TRABALHO TEÓRICO-PRÁTICO Etapa 1: Circuitos Digitais

Yaçanã Isis L. J. da Silveira[1], Yan B. da Silva[1]

[1] Universidade Federal de Ciências da Saúde de Porto Alegre (UFCSPA)
R. Sarmento Leite, 245 – 90050-170 – Porto Alegre – RS – Brasil

yacana.silveira@ufcspa.edu.br, yan.silva@ufcspa.edu.br

Abstract. The present work aims to present the responses obtained in the theoretical-practical work on digital circuits in the Computer Architecture course of the Biomedical Informatics Bachelor's program.

Resumo. O presente trabalho visa apresentar as respostas obtidas no trabalho teórico-prático sobre circuitos digitais da disciplina de Arquitetura de Computadores do Bacharelado em Informática Biomédica.

1. Introdução

O presente trabalho visa sanar as explicações e elaborações necessárias para corretamente responder às questões impostas pelo Docente Regente da disciplina. Além disso, usamos de apoio visual de circuitos criados a partir do *software* Logisim para a melhor compreensão dos resultados aqui expostos.

2. Questão 1

Circuito multiplexador 8x1 criado a partir de exemplo, onde d[0], d[1], d[2], d[3], d[4], d[5], d[6] e d[7] são as entradas, q é o sinal de saída e A, B e C são os sinais de seleção do sinal de entrada. Para transformar o exemplo 4x1 para 8x1 foi necessária a inclusão de mais 4 entradas, 1 sinal de seleção do sinal de entrada e 4 portas lógicas do tipo AND para que todas as possibilidades pudessem ser testadas efetivamente.

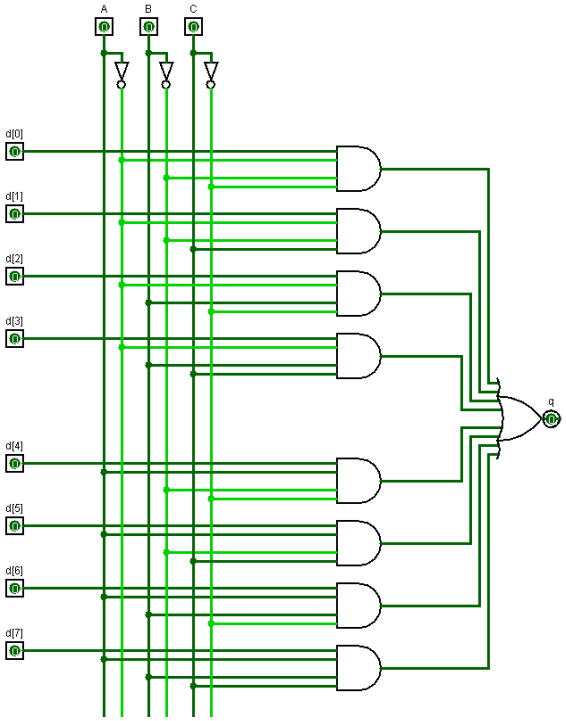

Figura 1. Circuito Multiplexador 8x1

3. Questão 2

A questão apresentava um registrador de escrita paralela de 4 bits e solicitava, a partir do exemplo, a implementação de um circuito similar contendo a alteração de 4 para 6 bits, adição de botões liga/desliga para os sinais de entrada, LEDs para os sinais de saída botão push-button para o relógio (Clock).

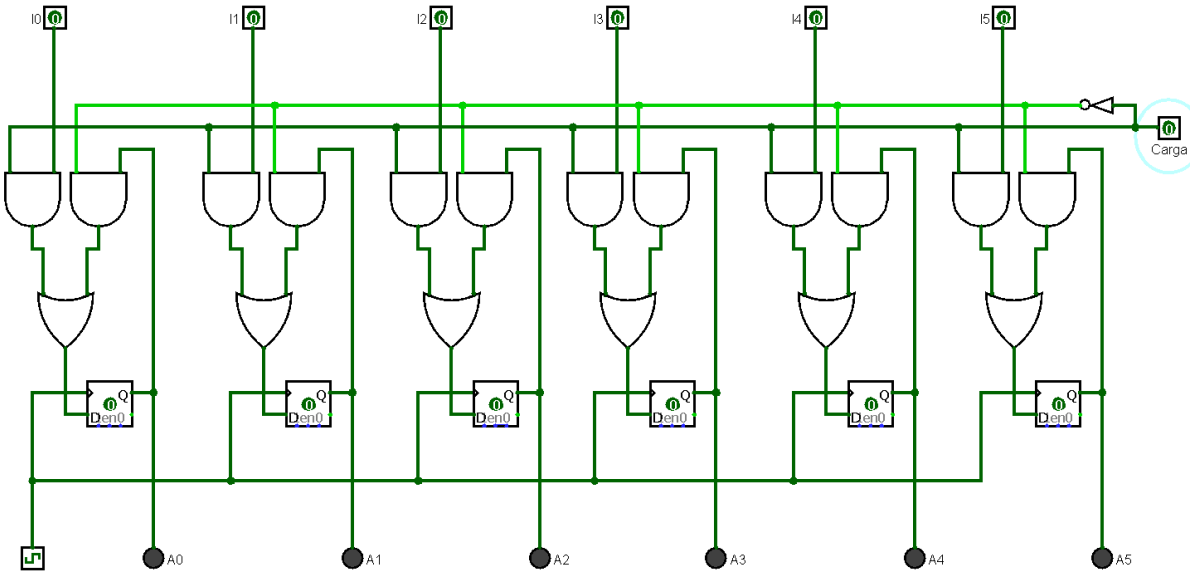

Figura 2. Registrador de escrita paralela de 6 bits

Após a implementação das modificações, o circuito foi testado para seguintes combinações de bits: 011001, 111000, 110011, 101010.

3.1 Combinação 011001

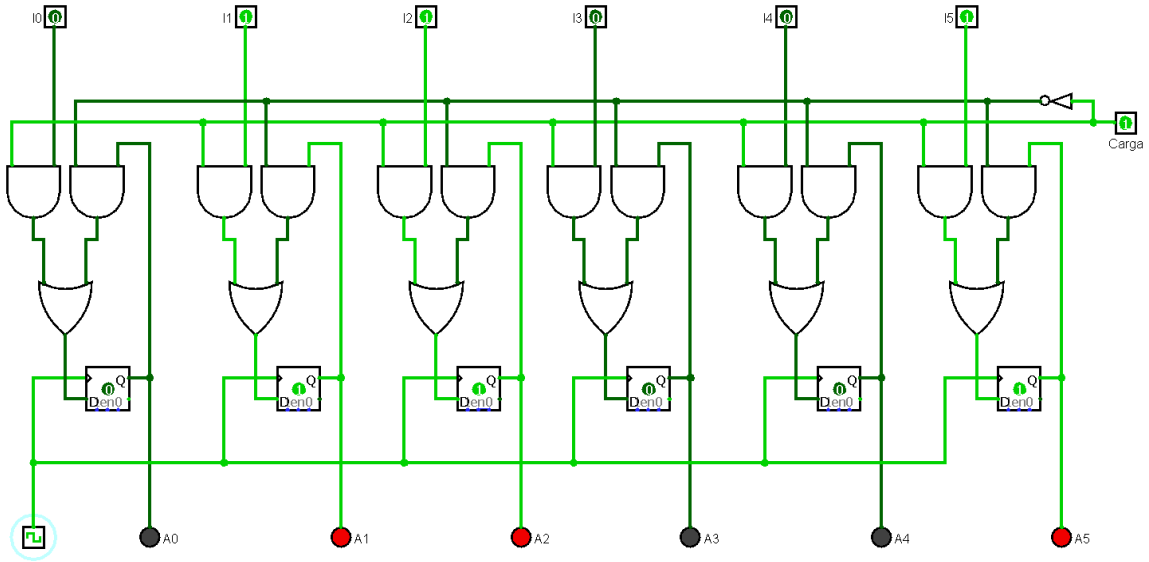

Figura 3. *Clock* e carga ligados transmitindo a informação para os registradores

