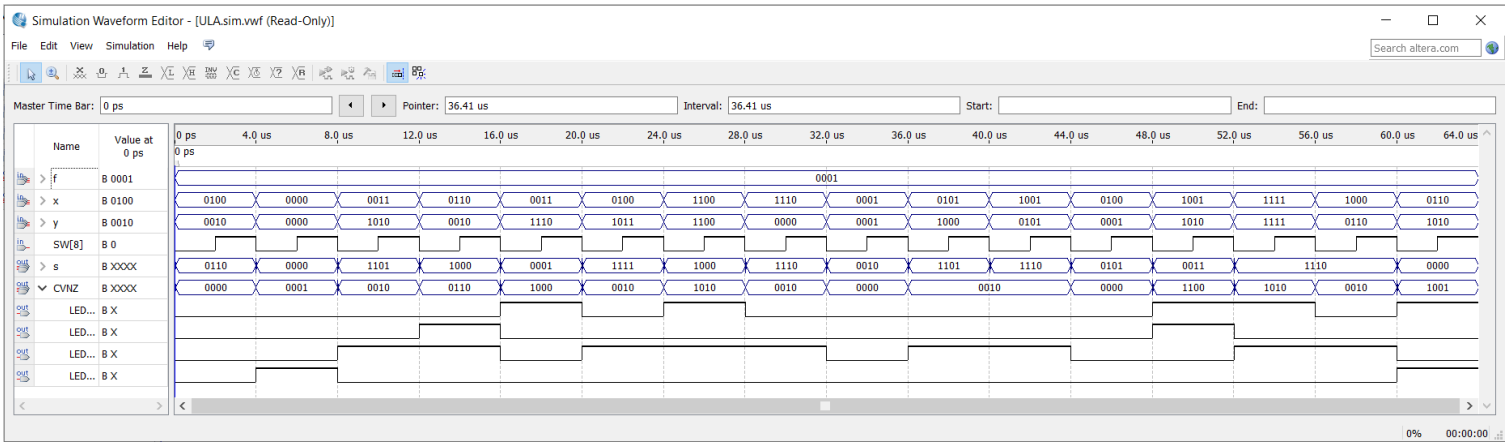

Figura 41: Simulação temporal da função ADD (com entrada seletora 0001)

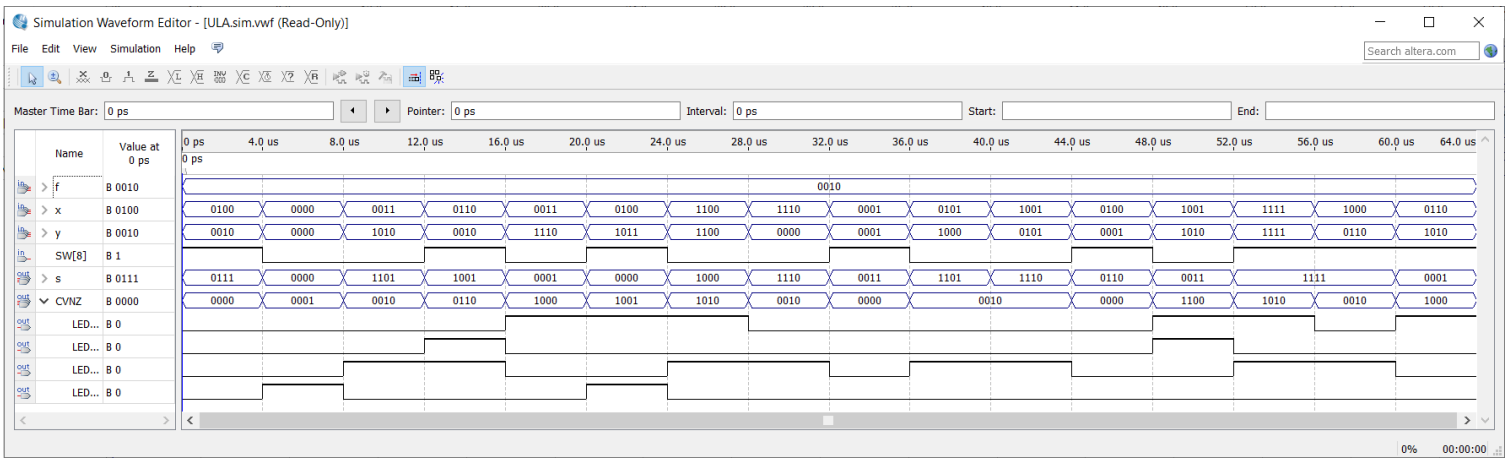

Figura 42: Simulação funcional da função ADDC (com entrada seletora 0010)

