Parte 1 (5 pontos)

Projeto de uma unidade lógica e aritmética (**ULA**) de 4 bits com 8 operações conforme a tabela abaixo:

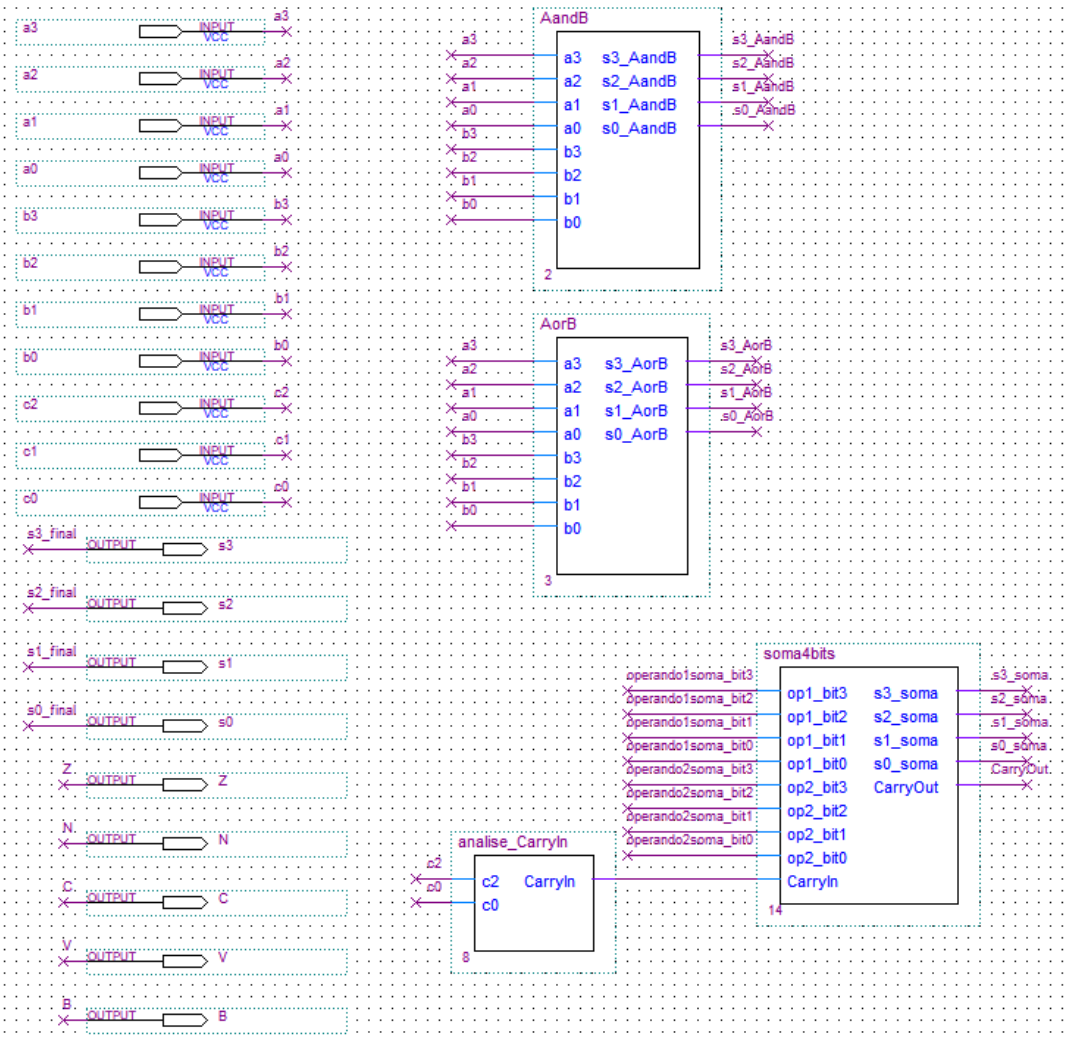
C2	C1	C0	Operação
0	0	0	Passa B (S = B)
0	0	1	A AND B
0	1	0	A OR B
0	1	1	NOT (A)
1	0	0	NEG (A)
1	0	1	(ADD) A + B (soma em complemento de 2)
1	1	0	(SUB) A – B (subtração em complemento de 2)
1	1	1	(SHL) Deslocamento à esquerda (x2) - *shift left* (A)

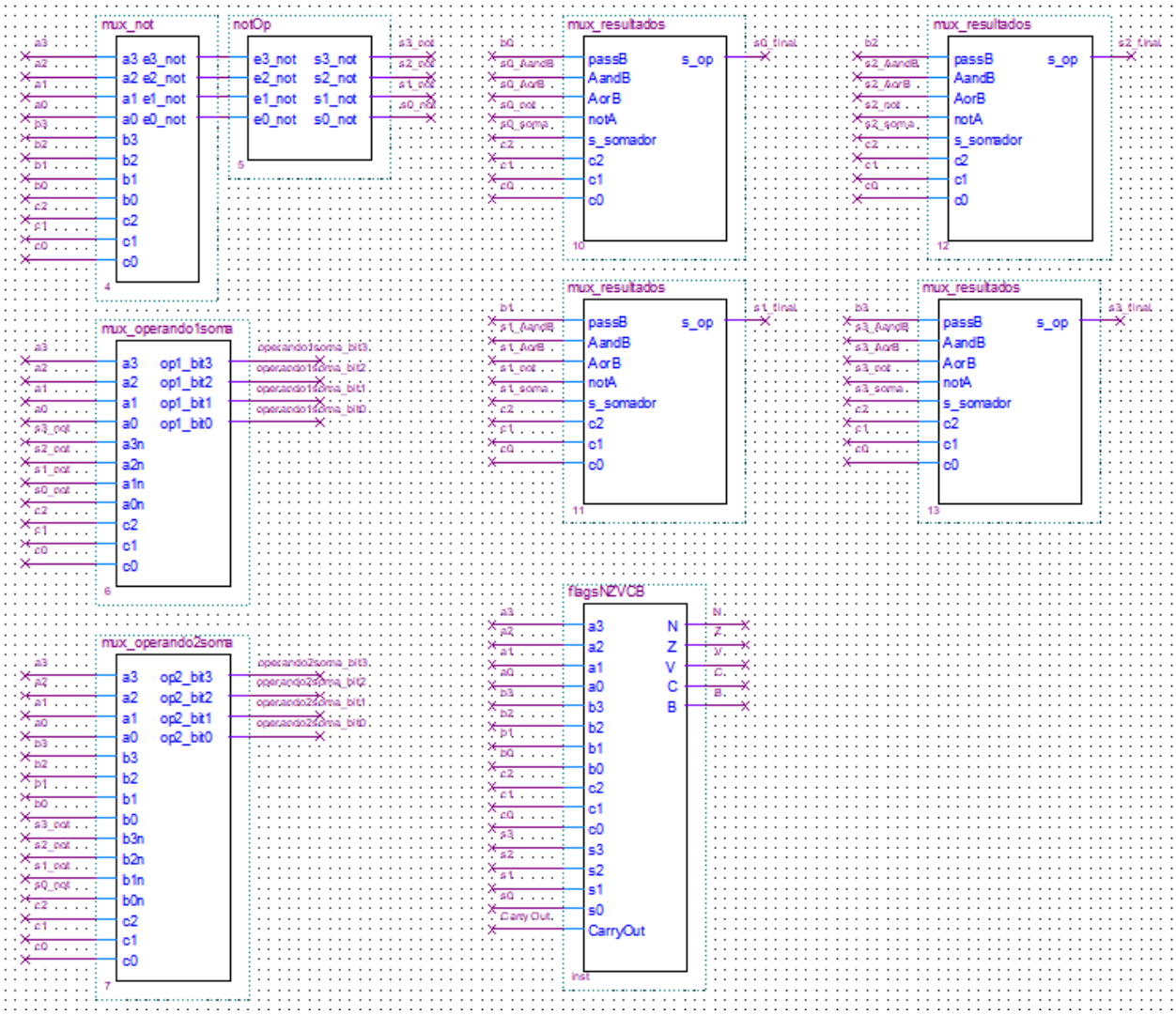
* A e B são vetores de 4 bits, assim como a saída S.

Devem ser gerados os seguintes sinais de condição (flags):
Z (zero), N (negativo), C (carry), B (borrow) e V (overflow).

Obs.: As operações NEG e SUB, em complemento de 2, podem ser construídos a partir de um somador: A – B = A + NOT(B) + 1. Portanto, é possível ter apenas um somador de 4 bits na ULA para realizar as 3 operações. Essa otimização será considerada na avaliação do trabalho.

Nas duas imagens abaixo, visualizamos a implementação geral da ULA (sem demonstrar de forma específica cada um dos circuitos lógicos responsáveis por cada operação), presente no arquivo ULA_main. Utilizou-se apenas um somador para realizar as operações NEG, ADD, SUB e SHL.