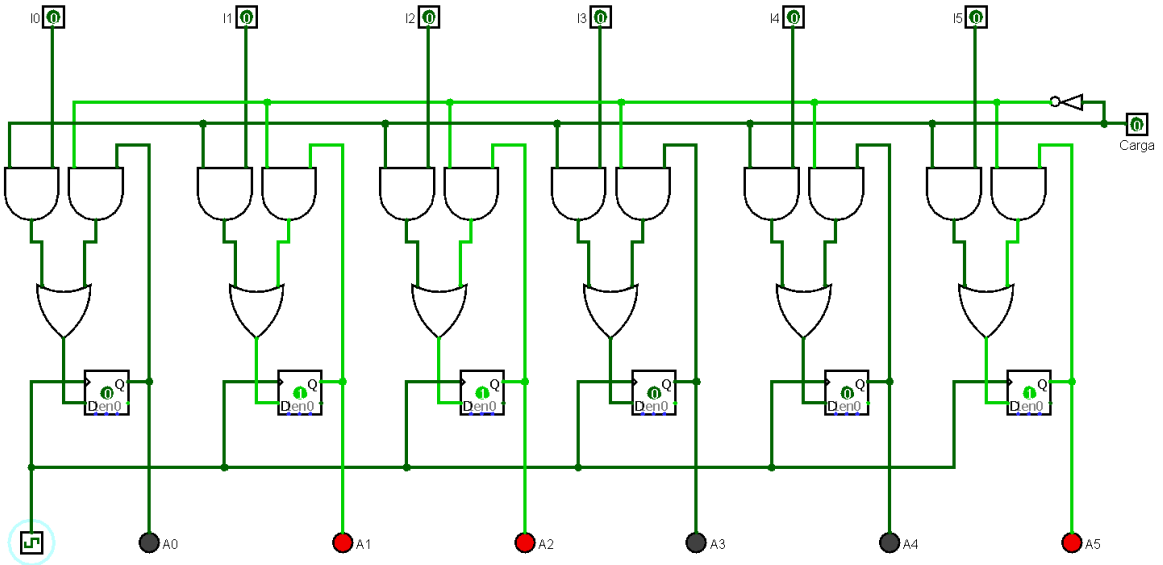

Figura 4. *Clock* e carga desligados, informação armazenada nos registradores

3.2 Combinação 111000

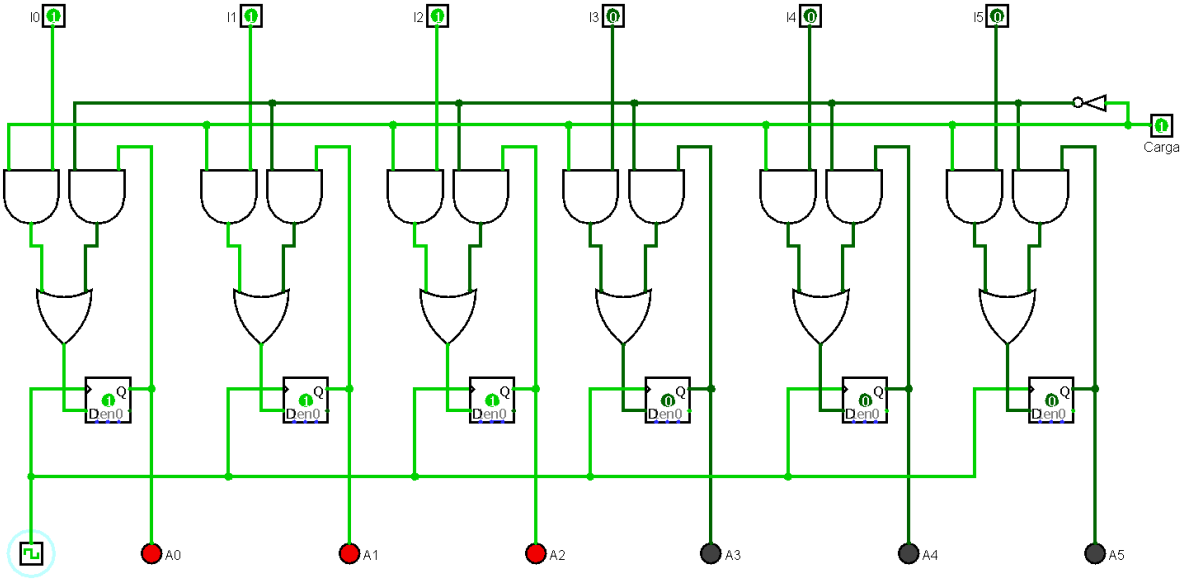

Figura 5. *Clock* e carga ligados transmitindo a informação para os registradores

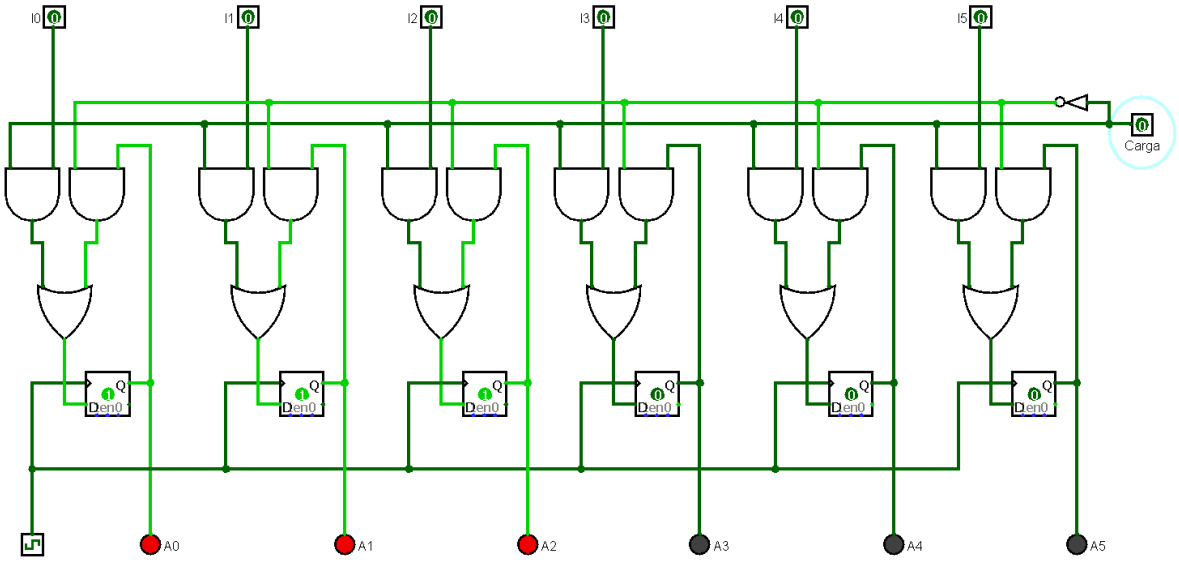

Figura 6. *Clock* e carga desligados, informação armazenada nos registradores

4.3 Combinação 110011

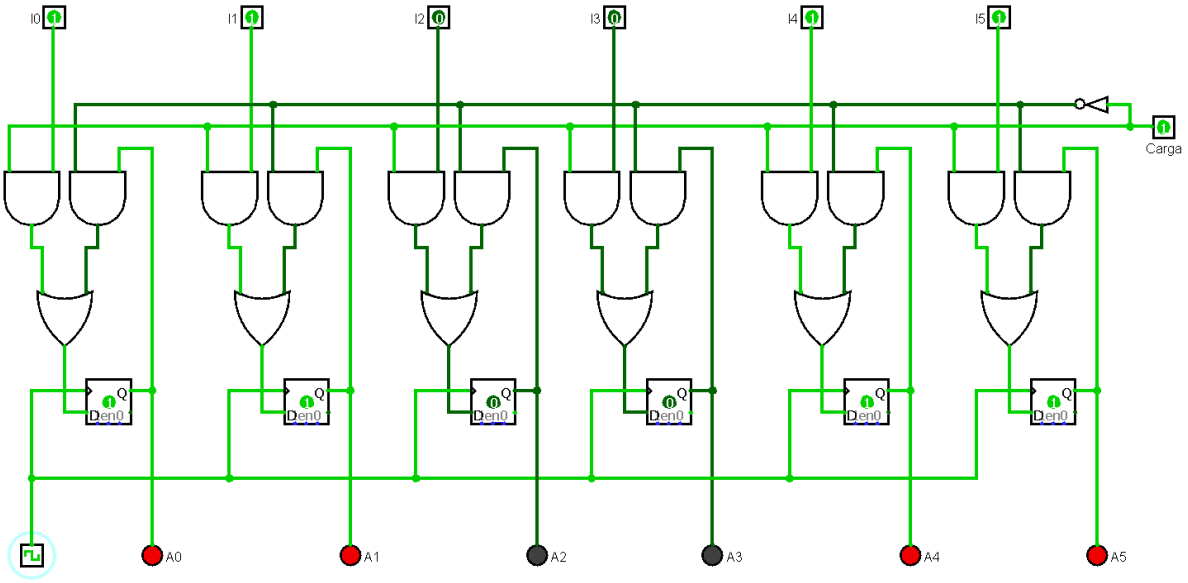

Figura 7. *Clock* e carga ligados transmitindo a informação para os registradores