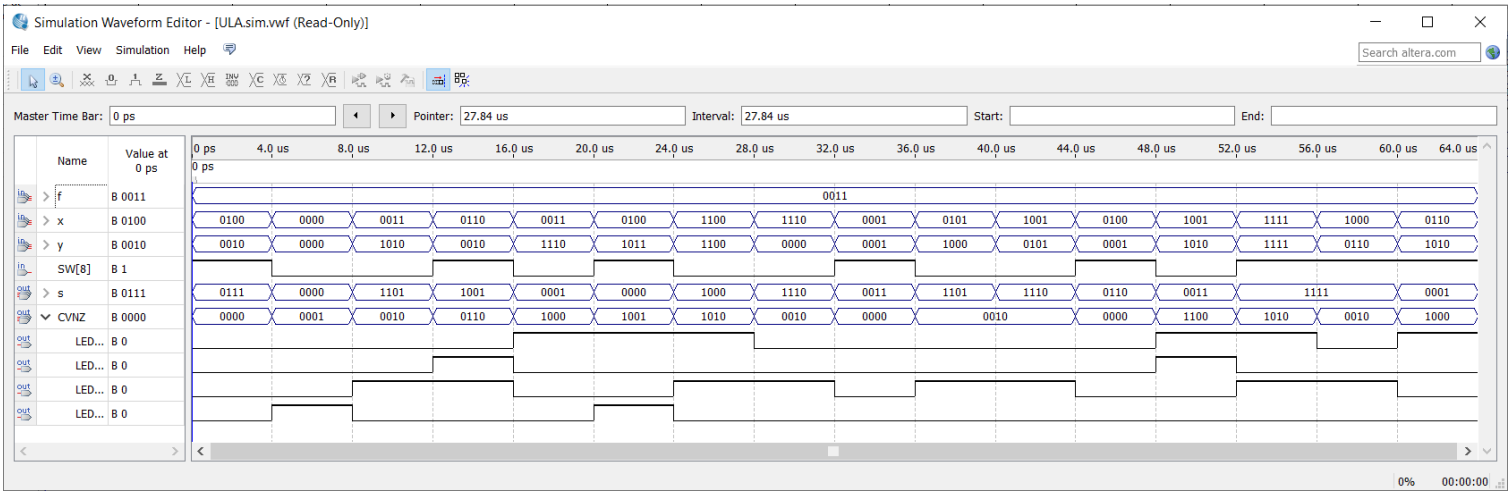

Figura 43: Simulação funcional da função ADDC (com entrada seletora 0011)

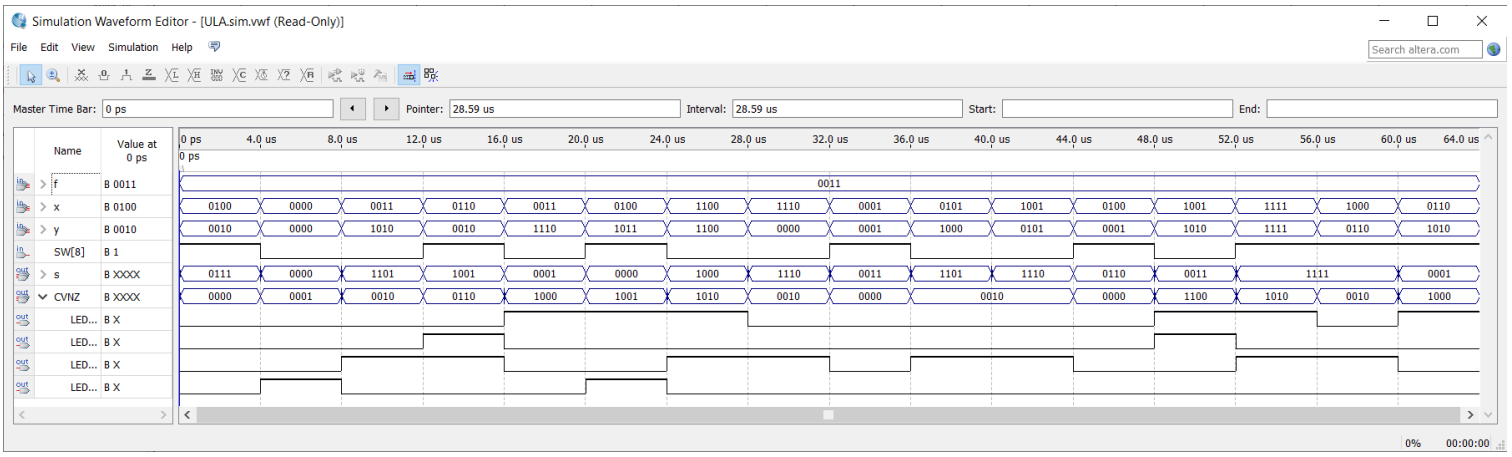

Figura 43: Simulação temporal da função ADDC (com entrada seletora 0011)