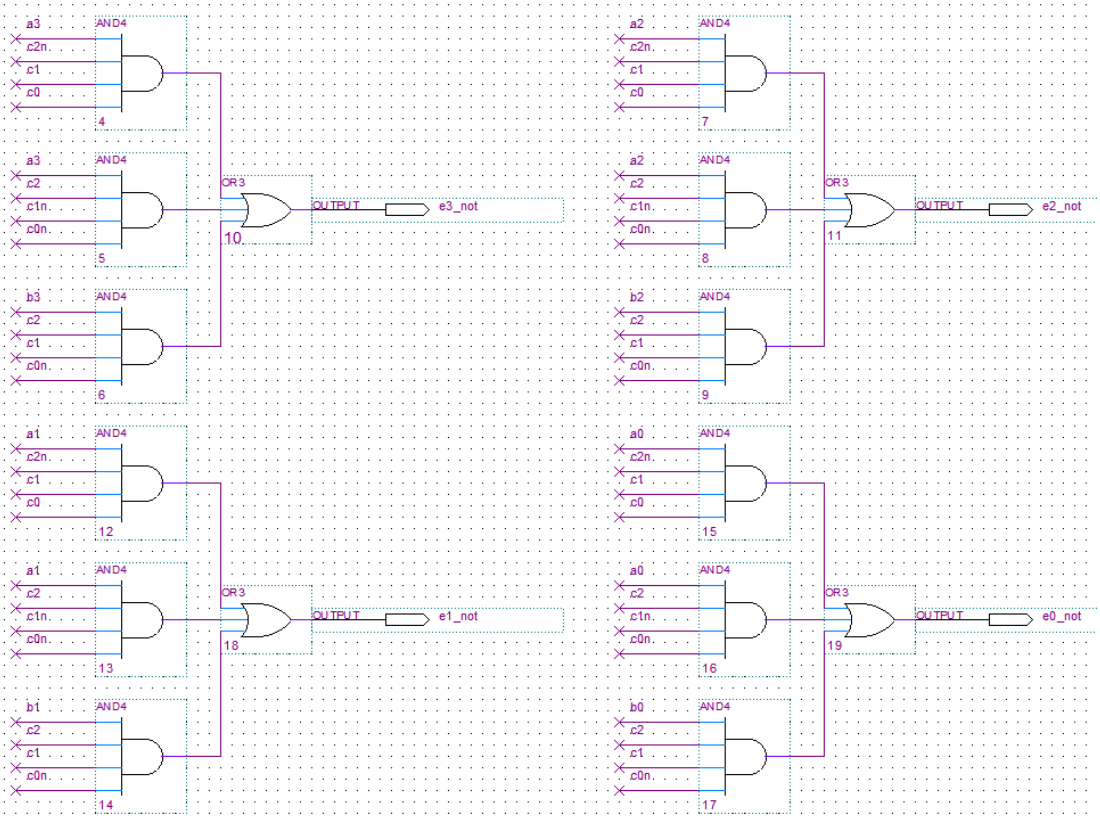


1 - CIRCUITOS AUXILIARES

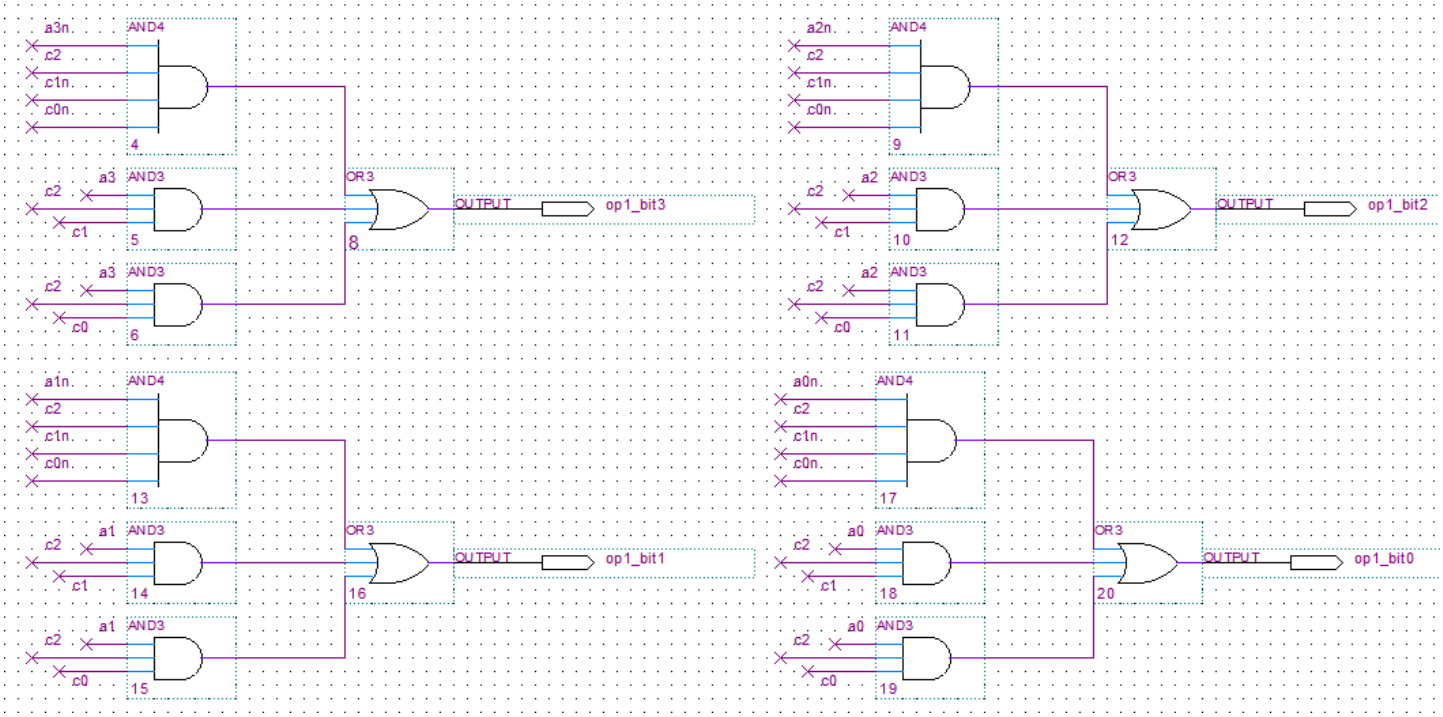
Abaixo, estão listados os circuitos que não são mencionados diretamente na especificação do trabalho, mas estão presentes na ULA para garantir seu funcionamento, junto com a sua função dentro da ULA e a sua representação interna:

analise_CarryIn - Circuito utilizado para analisar se será necessário inserir um bit “1” na entrada “Carry In” do somador de 4 bits. Isso ocorre caso o usuário escolha as funções NEG ou SUB nos bits de controle. Dessa forma, tem-se a soma “+ 1” que é necessária para realizar o complemento de um número, na representação em complemento de dois.

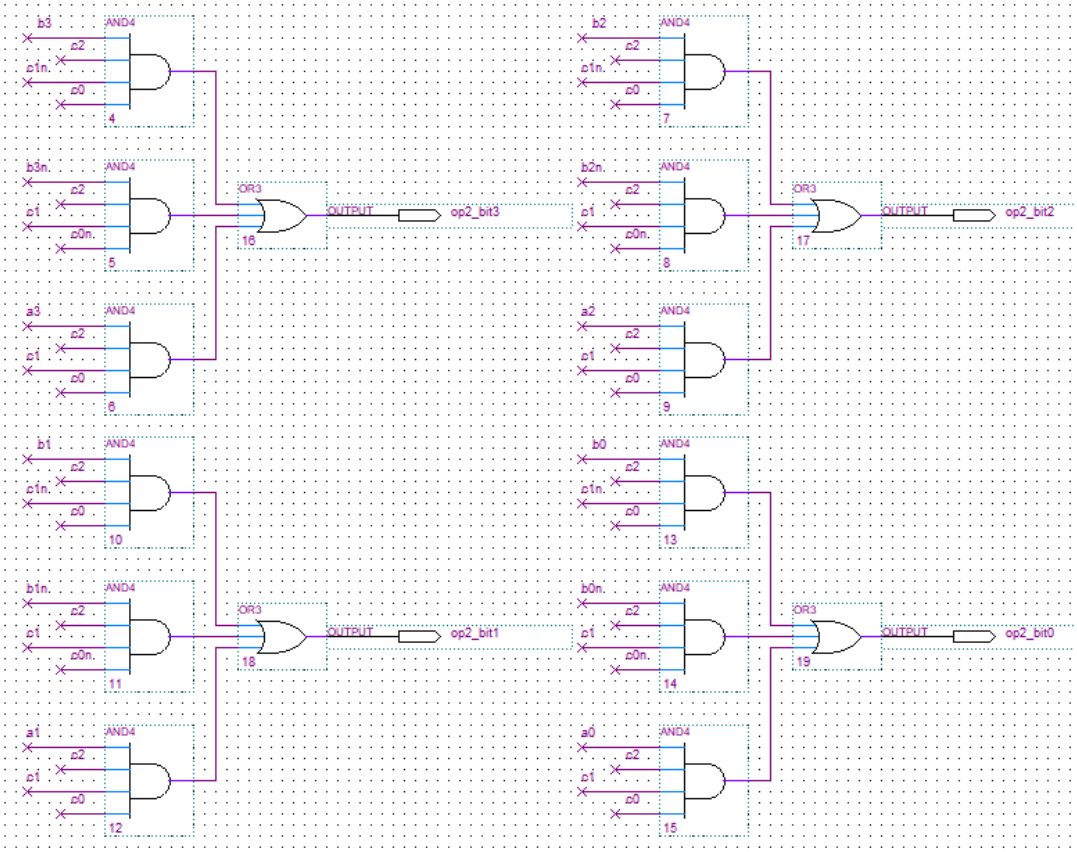
mux_not - Multiplexador 8x4 utilizado para definir qual será o operando (A ou B) que terá todos os seus bits negados, visto que é necessário negar o operando A quando as operações NOT(A) ou NEG(A) são escolhidas, ou negar o operando B quando a operação SUB é escolhida. A decisão ocorre com base na operação escolhida com os três bits de controle: c2, c1 e c0.