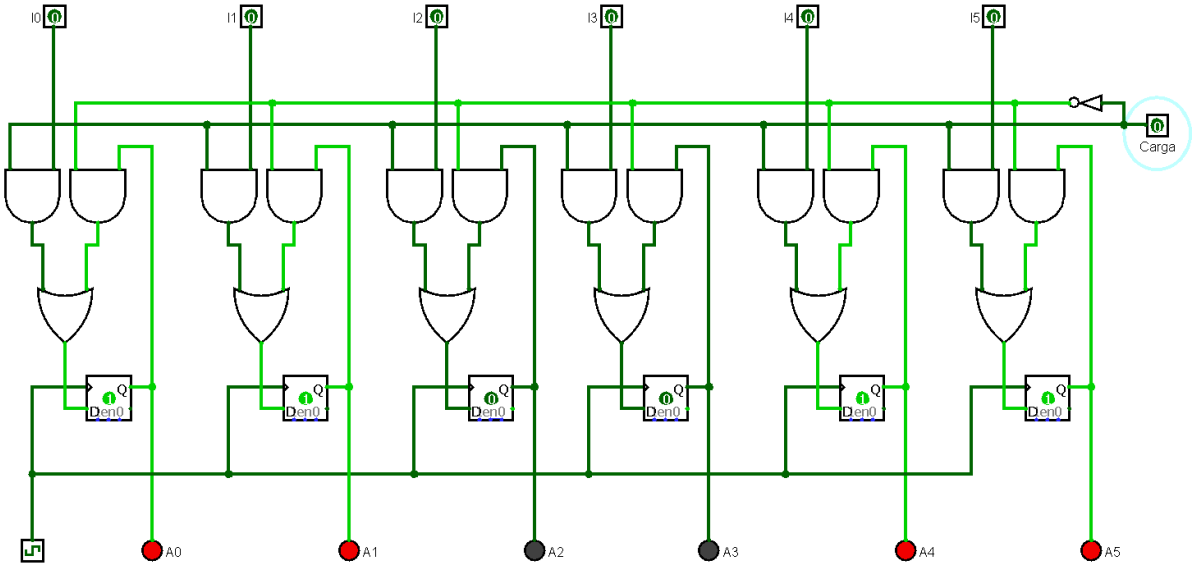

Figura 8. *Clock* e carga desligados, informação armazenada nos registradores

4.3 Combinação 101010

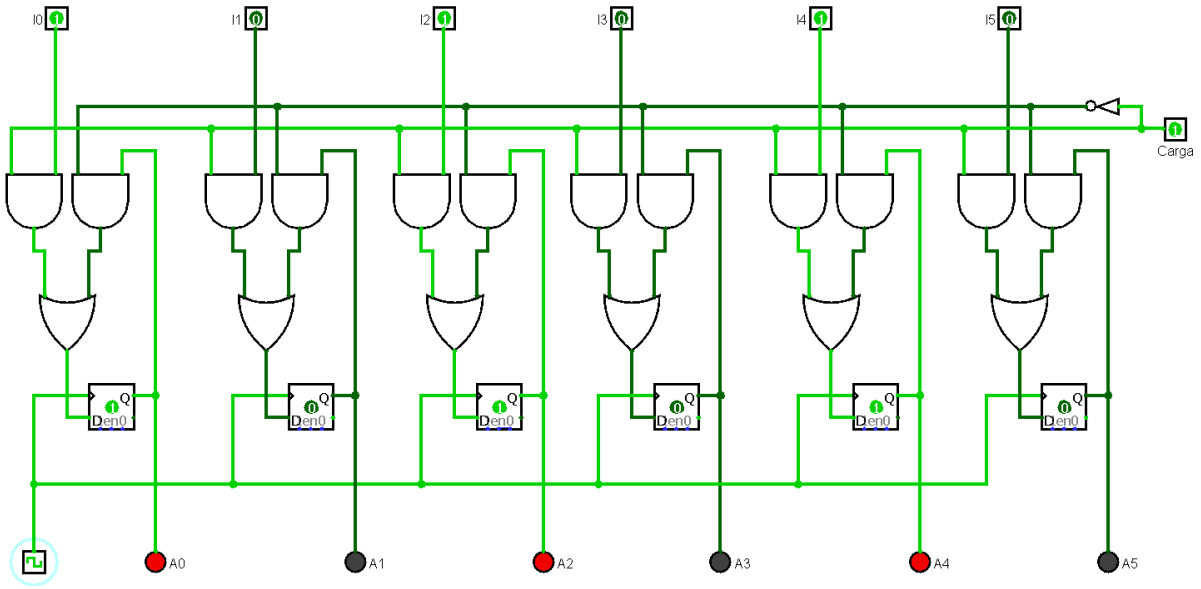

Figura 9. *Clock* e carga ligados transmitindo a informação para os registradores

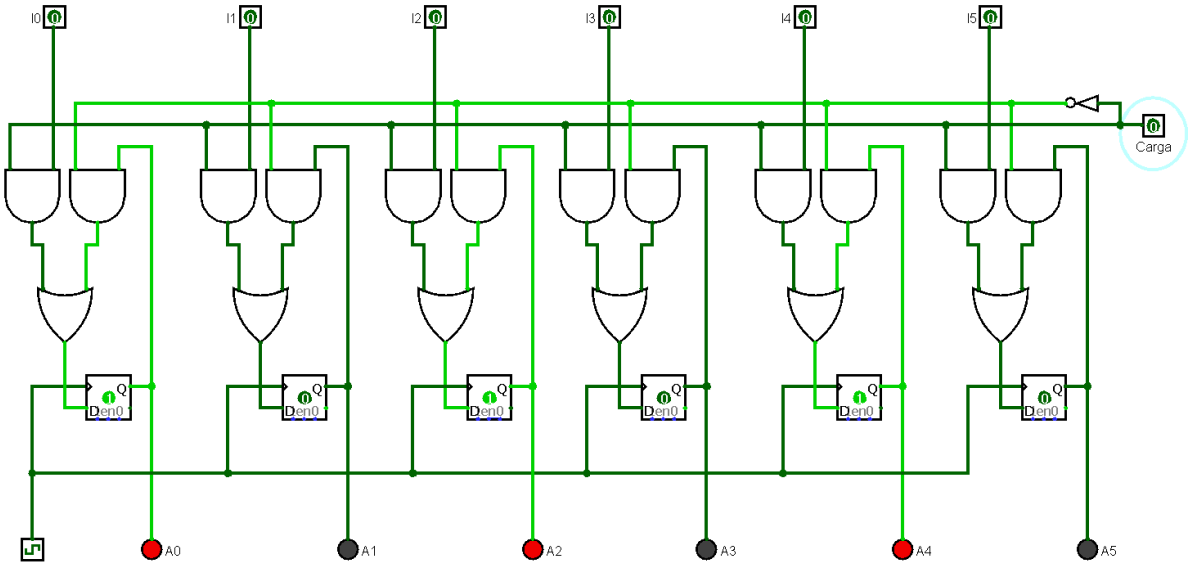

Figura 10. *Clock* e carga desligados, informação armazenada nos registradores

4. Questão 3

Essa questão é elaborada sobre a perspectiva do conhecimento das operações de deslocamento de bits dos registradores das CPUs, fundamentais para operações advindas da própria unidade de processamento. Usando flip-flops alinhados e sequenciais, juntamente de um clock paralelo a todos eles, é possível verificar a informação em forma de bits 0 ou 1 sendo transferidos sequencialmente durante o processo do registrador de deslocamento de 6 bits. O valor testado foi: 01101011

A entrada do valor é feita a cada *tick* do *clock*, sendo apresentado da esquerda para a direita. E a saída serial é representada por meio do último conector que sai à direita do

diagrama.