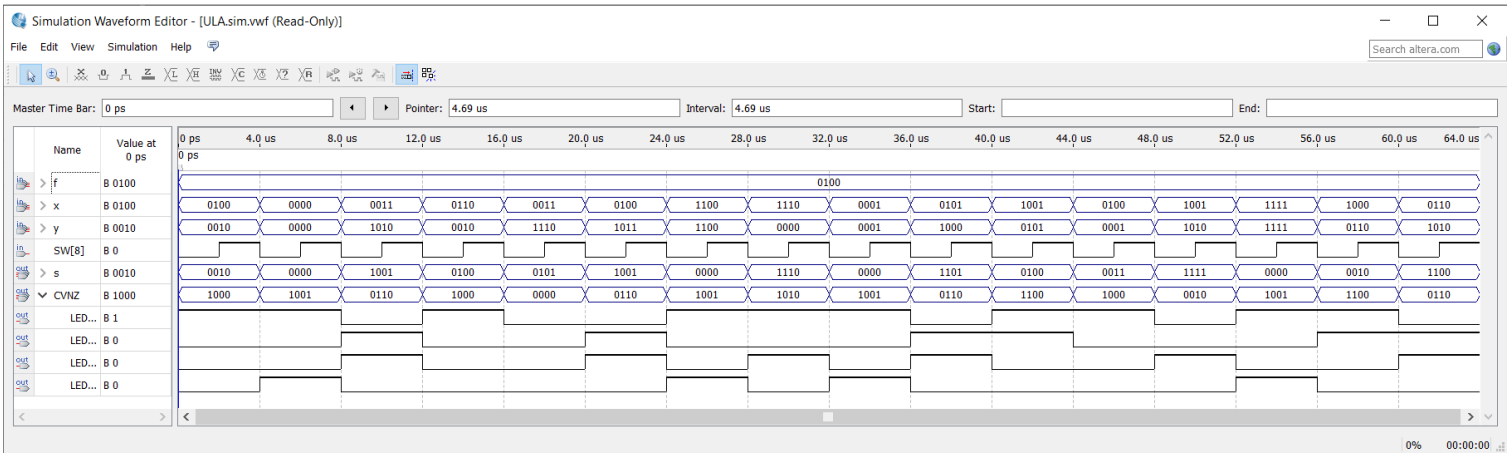

Figura 44: Simulação funcional da função SUB (com entrada seletora 0100)

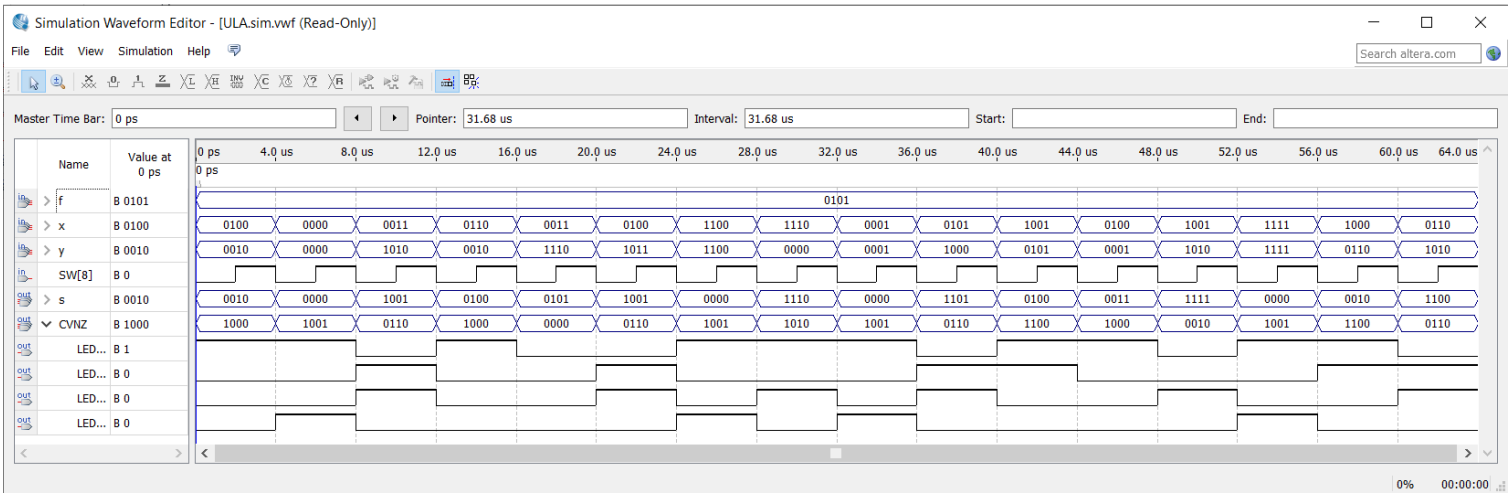

Figura 45: Simulação funcional da função SUB (com entrada seletora 0101)