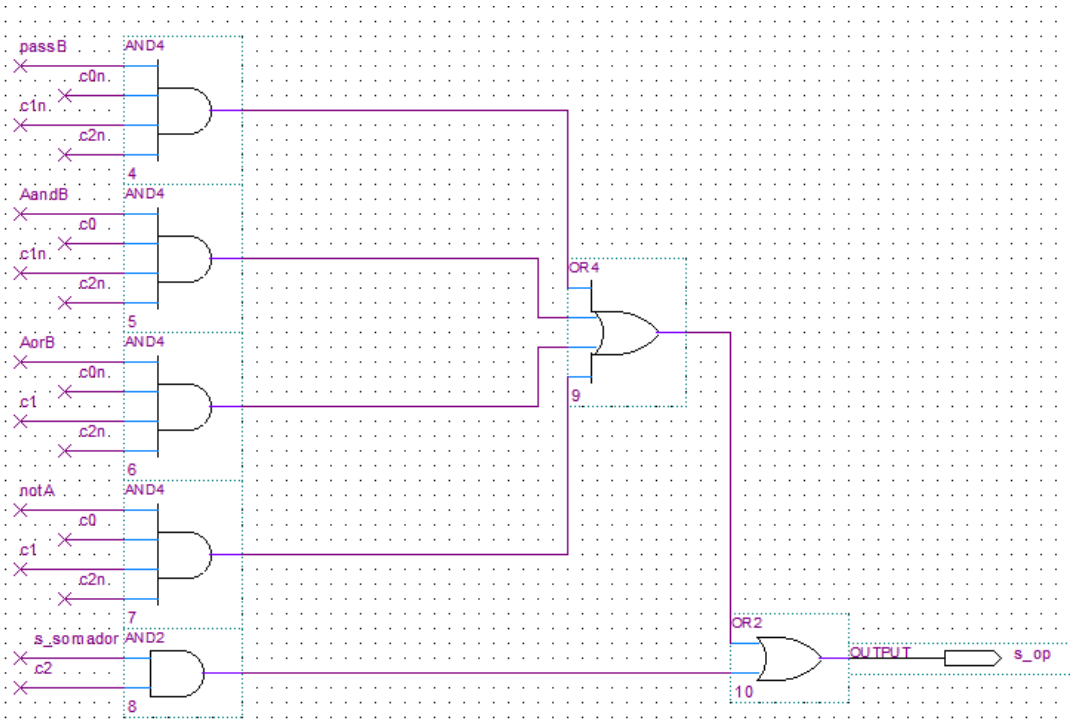
mux_operando1soma - Multiplexador 8x4 utilizado para definir qual será o primeiro operando (A ou NOT(A)) que será repassado ao somador de 4 bits. A decisão ocorre com base na operação escolhida com os três bits de controle: c2, c1 e c0.



mux_operando2soma - Multiplexador 8x4 utilizado para definir qual será o segundo operando (A, B ou NOT(B)) que será repassado ao somador de 4 bits. A decisão ocorre com base na operação escolhida com os três bits de controle: c2, c1 e c0.



mux_resultados - Esse multiplexador 5x1 é utilizado quatro vezes (uma para cada bit de saída), e decide qual dos resultados das operações realizadas pela ULA será repassado à saída final do circuito. A decisão ocorre com base na operação escolhida com os três bits de controle: c2, c1 e c0.



flags_NZVCB - Esse circuito recebe como entrada todos os bits do primeiro operando da ULA, do segundo operando da ULA e da saída final da ULA (12 bits no total), além dos três bits de controle. A função desse circuito é determinar quais flags da ULA serão ativadas, analisando os operandos, o resultado obtido e a operação escolhida.

OPERAÇÃO	FLAGS AFETADAS
PASSA B	N, Z
A AND B	N, Z
A OR B	N, Z
NOT (A)	N,Z
NEG (A)	N, Z, C
ADD	N, Z, V, C
SUB	N, Z, V, B
SHL	N, Z, C

Abaixo, seguem as explicações sobre o funcionamento de cada flag.

Flag N - acionada quando o bit mais significativo da saída final da ULA é igual a 1, indicando que o número é negativo, na representação em complemento de dois;

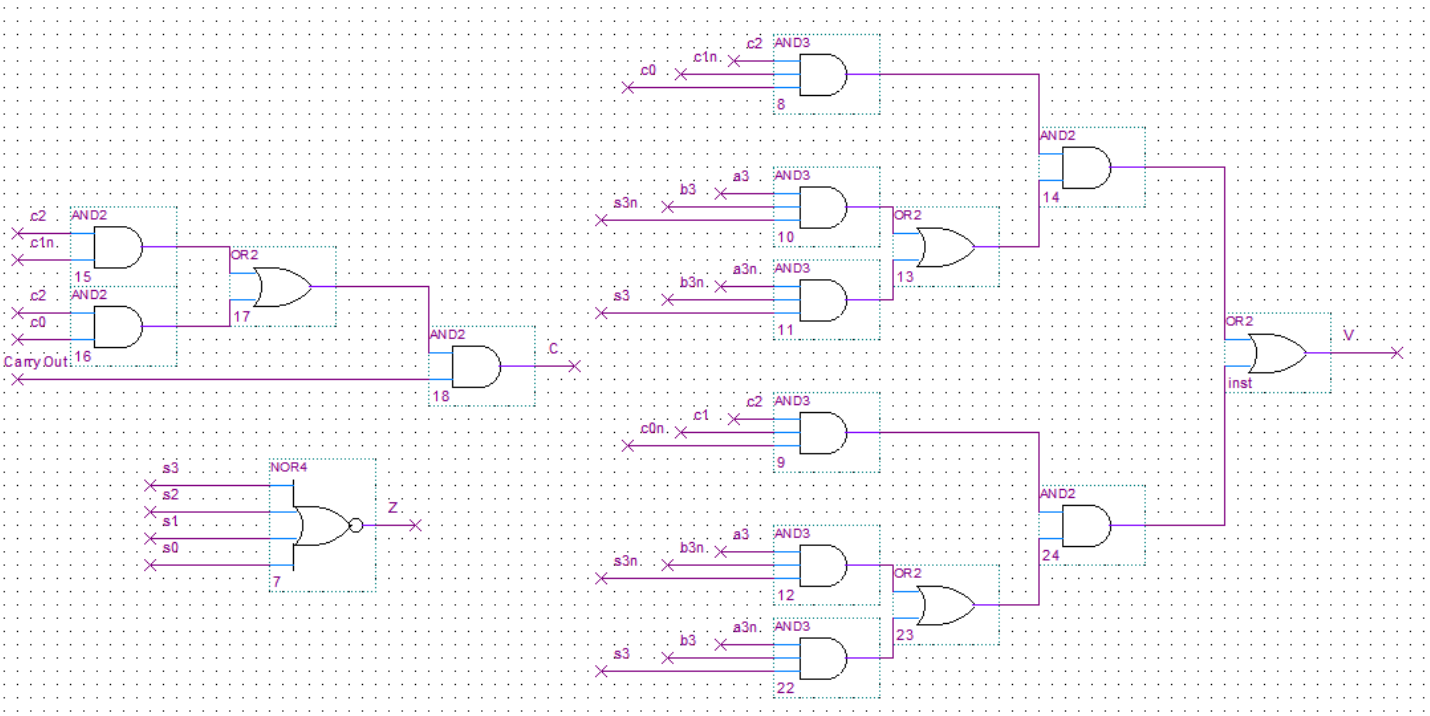
Flag Z - acionada quando todos os bits da saída são iguais a zero;