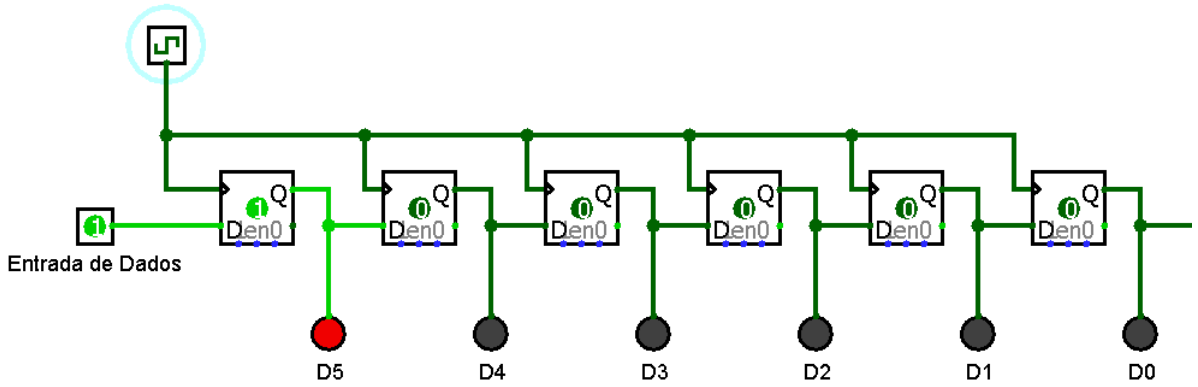

Figura 11. Primeiro bit de entrada transmitido para o registrador

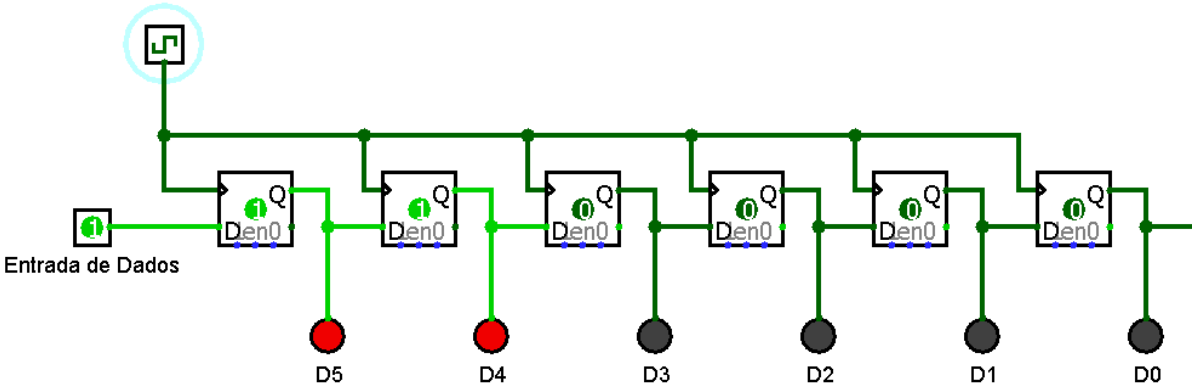

Figura 12. Segundo bit de entrada transmitido para o registrador

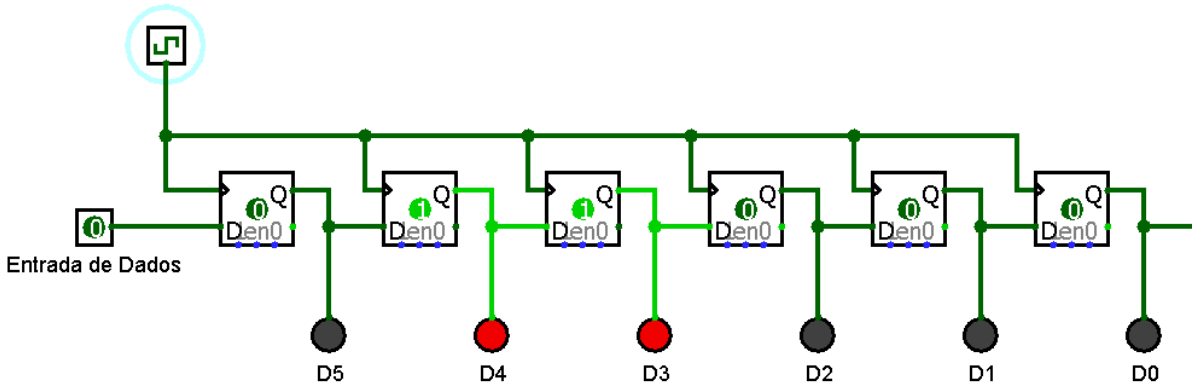

Figura 13. Terceiro bit de entrada transmitido para o registrador

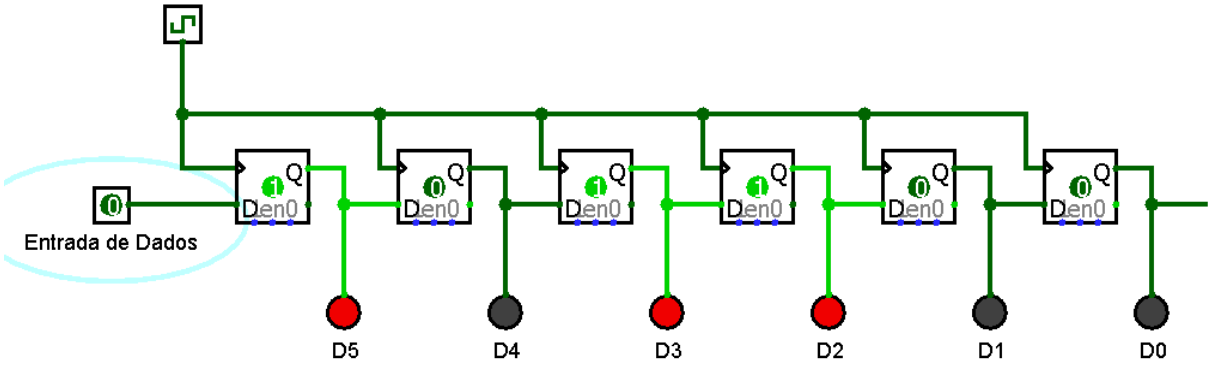

Figura 14. Quarto bit de entrada transmitido para o registrador

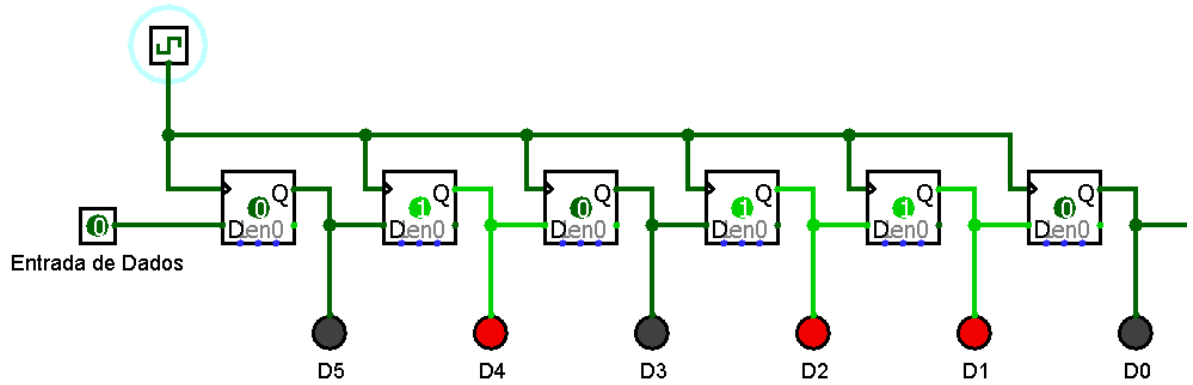

Figura 15. Quinto bit de entrada transmitido para o registrador

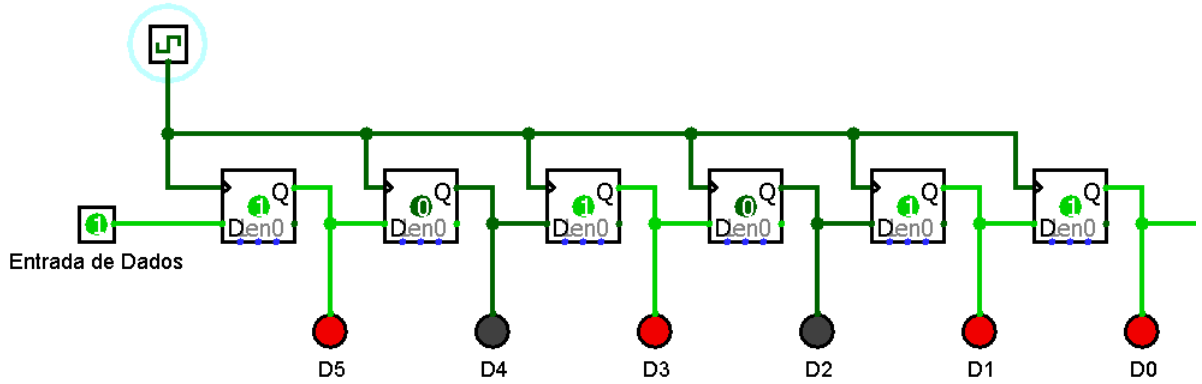

Figura 16. Sexto bit de entrada transmitido para o registrador

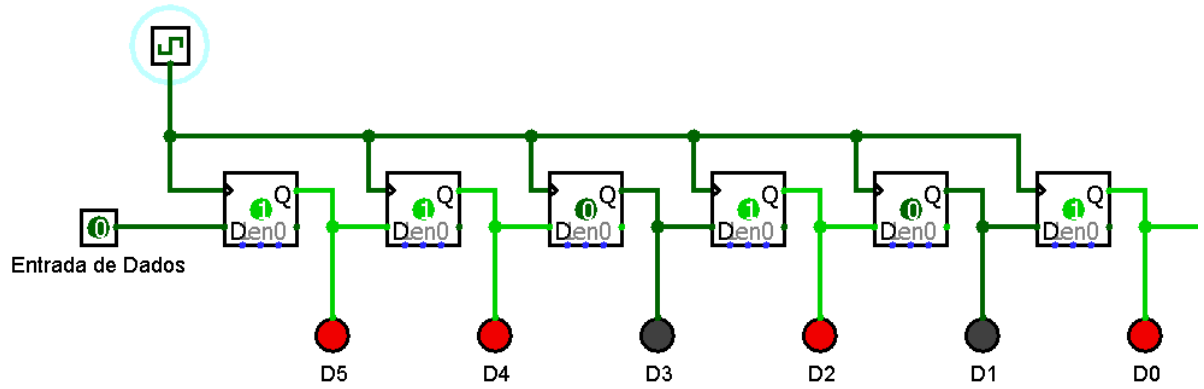

Figura 17. Sétimo bit de entrada transmitido para o registrador

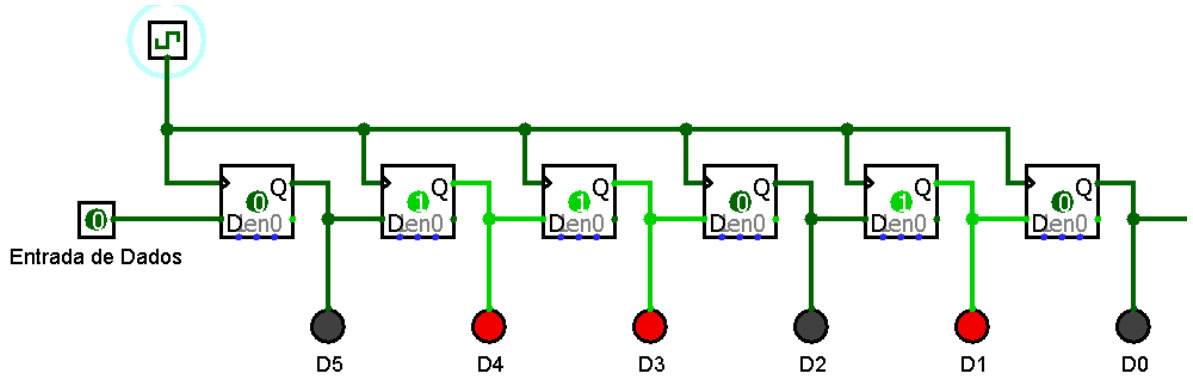

Figura 18. Oitavo bit de entrada transmitido para o registrador

5. Questão 4

Essa questão aborda a implementação das operações aritméticas mais elementares, soma e subtração de bits em uma CPU. Isso é realizado pela Unidade Aritmético-Lógica, que para sua implementação foram usadas as portas AND, OR, XOR, e NOT para o correto funcionamento. Sendo X[0], X[1], X[2], X[3] as entradas que representam os 4 bits do primeiro número, Y[0], Y[1], Y[2], Y[3] as entradas que compõem o segundo número, S[0], S[1], S[2], S[3] as saídas que compõem o número resultante do cálculo.

O circuito foi implementado de tal forma que, quando os sinais de adição e subtração estiverem desligados, os dados de X são replicados em S respectivamente.