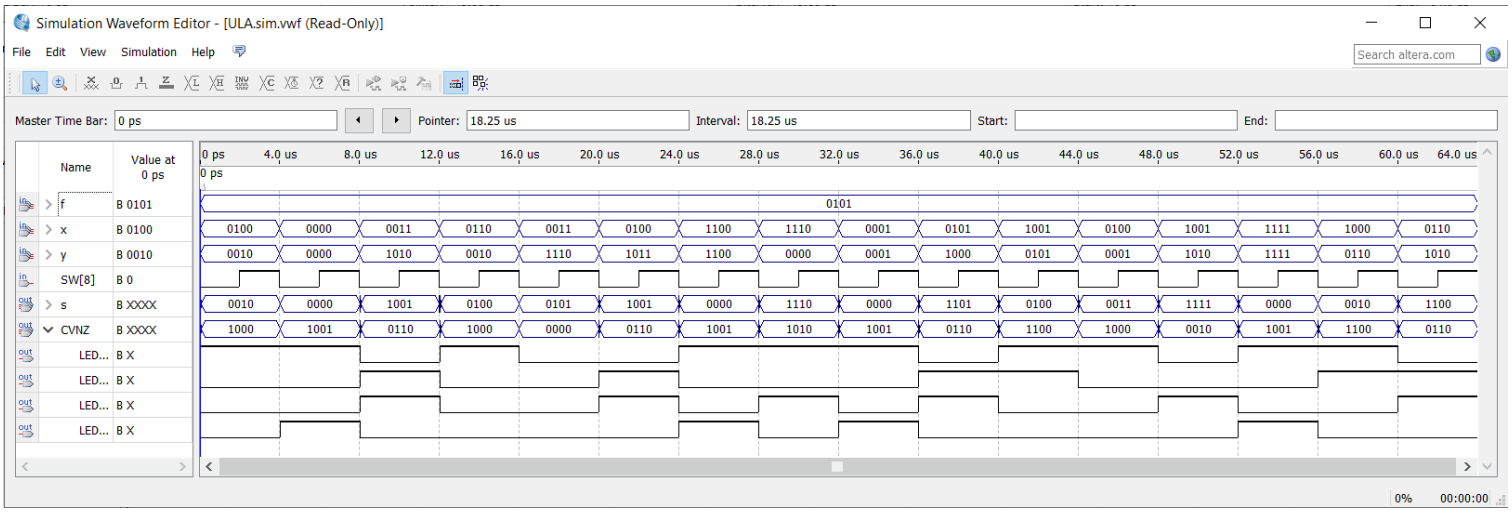

Figura 46: Simulação temporal da função SUB (com entrada seletora 0101)

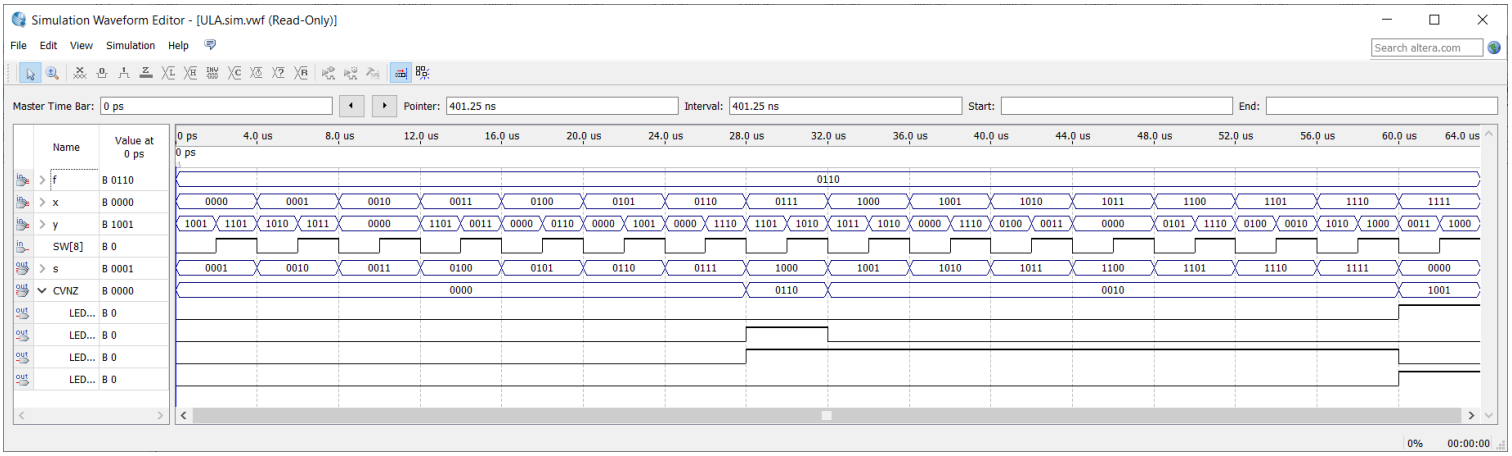

Figura 47: Simulação funcional da função INC (com entrada seletora 0110)