Flag V - acionada quando é realizada uma soma na qual os bits mais significativos dos dois operandos são iguais a um determinado valor booleano, e o bit mais significativo do valor da saída é diferente desse valor;

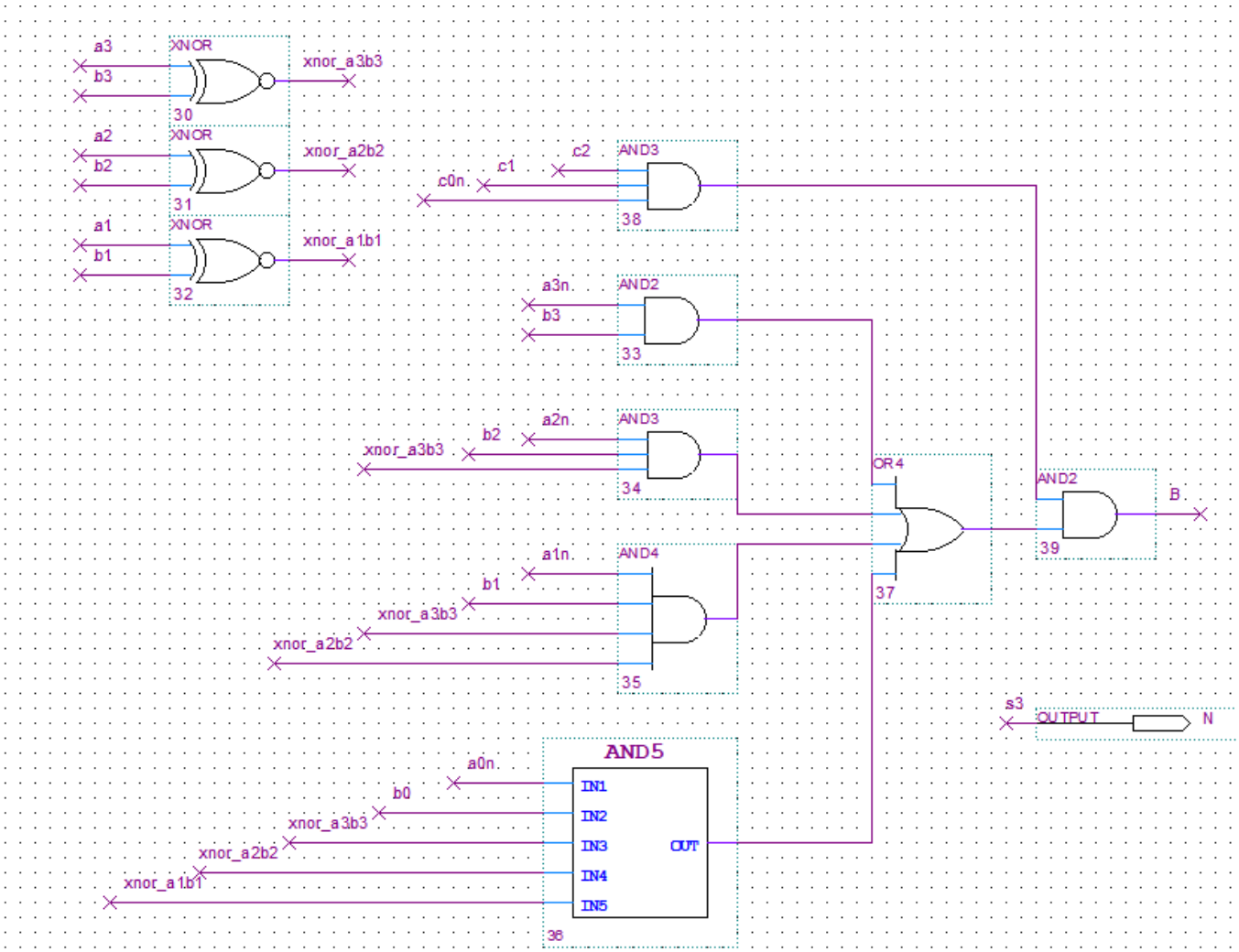
Flag C - acionada quando a operação ADD ou SHL gera “Carry out”;

Flag B - acionada quando ocorre uma operação SUB na qual o operando A possui valor em magnitude (ou seja, desconsiderando o sinal que seria imposto pela representação em complemento de dois) inferior ao valor B.

Abaixo, vemos o trecho do circuito que determina as saídas das flags Z, V e C.

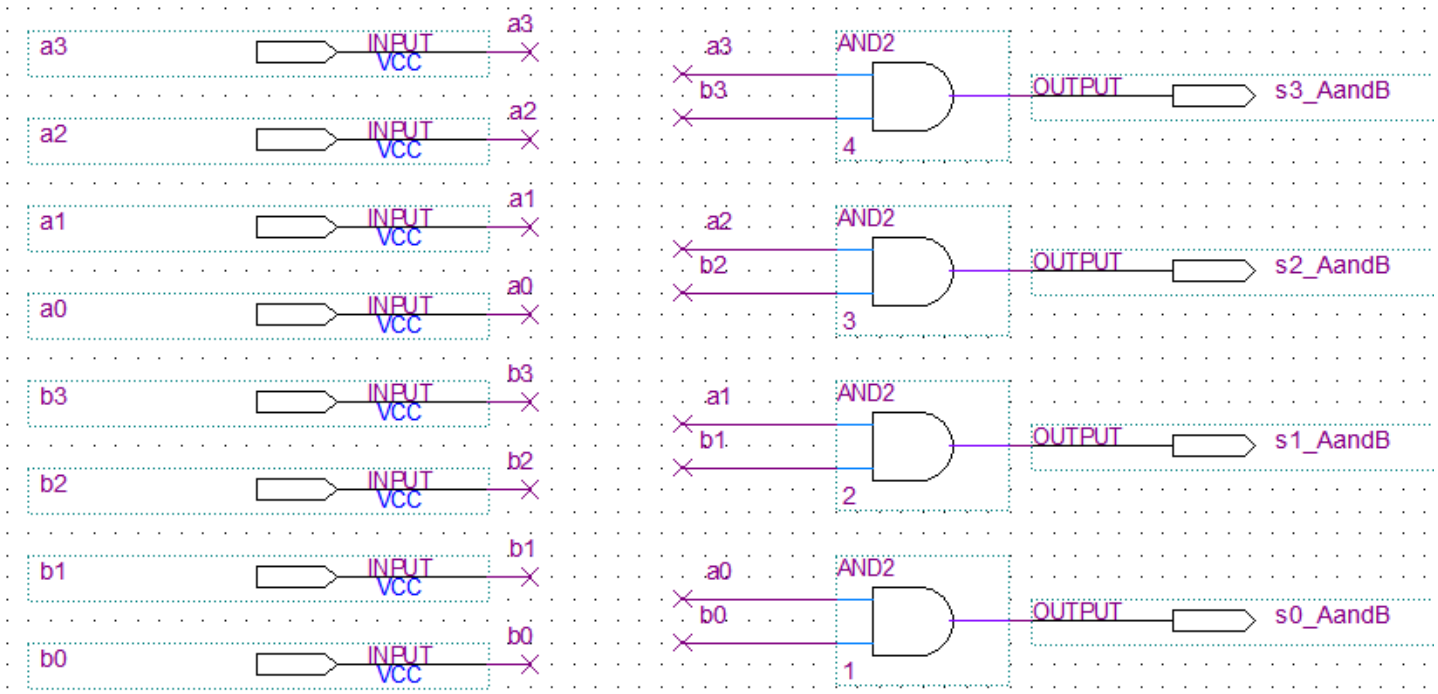


Em seguida, vemos o trecho que determina as saídas das flags N e B. Para determinar a saída da flag B, utilizou-se um circuito comparador do tipo (A < B). Para determinar a flag N, basta analisar se o bit mais significativo da saída (s3) é igual a 1, de forma direta.