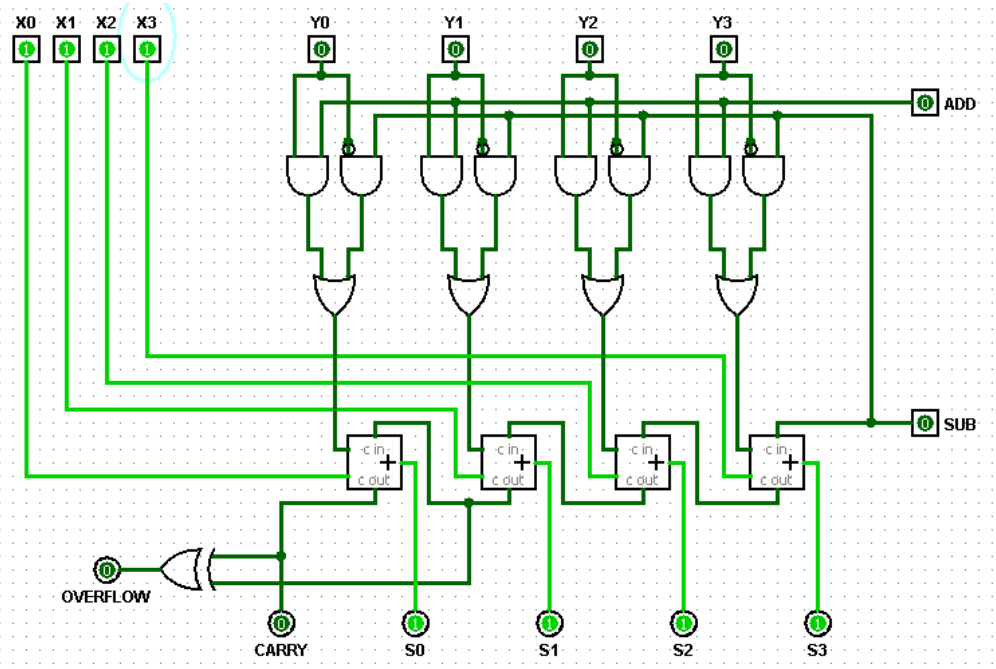

Figura 19. Dados replicados de X para S, sem alteração.

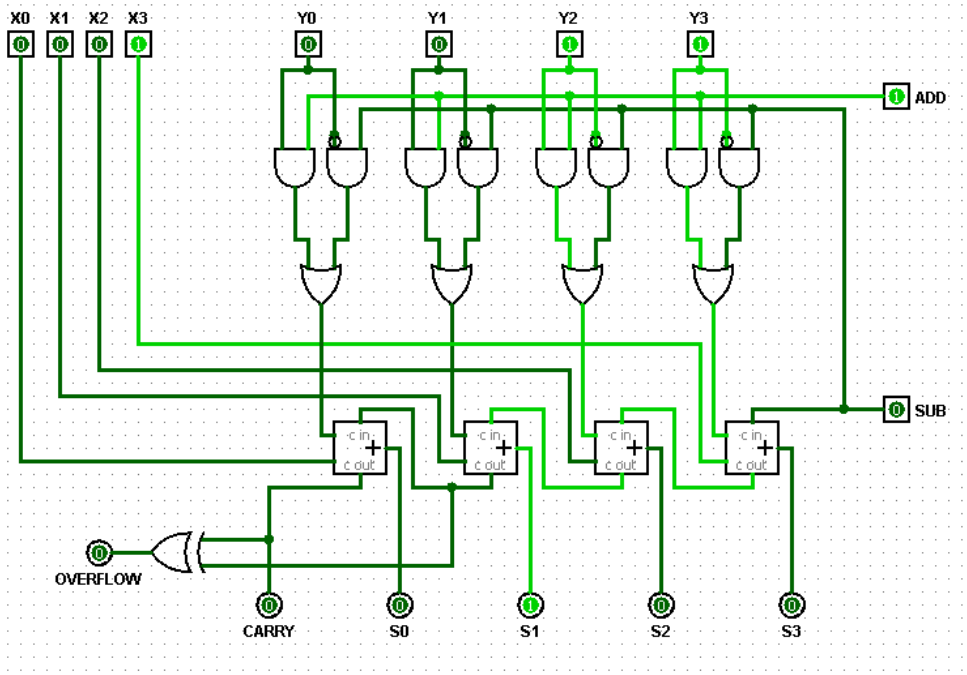

Figura 20. Cálculo (+1)+(+3), usando o operando adequado de soma (ADD).

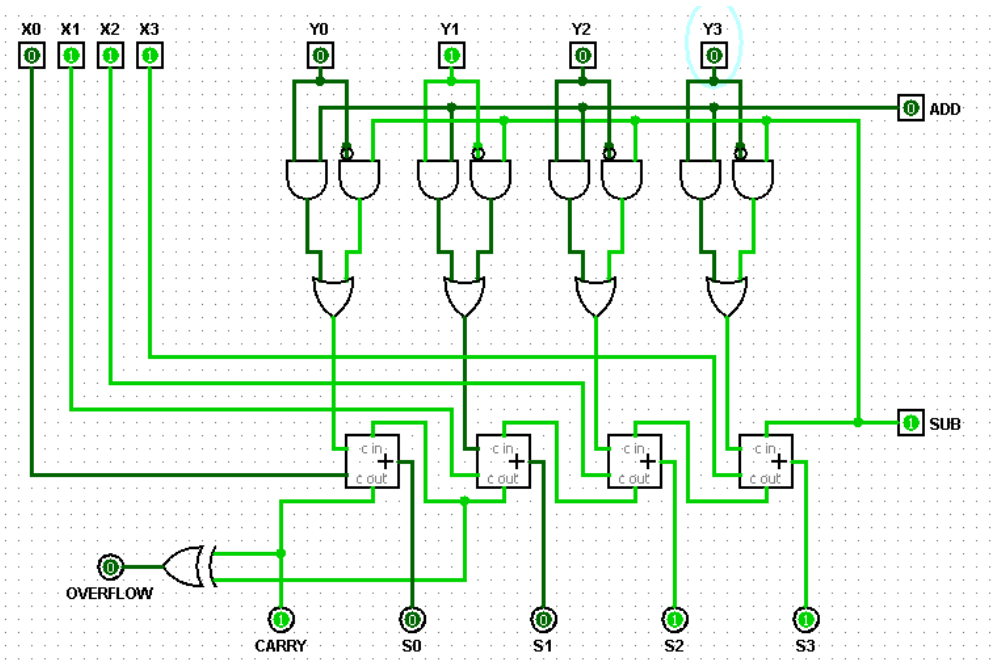

Figura 22. Cálculo (+7) - (+4), usando o operando adequado de subtração(SUB).

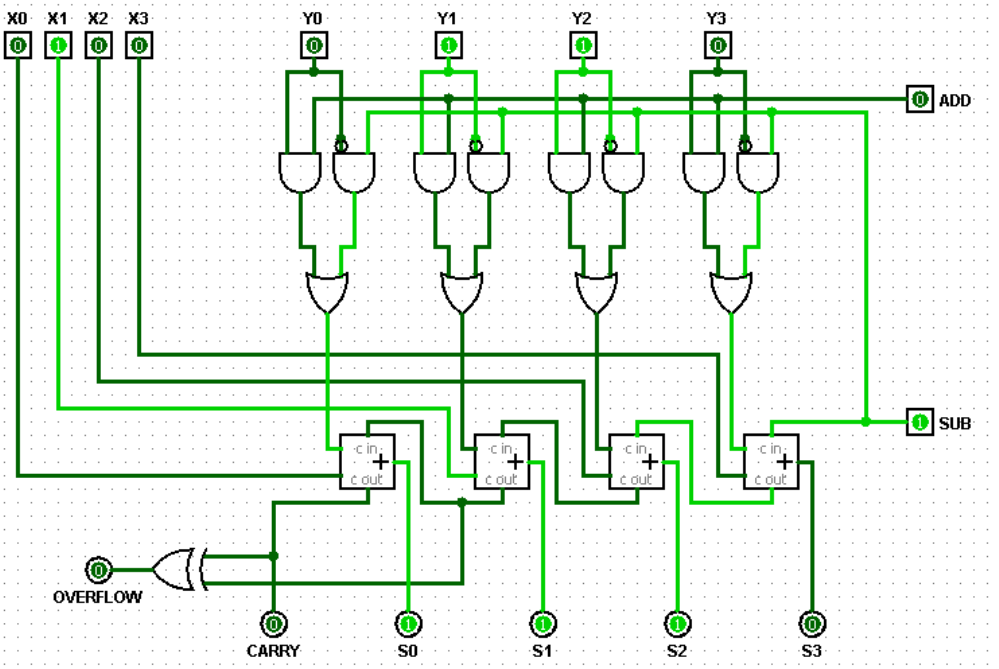

Figura 23. Cálculo (+4) - (+6), usando o operando adequado de subtração(SUB).