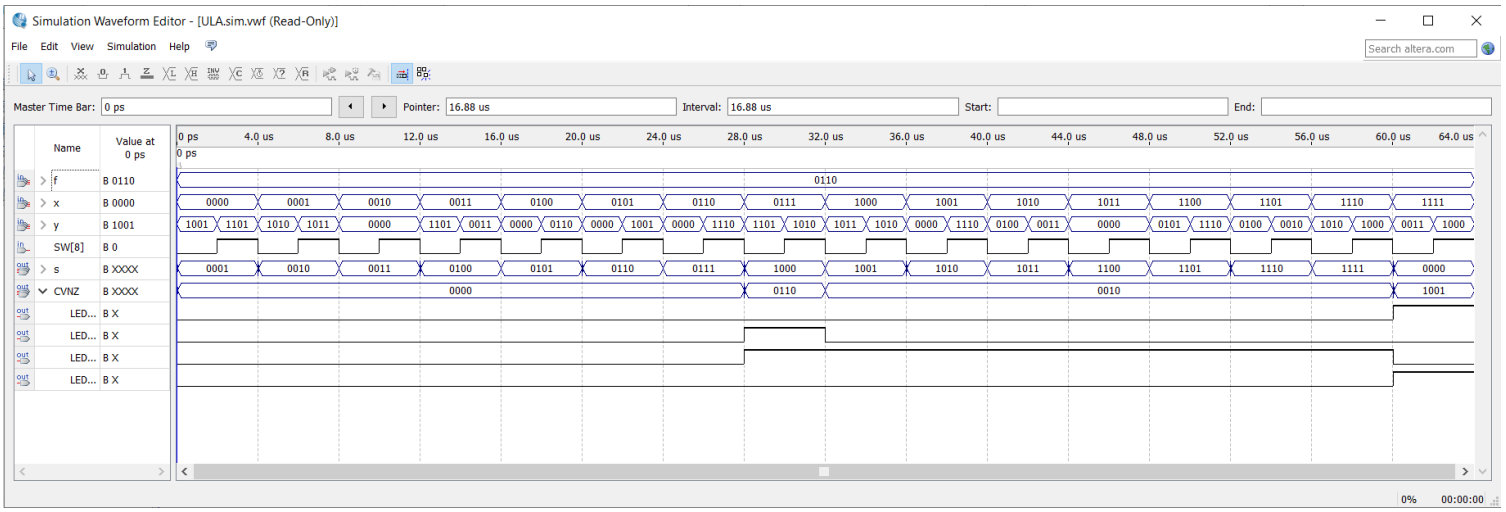

Figura 48: Simulação temporal da função INC (com entrada seletora 0110)