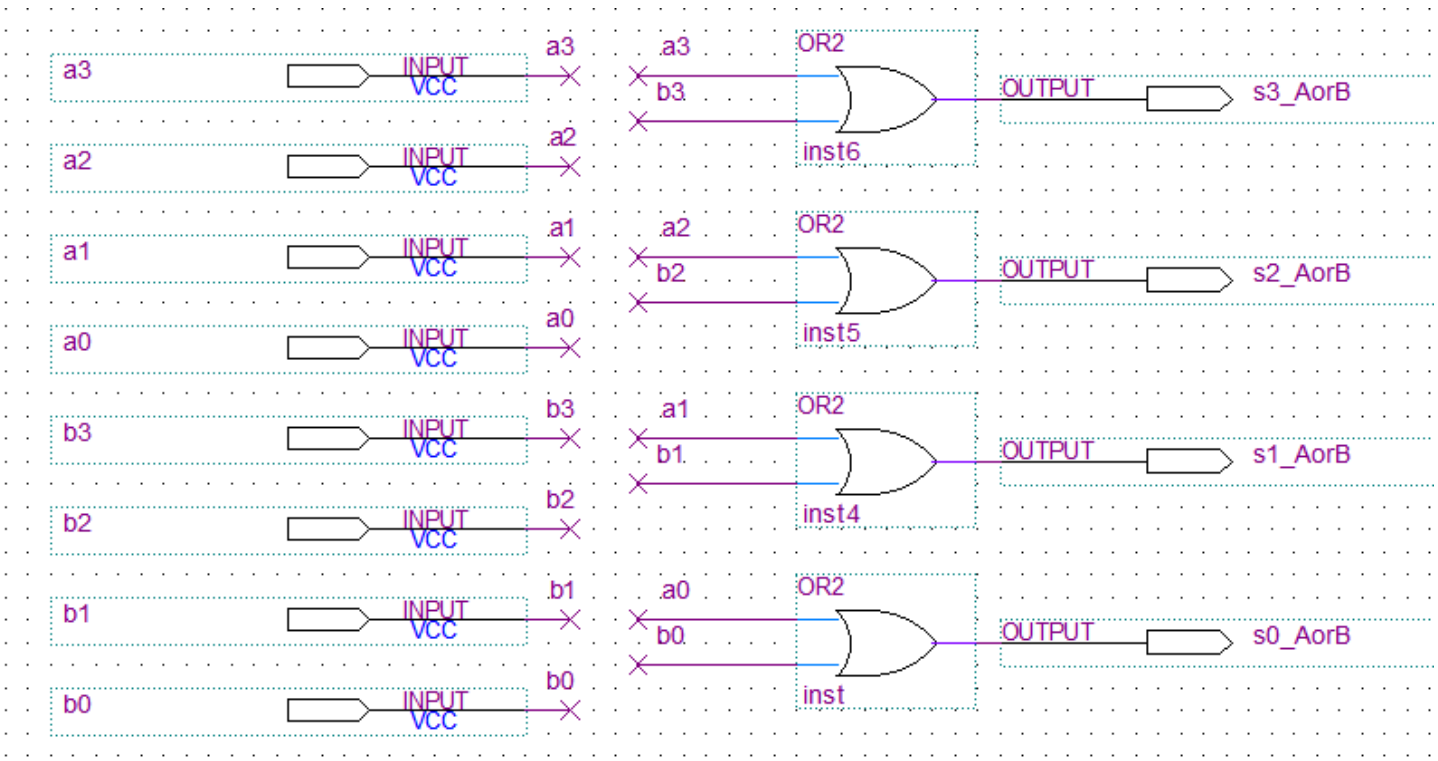


2 - CIRCUITOS PRINCIPAIS PARA CADA OPERAÇÃO

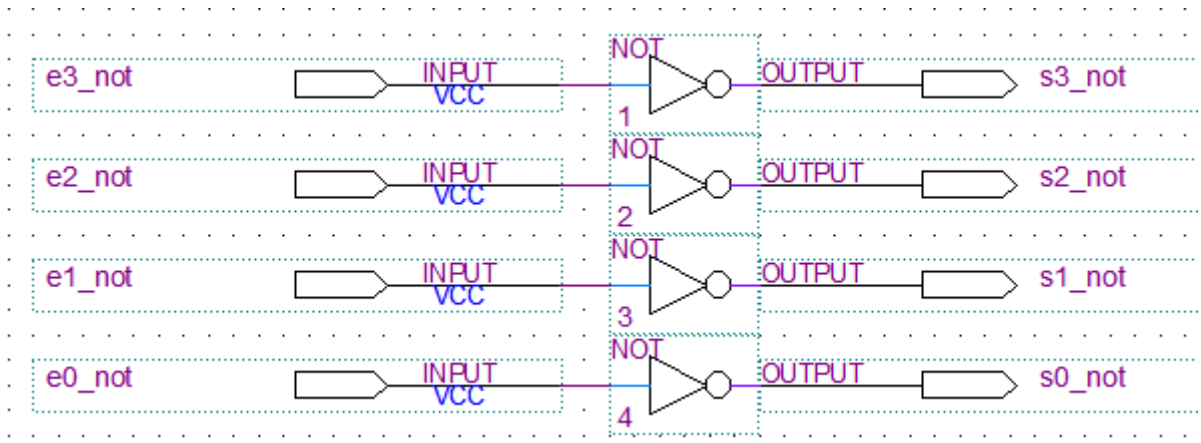
AandB - Circuito utilizado para aplicar a operação AND entre cada um dos bits dos operando A e B.



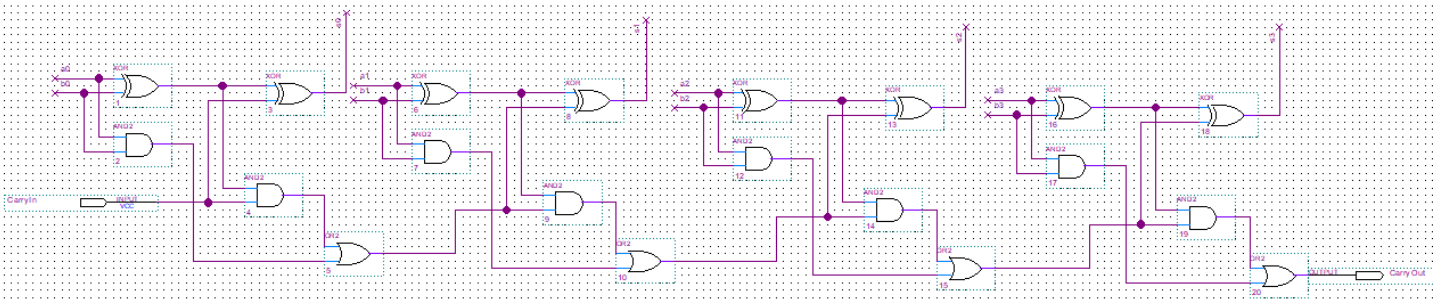
AorB - Circuito utilizado para aplicar a operação OR entre cada um dos bits dos operando A e B.



notOp - Circuito utilizado para aplicar a operação NOT em cada um dos bits do operando que é repassado pelo circuito **mux_not**, conforme já explicado.



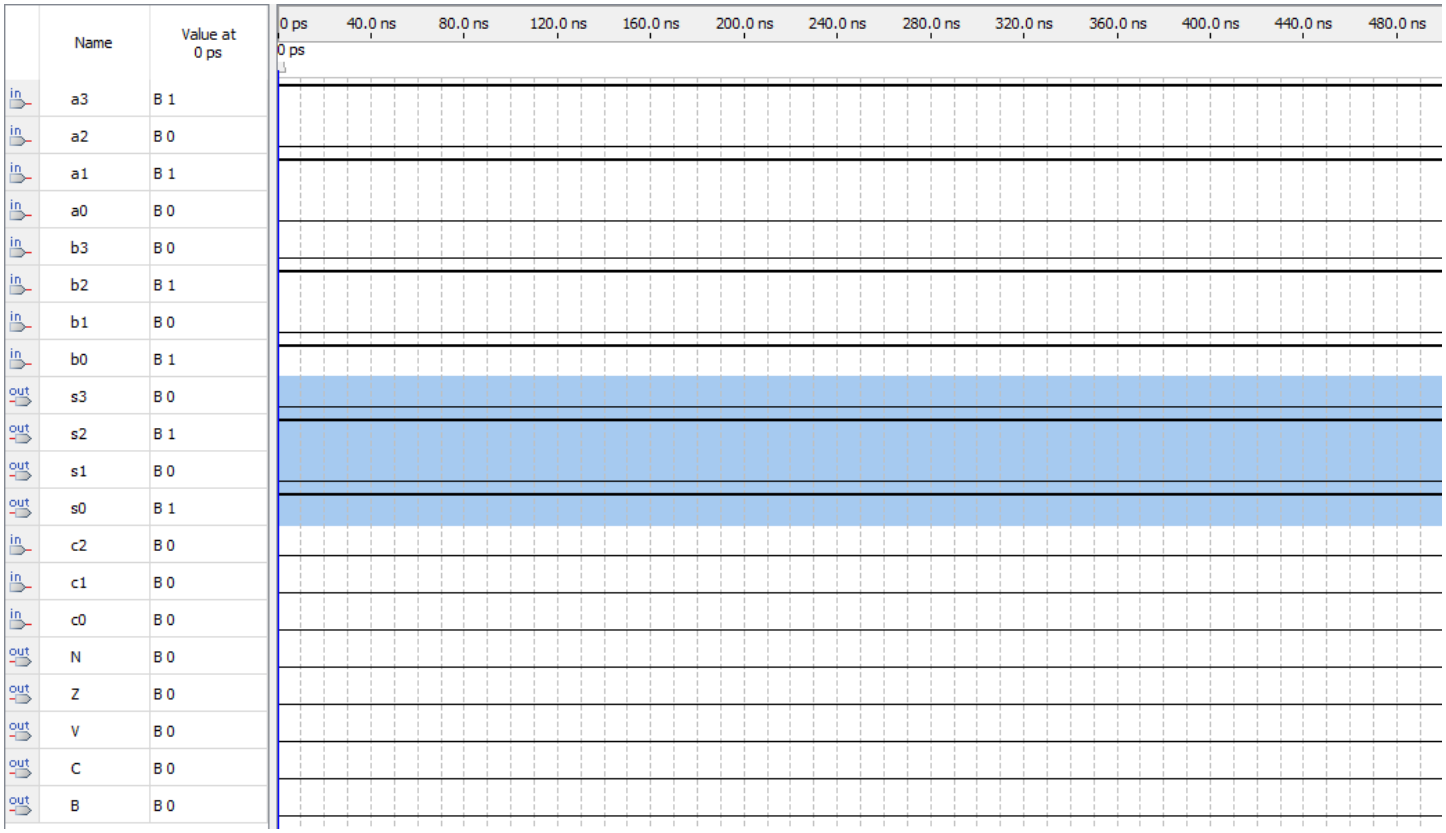
soma4bits - Circuito somador completo utilizado para realizar as operações ADD, SUB, NEG(A) e SHL(A), a partir dos operandos que são obtidos por meio dos circuitos **mux_operando1soma** e **mux_operando2soma**, conforme já explicado.