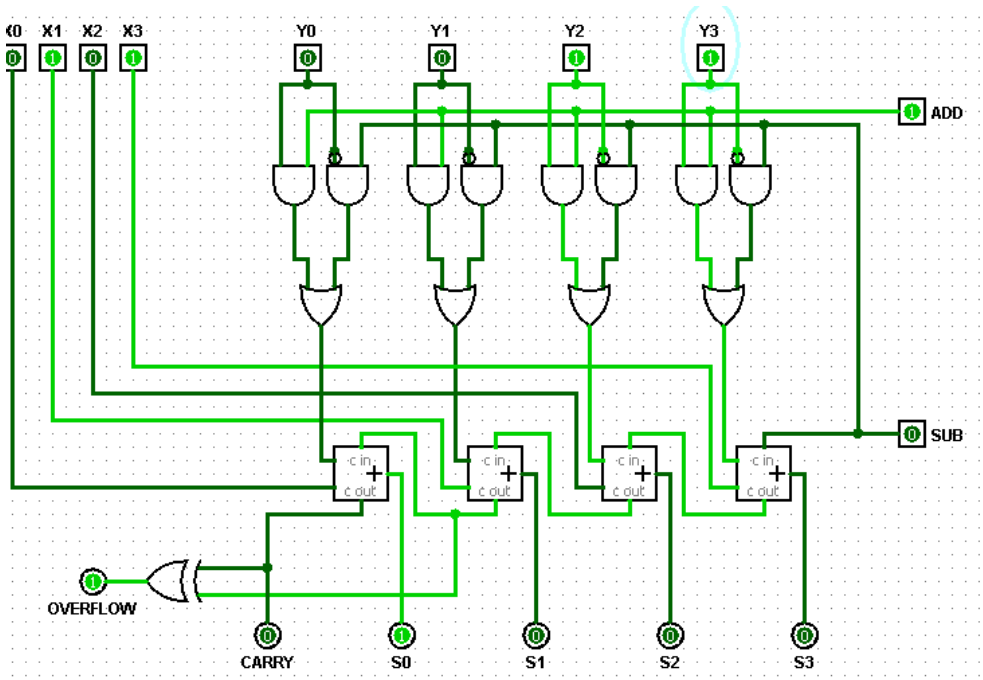

Figura 24. Cálculo (+5) + (+3), usando o operando adequado de soma (ADD).

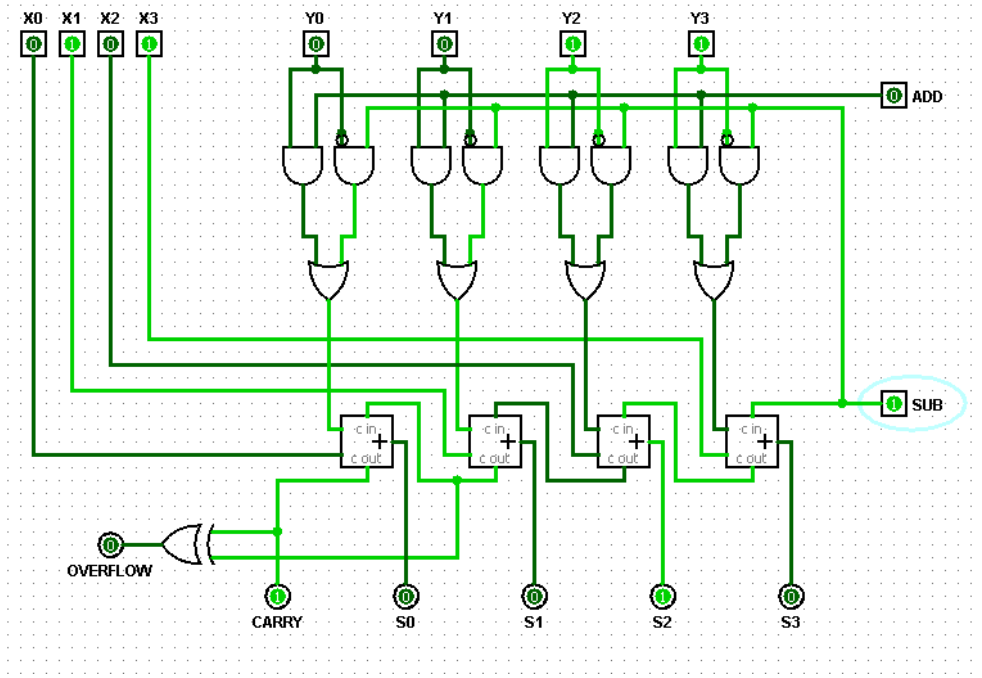

Figura 25. Cálculo (+5) - (+3), usando o operando adequado de subtração(SUB).

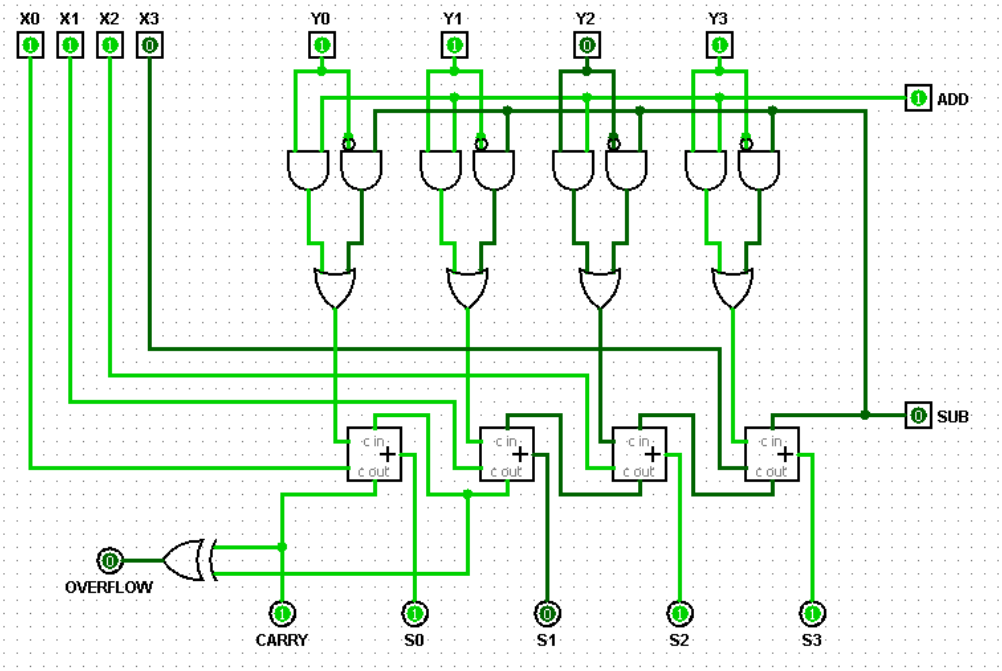

Figura 26. Cálculo (-2) + (-3) , usando o operando adequado de soma(ADD), além dos números terem sua representação em complemento de 2.

Para que a subtração ocorra, o sinal de entrada da subtração (SUB) é enviado para a sequência de portas AND que invertem os dados das entradas Y. O sinal de SUB também é ligado nas portas de Cin (*carry in*) do somador completo (*full-adder*), no bit menos significativo essa adição de 1 na porta Cin equivale a somar 1 bit à inversão dos bits de entrada, completando a regra do cálculo de soma e subtração binária em complemento de dois.

O circuito que detecta a ocorrência de *overflow* é feito a partir de uma porta lógica XOR. Comparando os dados Cout (*carry out*) dos dois circuitos aritméticos que dão origem aos bits mais significativos da saída de soma. Se, e somente se, esses bits forem diferentes, o circuito recebe 1 bit, indicando overflow. É possível observar o acionamento do circuito na figura 24.

6. Questão 5

Essa questão aborda as implementações elementares de uma ALU (Unidade Aritmética e Lógica) AND, OR e NOT, que são fundamentais para outras operações, como a soma e subtração de números, comparação de resultados e complementação. A seguir estão exemplificados os testes **A** AND **B**, **A** OR **B** e NOT **A**. Usamos os seguintes valores binários de A: 0010, 1011, 1100; e B: 0110, 1110, 1111; respectivamente em cada uma das operações pedidas, conforme seguem as figuras.

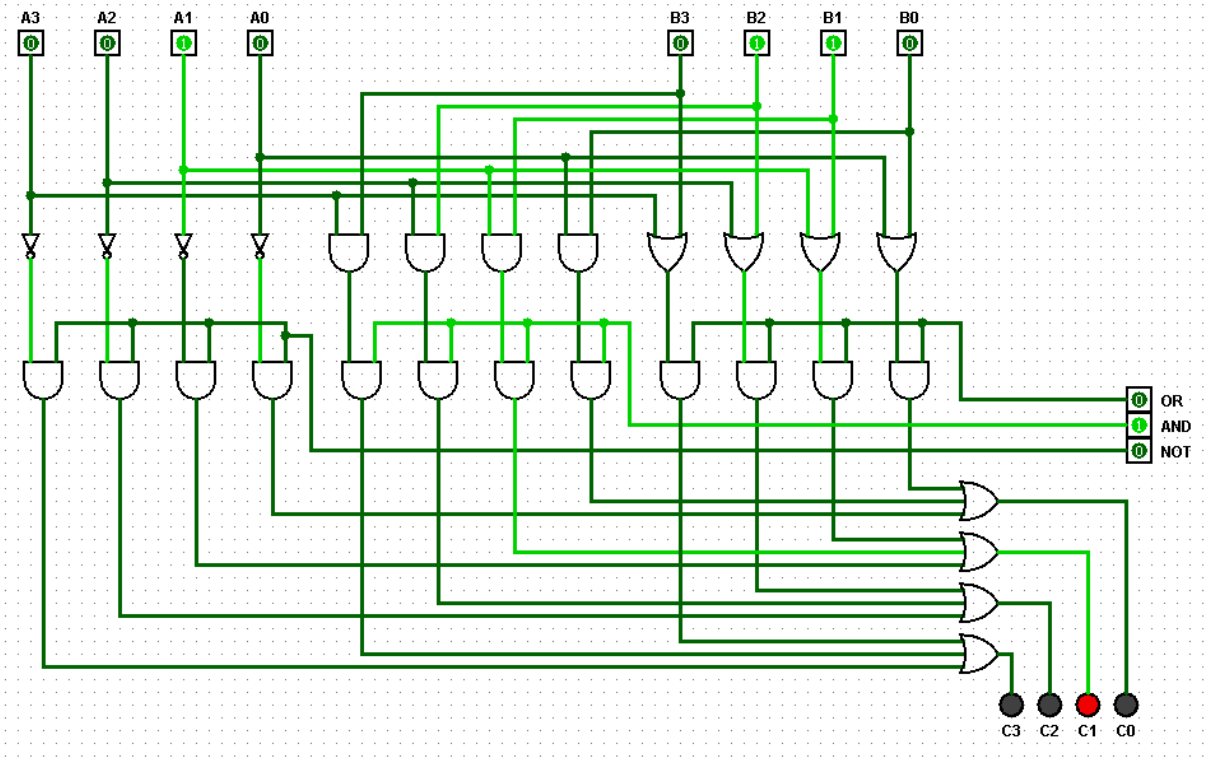

Figura 27. ALU com A (0010) AND B (0110).