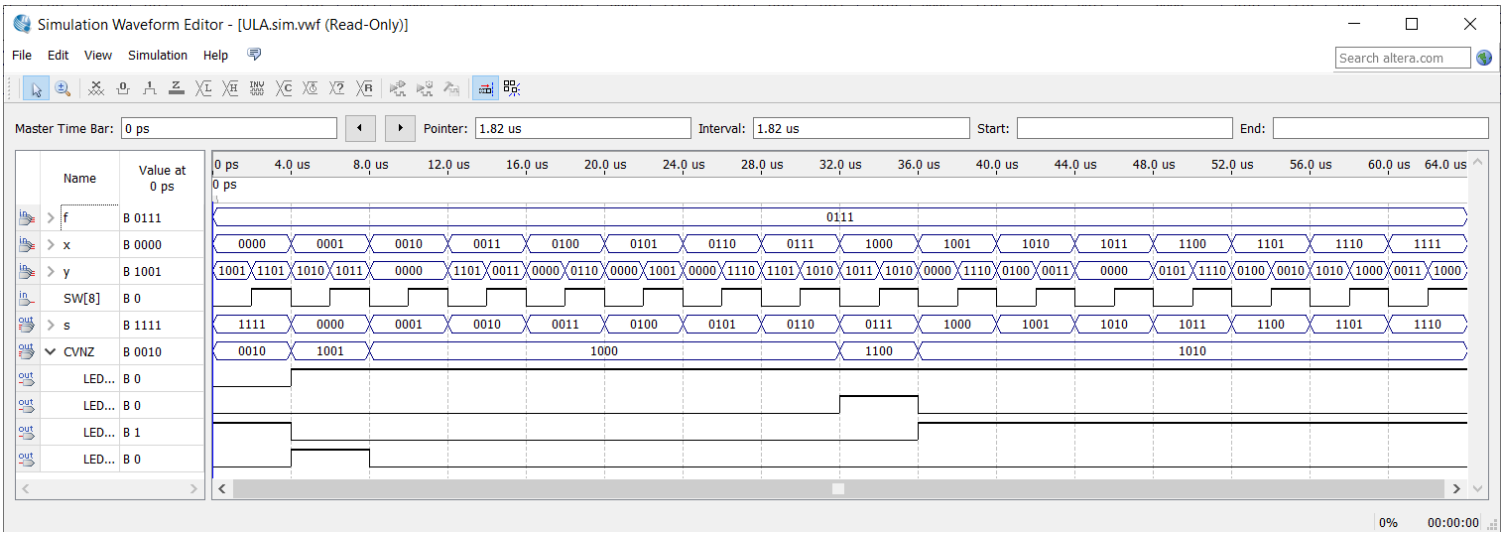

Figura 49: Simulação funcional da função DEC (com entrada seletora 0111)

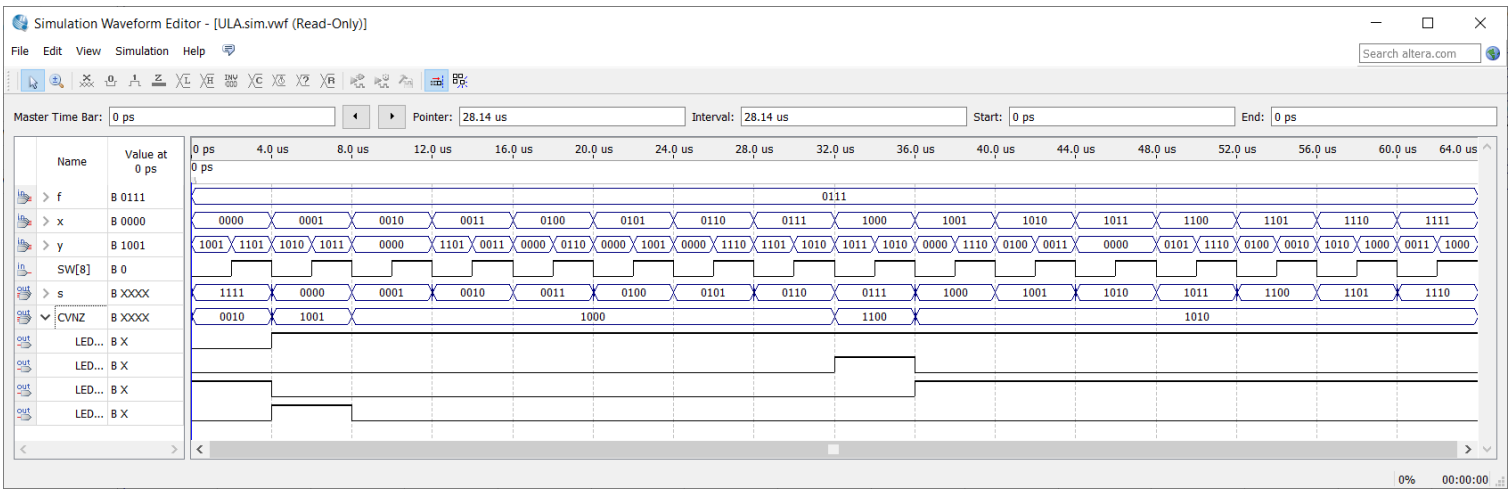

Figura 50: Simulação temporal da função DEC (com entrada seletora 0111)