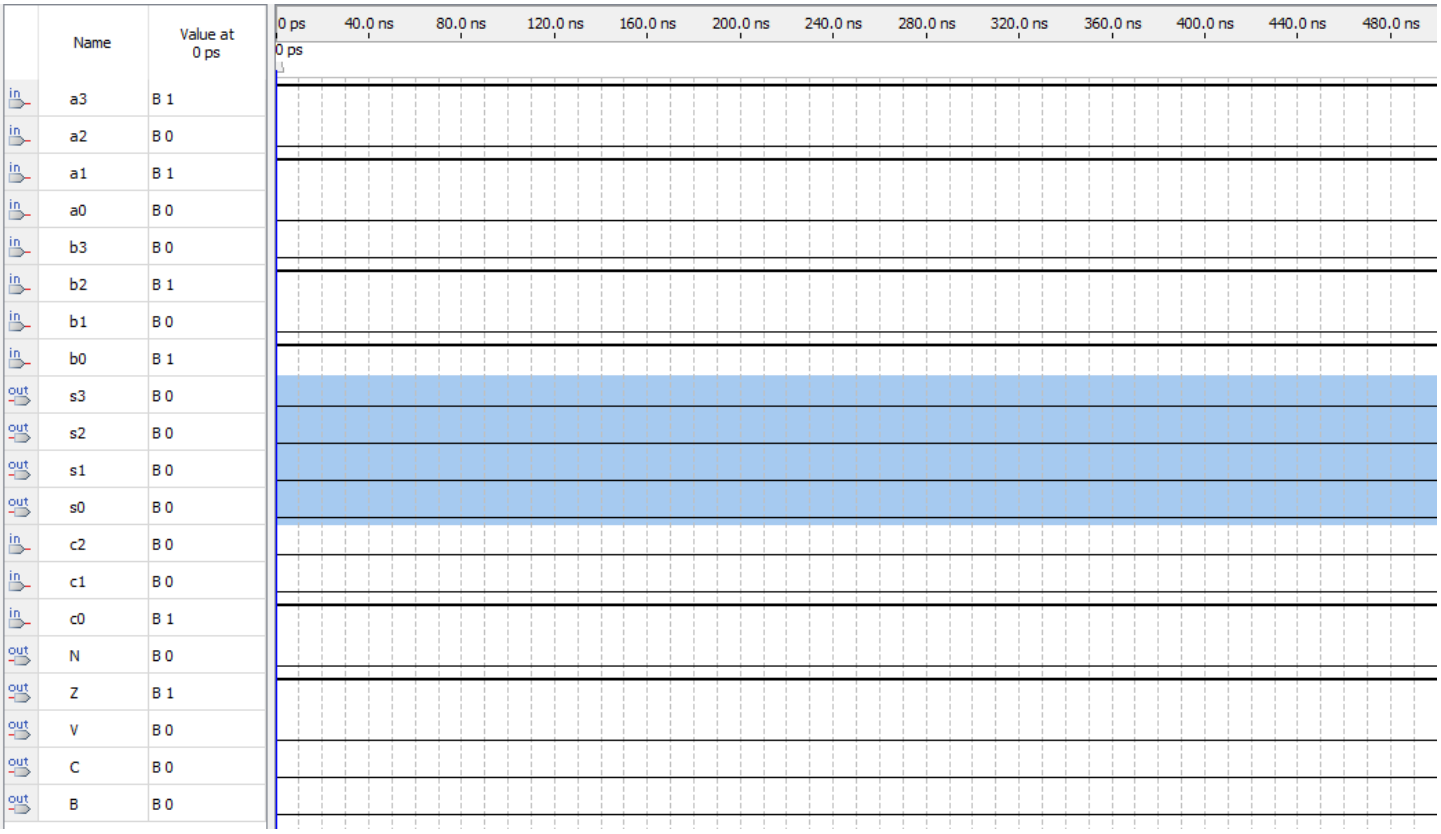
3 - SIMULAÇÕES DA ULA

As formas de onda que correspondem aos bits da saída serão demarcadas em azul, em cada simulação.

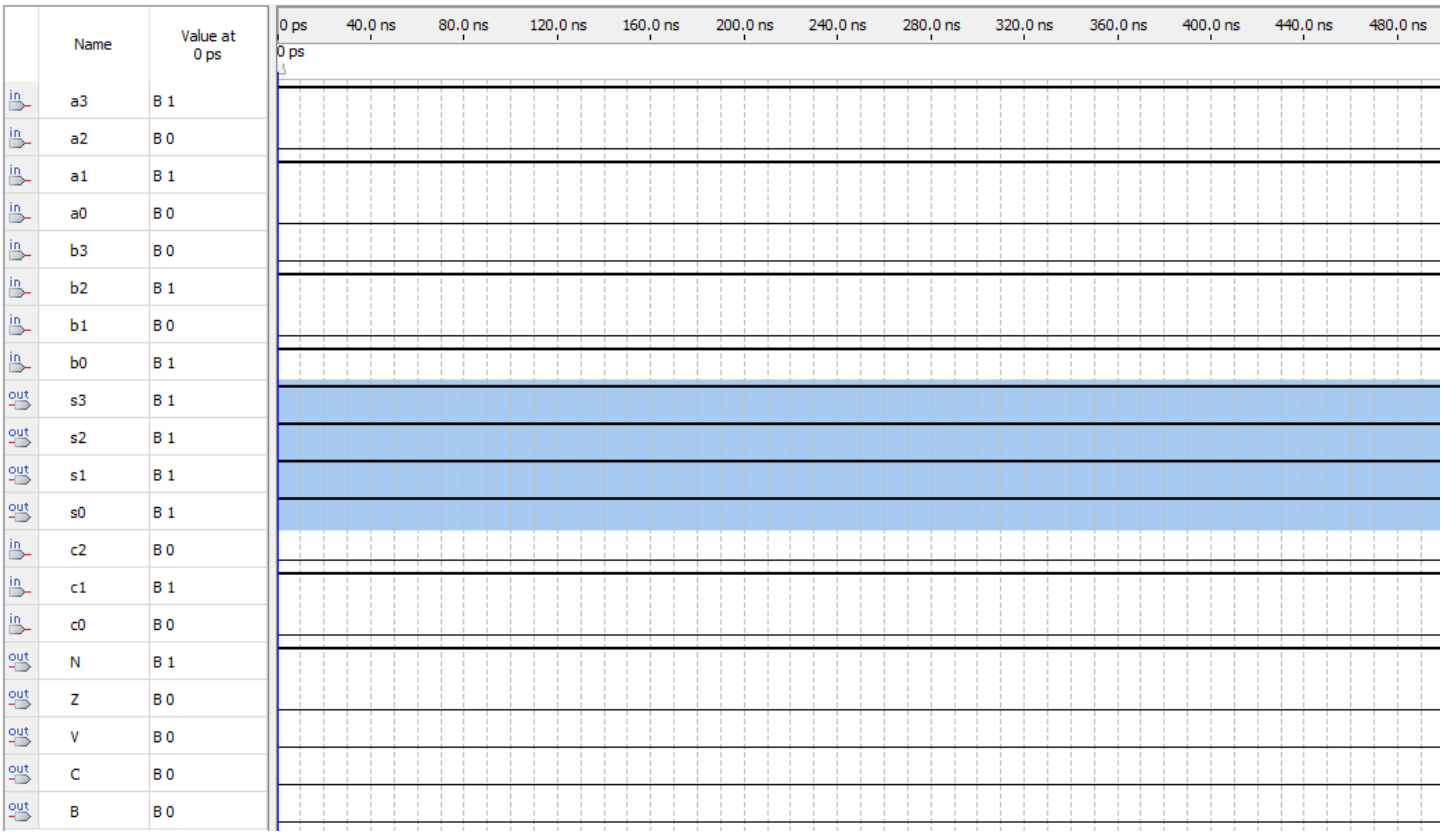
Simulação do PASSA B: abaixo, visualizamos um exemplo de simulação no qual a entrada A corresponde a 1010 e a entrada B corresponde a 0101. Com os bits de controle em 000, a saída final possui valor 0101 (mesmo valor da entrada B).