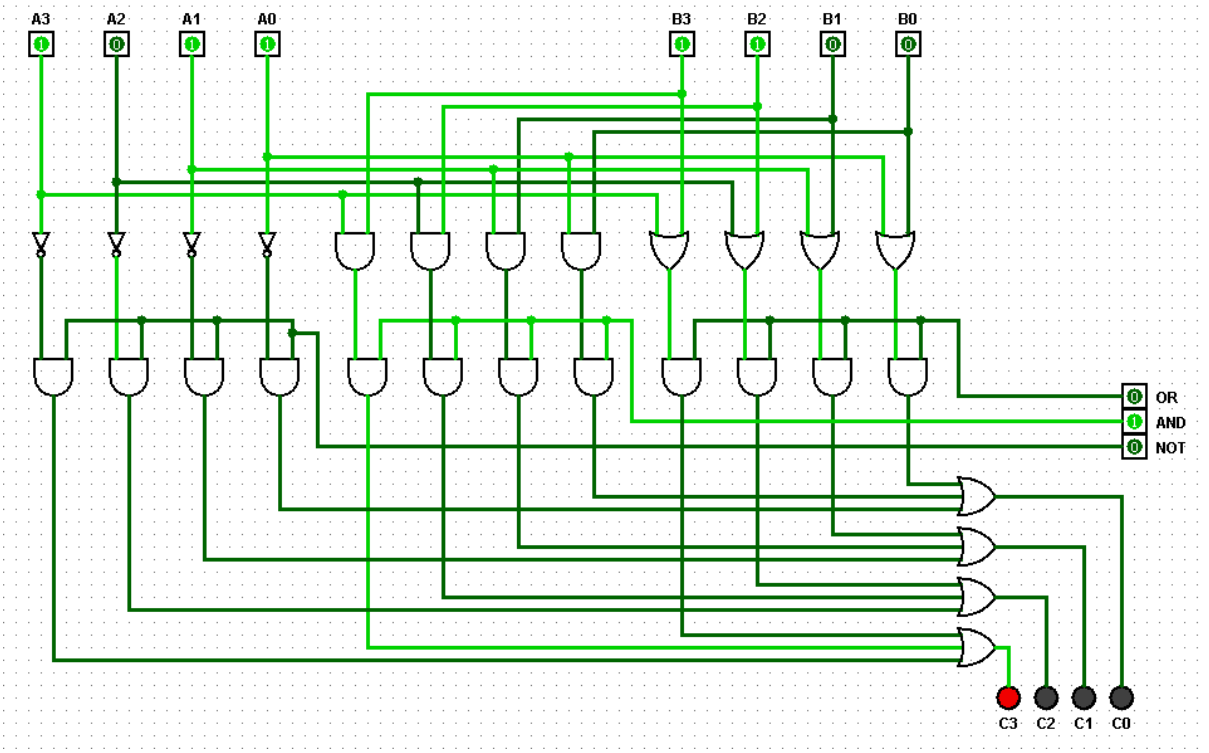

Figura 28. ALU com A (1011) AND B (1100).

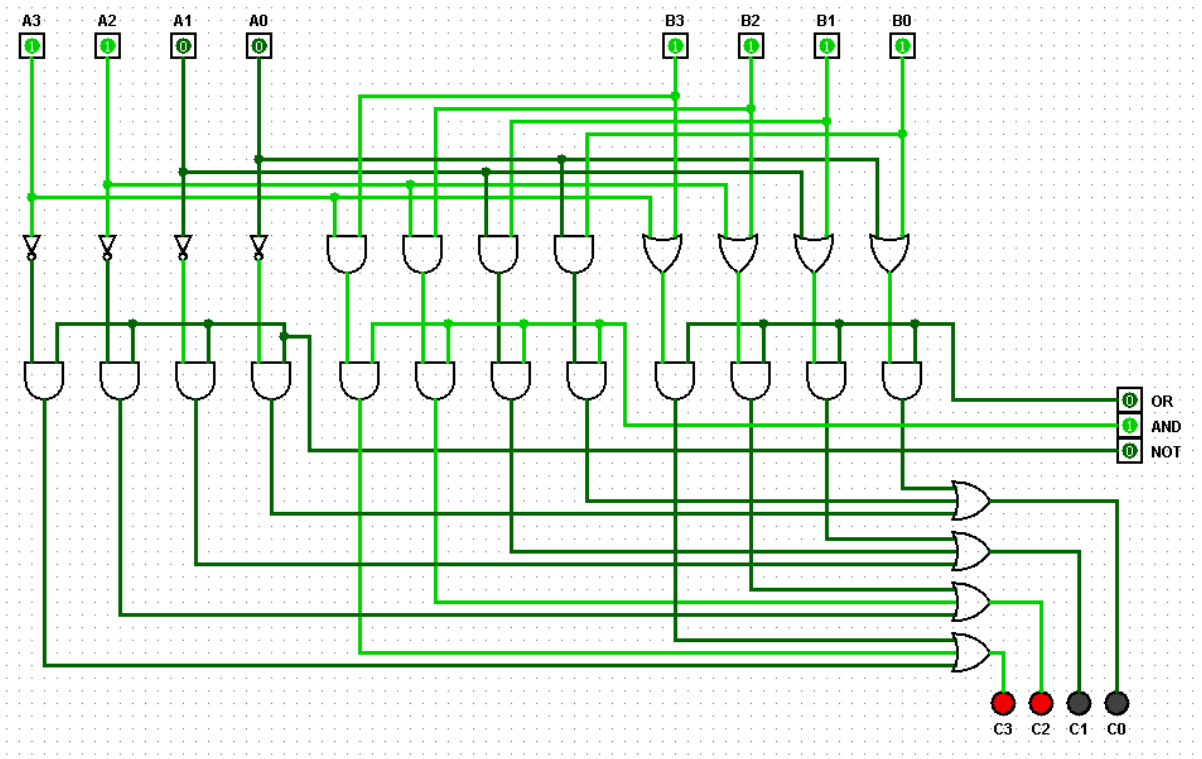

Figura 29. ALU com A (1100) AND B (1111).

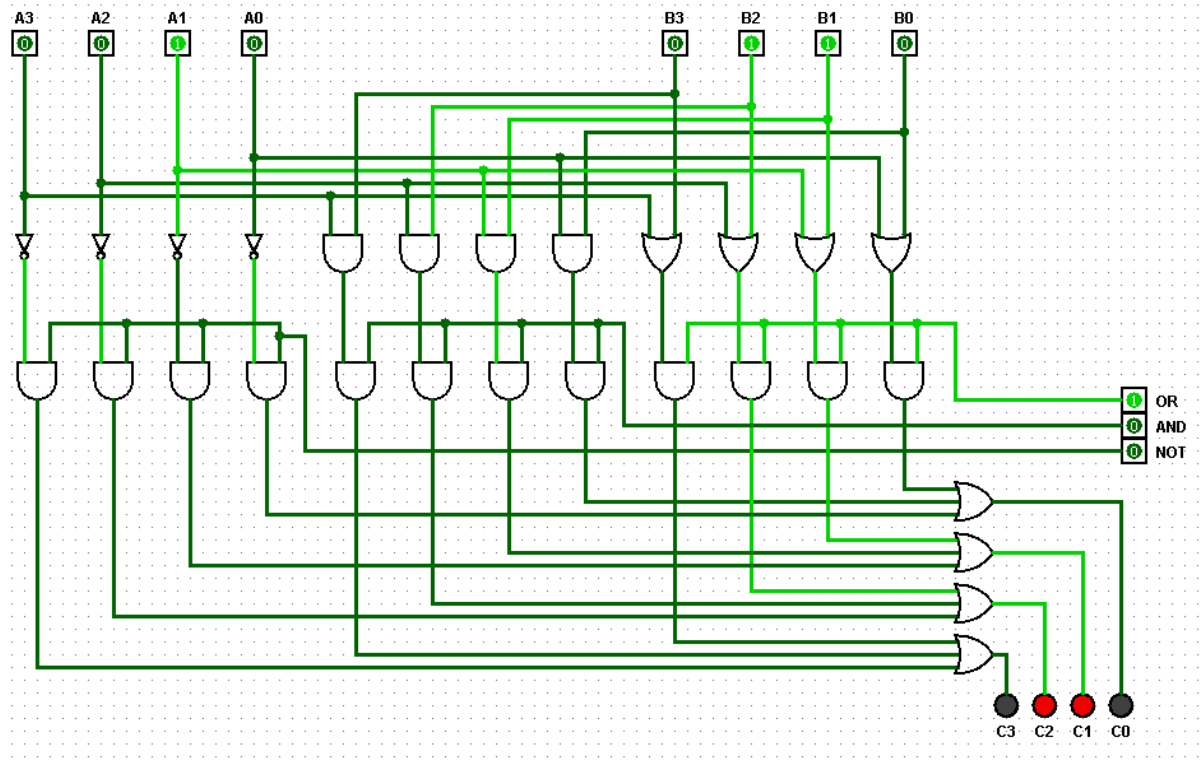

Figura 30. ALU com A (0010) OR B (0110).