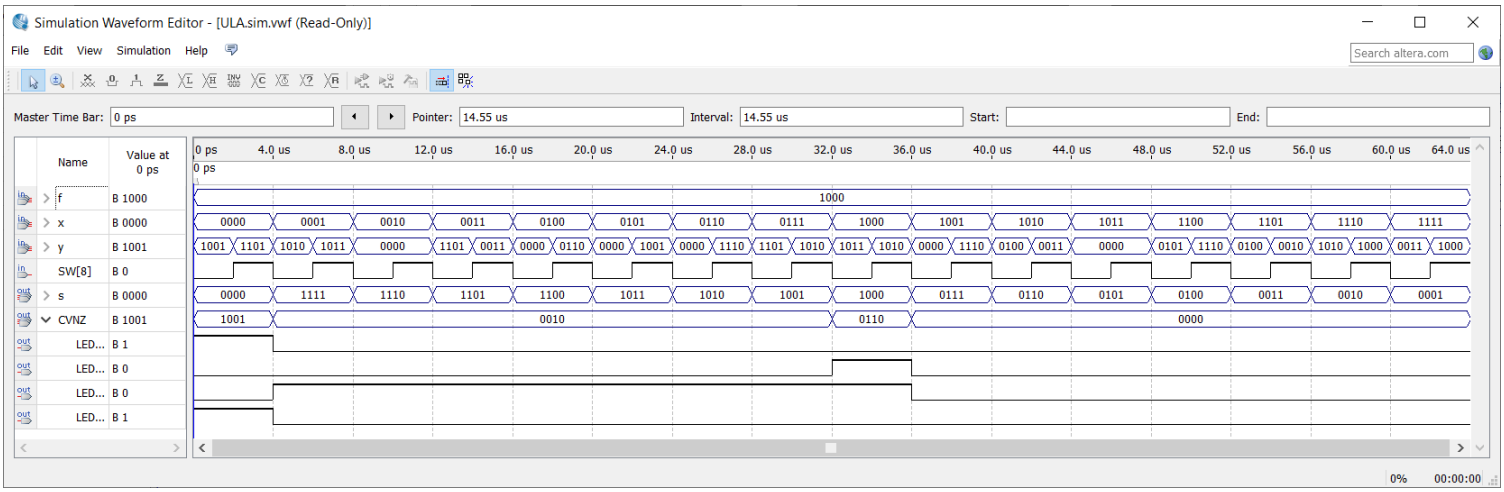

Figura 51: Simulação funcional da função NEG (com entrada seletora 1000)

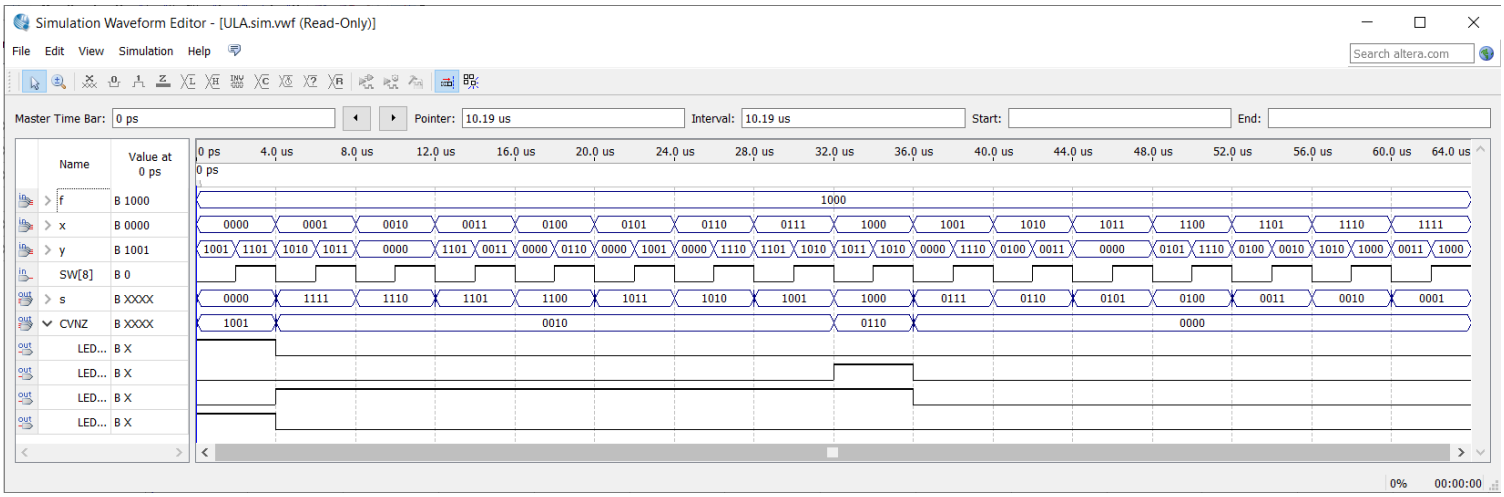

Figura 52: Simulação temporal da função NEG (com entrada seletora 1000)