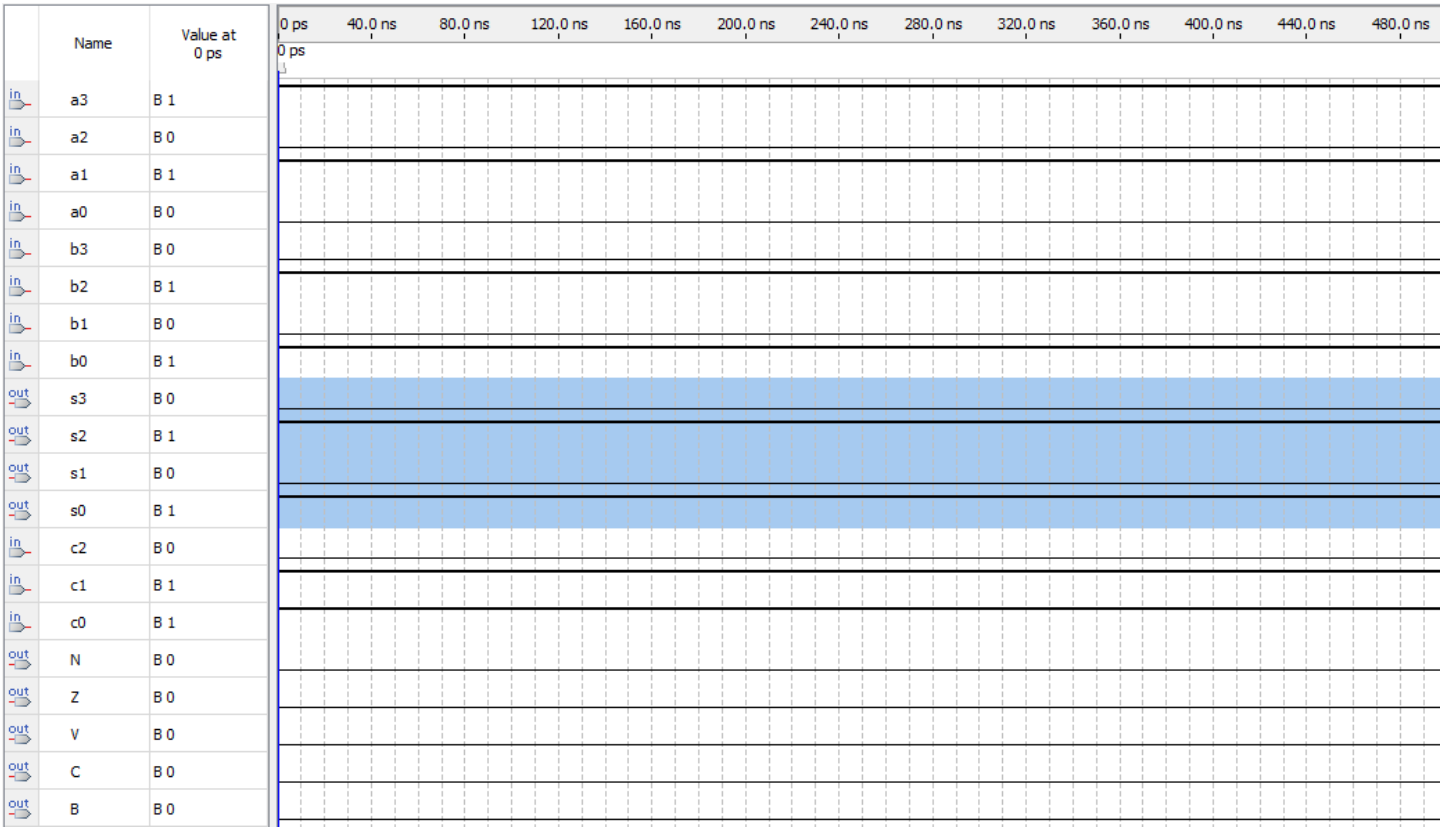
Simulação do A AND B: abaixo, visualizamos um exemplo de simulação no qual a entrada A corresponde a 1010 e a entrada B corresponde a 0101. Com os bits de controle em 001, a saída final possui valor 0000, após a aplicação da operação AND entre cada bit dos operandos. Nota-se que a flag Z foi acionada, após a operação.



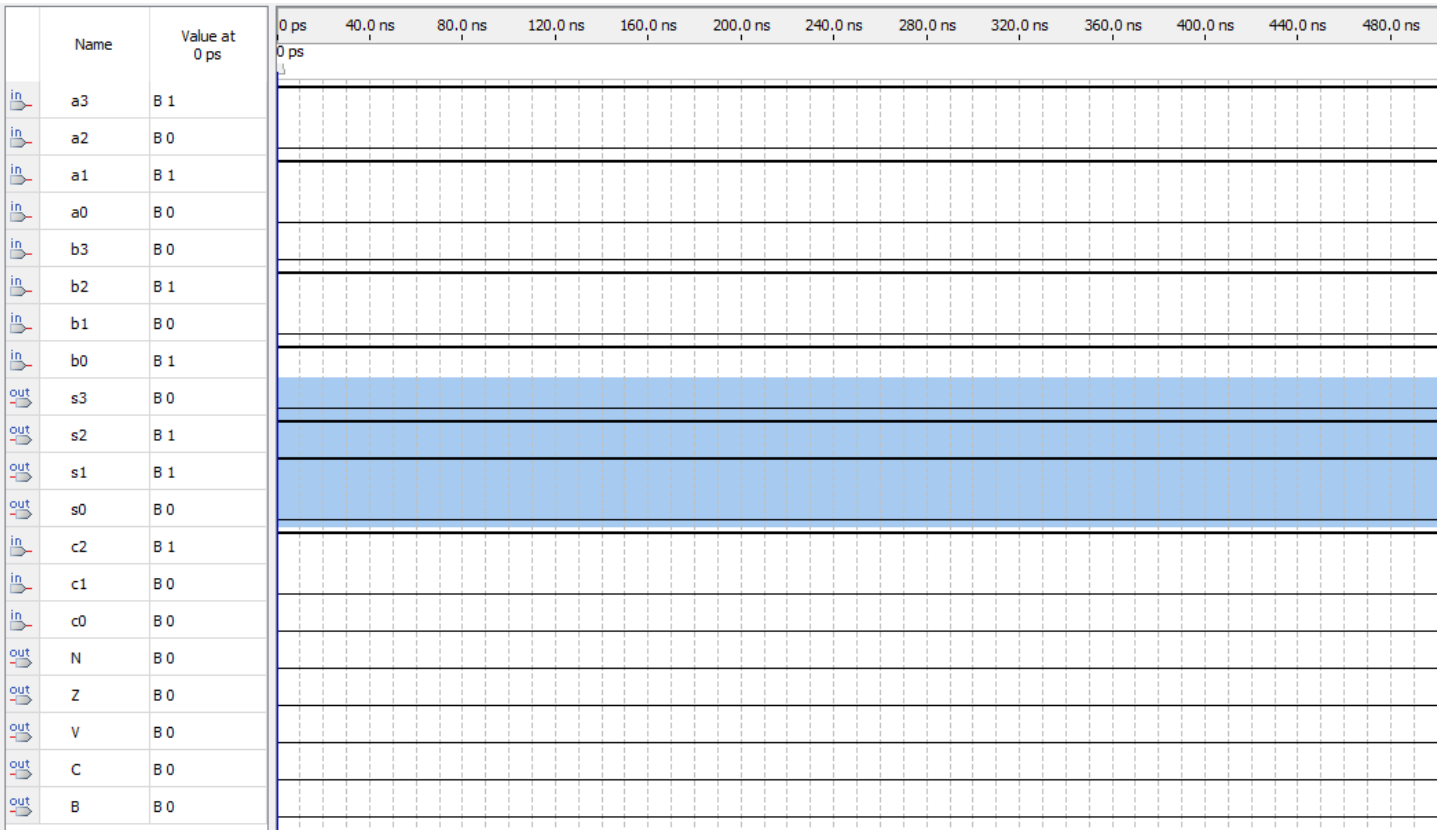
Simulação do A OR B: abaixo, visualizamos um exemplo de simulação no qual a entrada A corresponde a 1010 e a entrada B corresponde a 0101. Com os bits de controle em 010, a saída final possui valor 1111, após a aplicação da operação OR entre cada bit dos operandos. Nota-se que a flag N foi acionada, após a operação.



