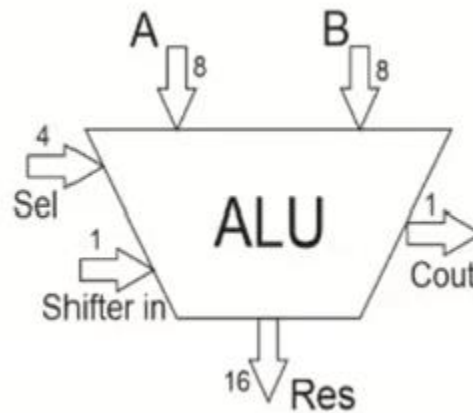


ALU



Selector	Operación
0000	$A+B$
0001	$A-B$
0010	$A*B$
0011	A/B
0100	$A\%B$
0101	$A \gg B$
0110	$A \gg B$, Acarreo
0111	$A \ll B$
1000	$A \ll B$, Acarreo
1001	$\&A$
1010	$ A$
1011	$\wedge A$
1100	$A \& B$
1101	$A B$
1110	$A \wedge B$
1111	00: $A > B$, 01: $A < B$, 10: $A = B$,

Se realiza la ALU de el mismo tamaño de la entrada, en clase se comentó que había esta posibilidad. A continuación, se muestra el código, así como el buen funcionamiento de la implementación.

/ALU_tb/Ctrl_tb	1111	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
/ALU_tb/shifter_tb	0																
/ALU_tb/A_tb	4	4	7	4	8	2	5			7				5			4
/ALU_tb/B_tb	8	4	2	3	2	4	1							4			8
/ALU_tb/C_tb	1	8	5	12	4	2	2	10		11	0	1		4	5	1	1
/ALU_tb/Carry_tb	St0																

```

4'b0000: c_temp_reg = A + B;
4'b0001: c_temp_reg = A - B;
4'b0010: c_temp_reg = A * B;
4'b0011: c_temp_reg = A / B;
4'b0100: c_temp_reg = A % B;

```

/ALU_tb/Ctrl_tb	1111	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
/ALU_tb/shifter_tb	0																
/ALU_tb/A_tb	0100	0100	0111	0100	1000	0010	0101				0111			0101			0100
/ALU_tb/B_tb	1000	0100	0010	0011	0010	0100	0001							0100			1000
/ALU_tb/C_tb	0001	1000	0101	1100	0100	0010	0010	1010		1011	0000	0001		0100	0101	0001	0001
/ALU_tb/Carry_tb	St0																

```

4'b0101: c_temp_reg = A>>B;
4'b0110:
if(shifter==1'b1)
    c_temp_reg = ~(~A>>B);
else
    c_temp_reg = A>>B;
4'b0111: c_temp_reg = A<<B;
4'b1000:
if(shifter==1'b1)
    c_temp_reg = ~(~A<<B);
else
    c_temp_reg = A<<B;

```

/ALU_tb/Ctrl_tb	1111	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
/ALU_tb/shifter_tb	0																
/ALU_tb/A_tb	0100	0100	0111	0100	1000	0010	0101				0111			0101			0100
/ALU_tb/B_tb	1000	0100	0010	0011	0010	0100	0001							0100			1000
/ALU_tb/C_tb	0001	1000	0101	1100	0100	0010	0010	1010		1011	0000	0001		0100	0101	0001	0001
/ALU_tb/Carry_tb	St0																

```

4'b1001: c_temp_reg = &A;
4'b1010: c_temp_reg = |A;
4'b1011: c_temp_reg = ^A;

```

/ALU_tb/Ctrl_tb	1111	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
/ALU_tb/shifter_tb	0																
/ALU_tb/A_tb	0100	0100	0111	0100	1000	0010	0101			0111				0101			0100
/ALU_tb/B_tb	1000	0100	0010	0011	0010	0100	0001							0100			1000
/ALU_tb/C_tb	0001	1000	0101	1100	0100	0010	0010	1010		1011	0000	0001		0100	0101	0001	0001
/ALU_tb/Carry_tb	St0																

```

4'b1100: c_temp_reg = A & B;
4'b1101: c_temp_reg = A | B;
4'b1110: c_temp_reg = A ^ B;

```

/ALU_tb/Ctrl_tb	1111	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
/ALU_tb/shifter_tb	0																
/ALU_tb/A_tb	0100	0100	0111	0100	1000	0010	0101			0111				0101			0100
/ALU_tb/B_tb	1000	0100	0010	0011	0010	0100	0001							0100			1000
/ALU_tb/C_tb	0001	1000	0101	1100	0100	0010	0010	1010		1011	0000	0001		0100	0101	0001	0001
/ALU_tb/Carry_tb	St0																

```

4'b1111: begin
    if(A>B)c_temp_reg = 0;
    else if (A<B)c_temp_reg = 1;
    else c_temp_reg = 2;
    end
default: c_temp_reg = 0;
endcase

if (A+B > mask)
begin
    carry_reg = 1'b1;
end
else begin
    carry_reg = 1'b0;
end
end

```

La practica tuvo un resultado satisfactorio, sin embargo, al momento de la realización se presentó un inconveniente que no se pudo resolver, el usar una entrada como variable. A pesar de este hecho, se utilizo una forma diferente de realizarla.