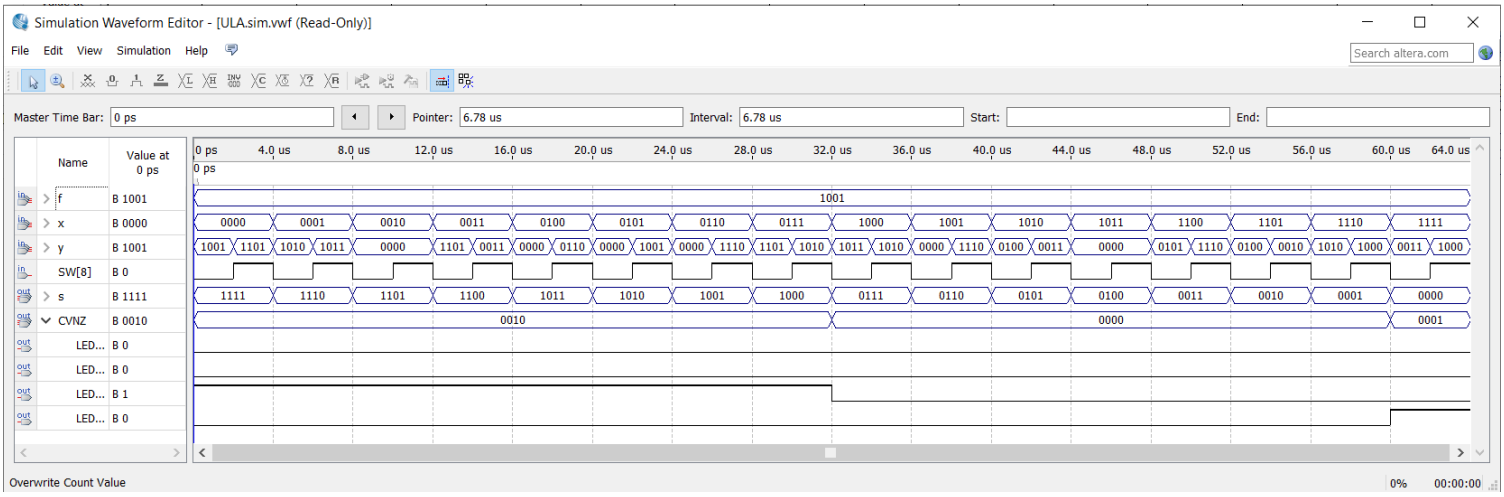

Figura 53: Simulação funcional da função CMPL (com entrada seletora 1001)

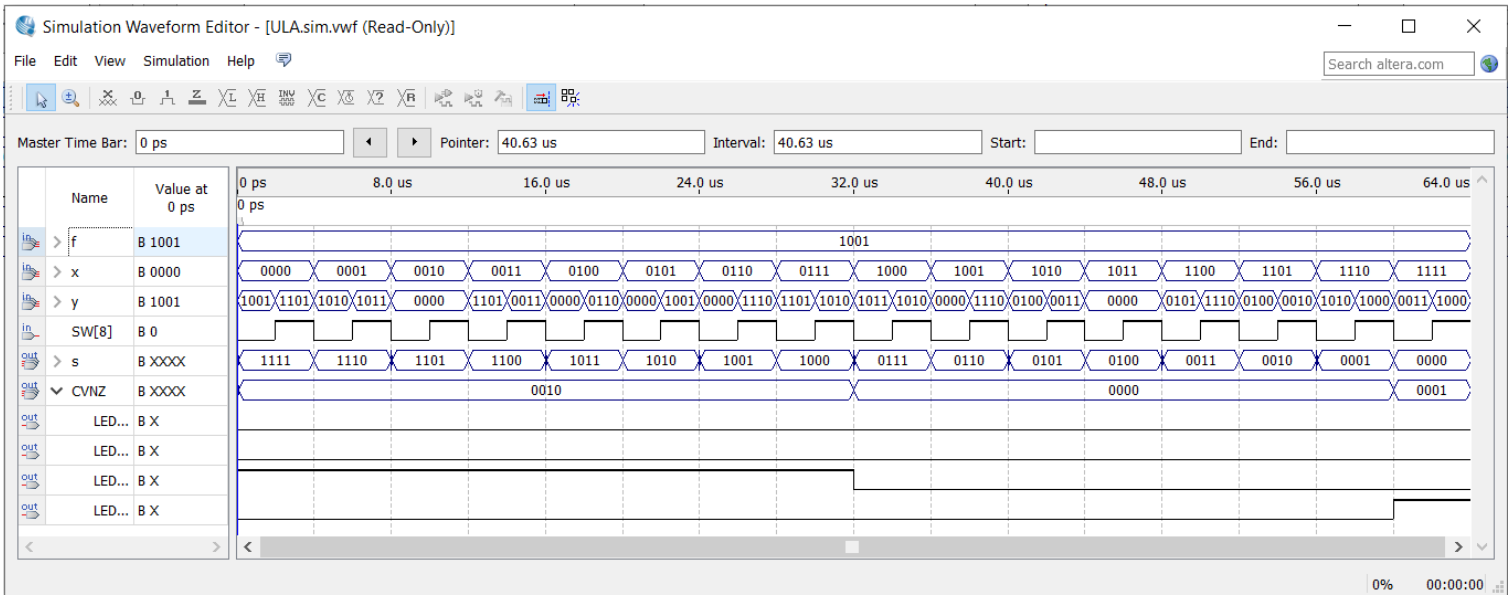

Figura 54: Simulação temporal da função CMPL (com entrada seletora 1001)