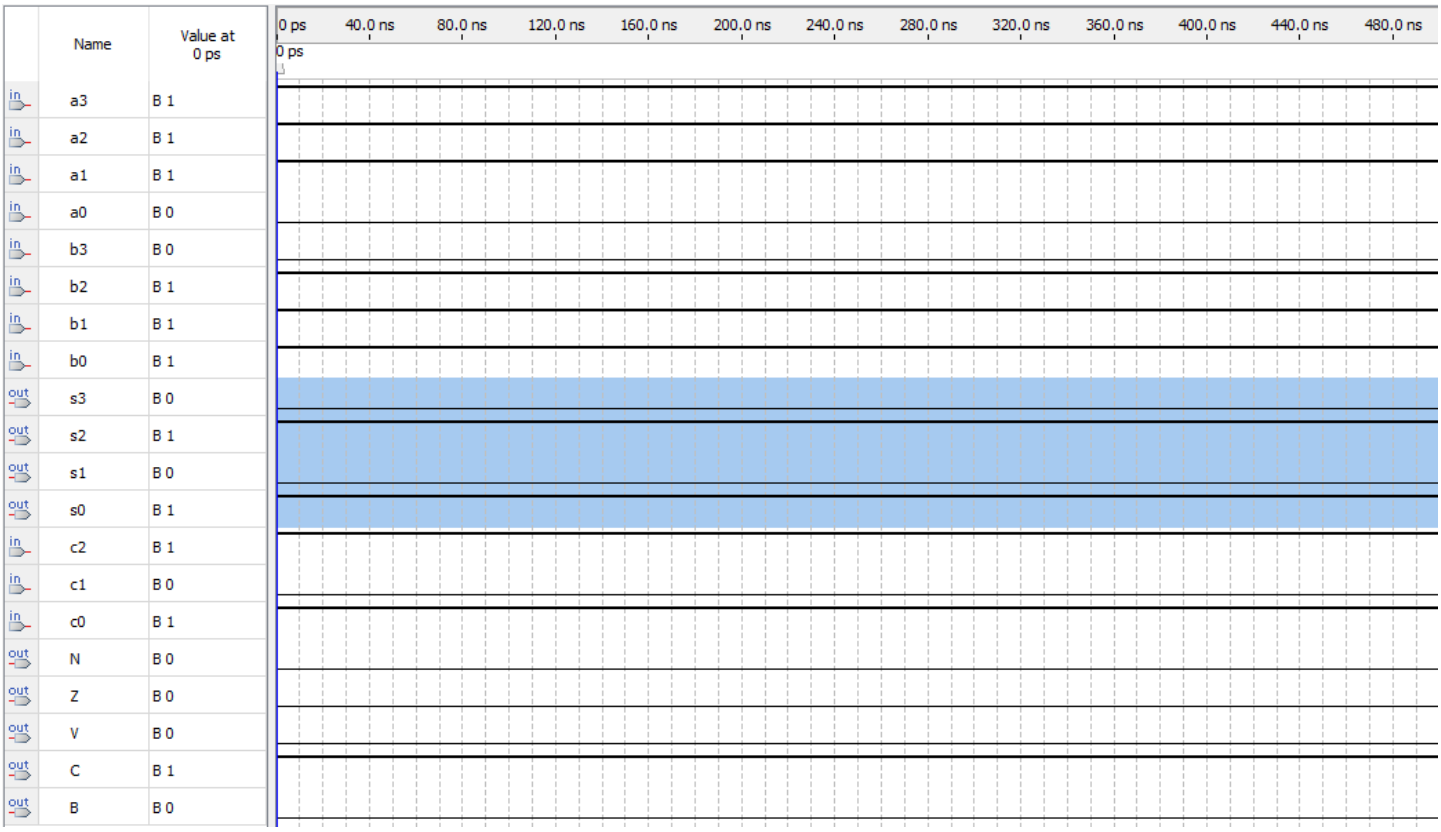
Simulação do NOT A: abaixo, visualizamos um exemplo de simulação no qual a entrada A corresponde a 1010. Com os bits de controle em 011, a saída final possui valor 0101, após a aplicação da operação NOT em cada bit do operando A.



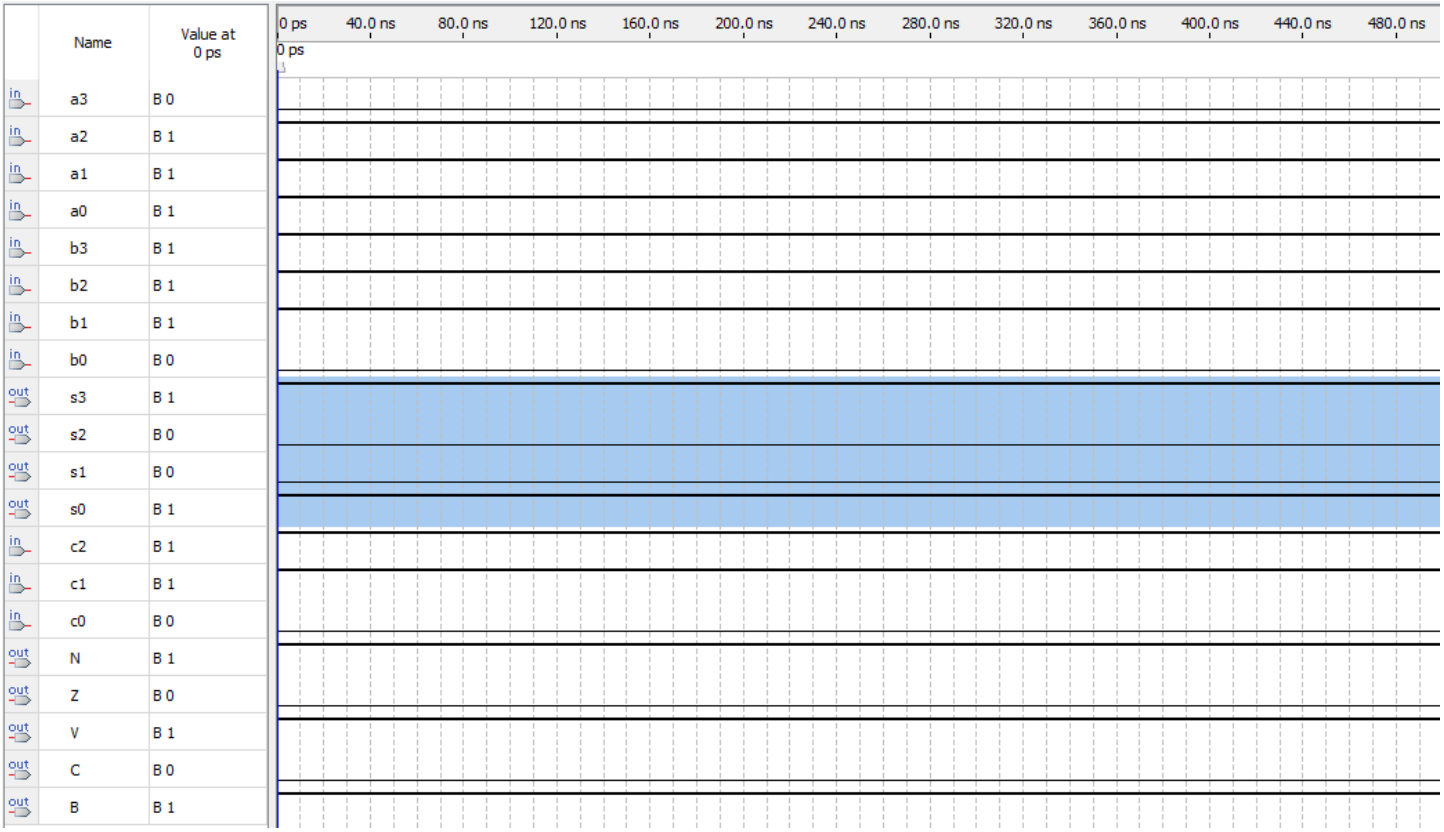
Simulação do NEG A: abaixo, visualizamos um exemplo de simulação no qual a entrada A corresponde a 1010. Com os bits de controle em 100, a saída final possui valor 0110, após a aplicação da operação NOT em cada bit do operando A e a soma do operando com 0001.



Simulação do ADD (A + B): abaixo, visualizamos um exemplo de simulação no qual a entrada A corresponde a 1110 e a entrada B corresponde a 0111. Com os bits de controle em 101, a saída final possui valor 0101, após a aplicação da operação. Nota-se que a flag C foi acionada, após a operação.



Simulação do SUB (A - B): abaixo, visualizamos um exemplo de simulação no qual a entrada A corresponde a 0111 e a entrada B corresponde a 1110. Com os bits de controle em 110, a saída final possui valor 1001, após a aplicação da operação. Nota-se que as flags N, V e B foram acionadas, após a operação.



Simulação do SHL A: abaixo, visualizamos um exemplo de simulação no qual a entrada A corresponde a 0111. Com os bits de controle em 111, a saída final possui valor 1110, após a aplicação da operação. Nota-se que a flag N foi acionada, após a operação.