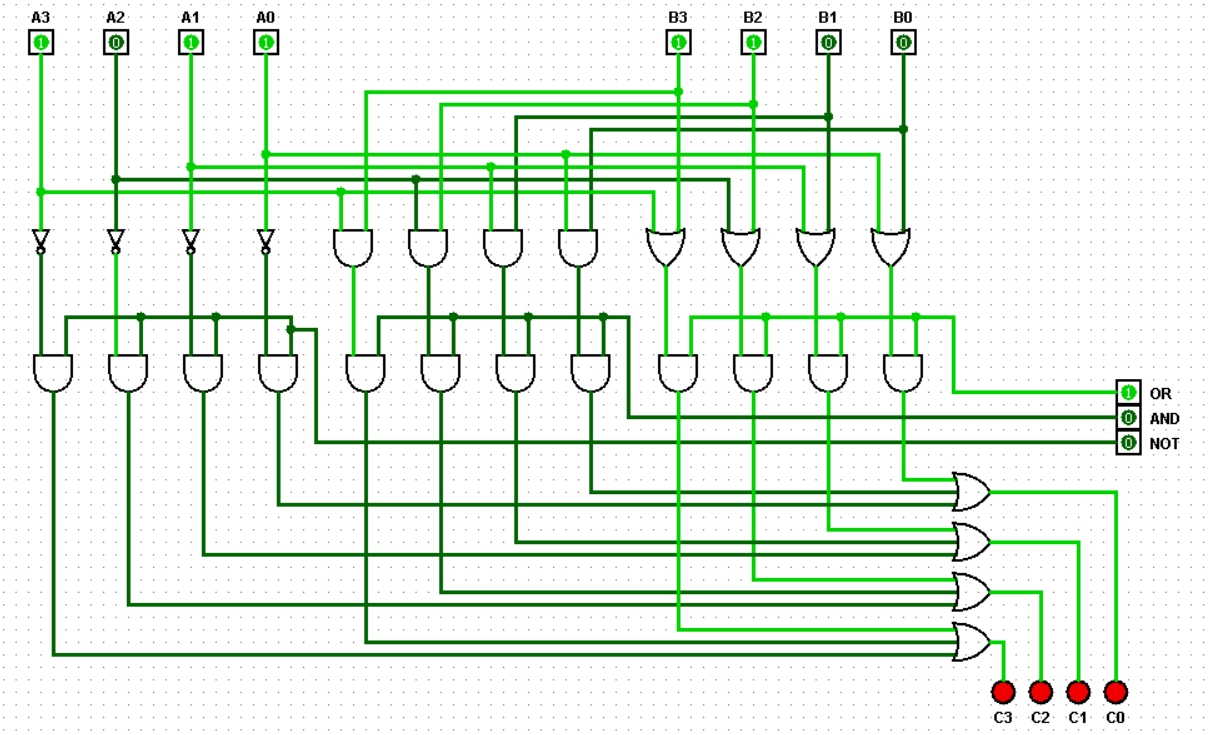

Figura 31. ALU com A (1011) OR B (1100).

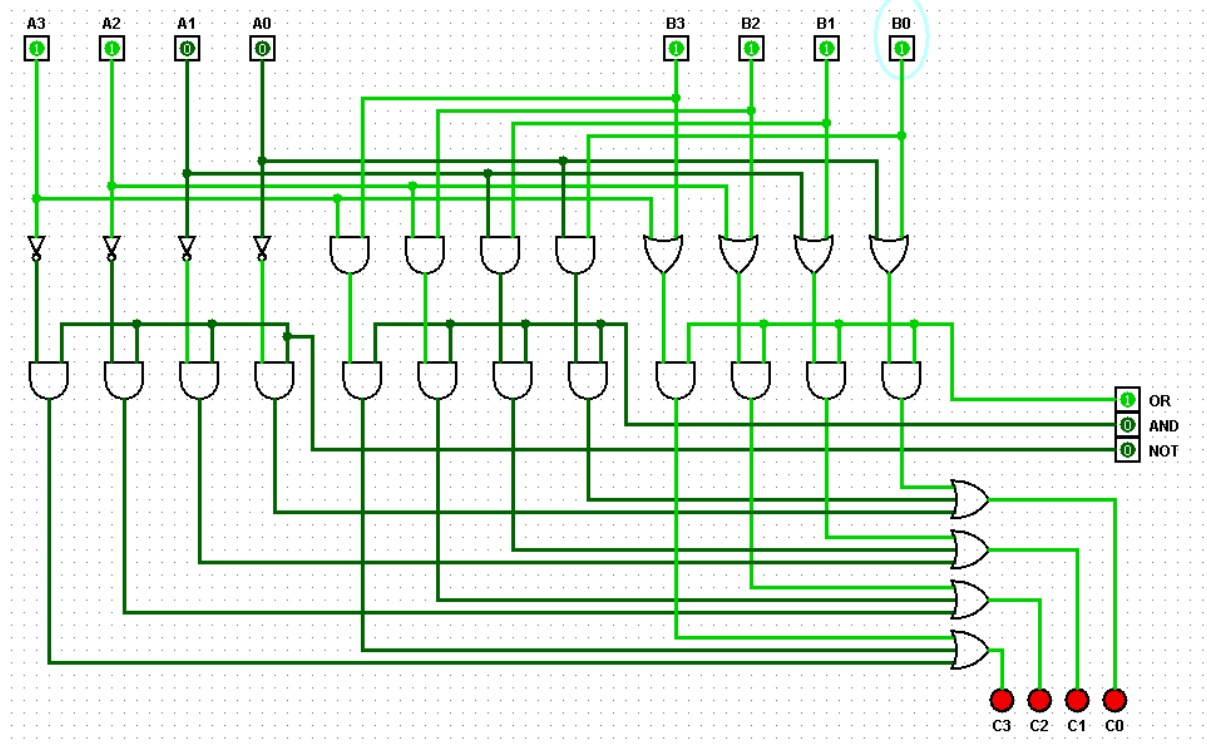

Figura 32. ALU com A (1100) OR B (1111).

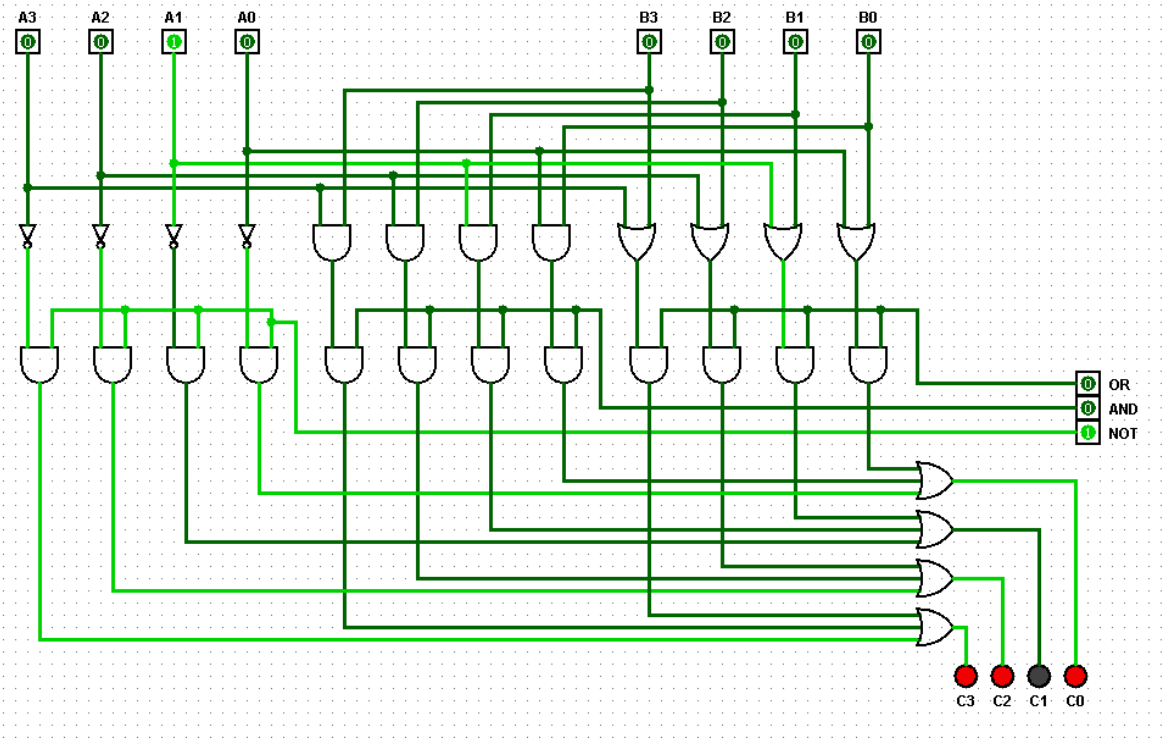

Figura 33. ALU com NOT A (0010).

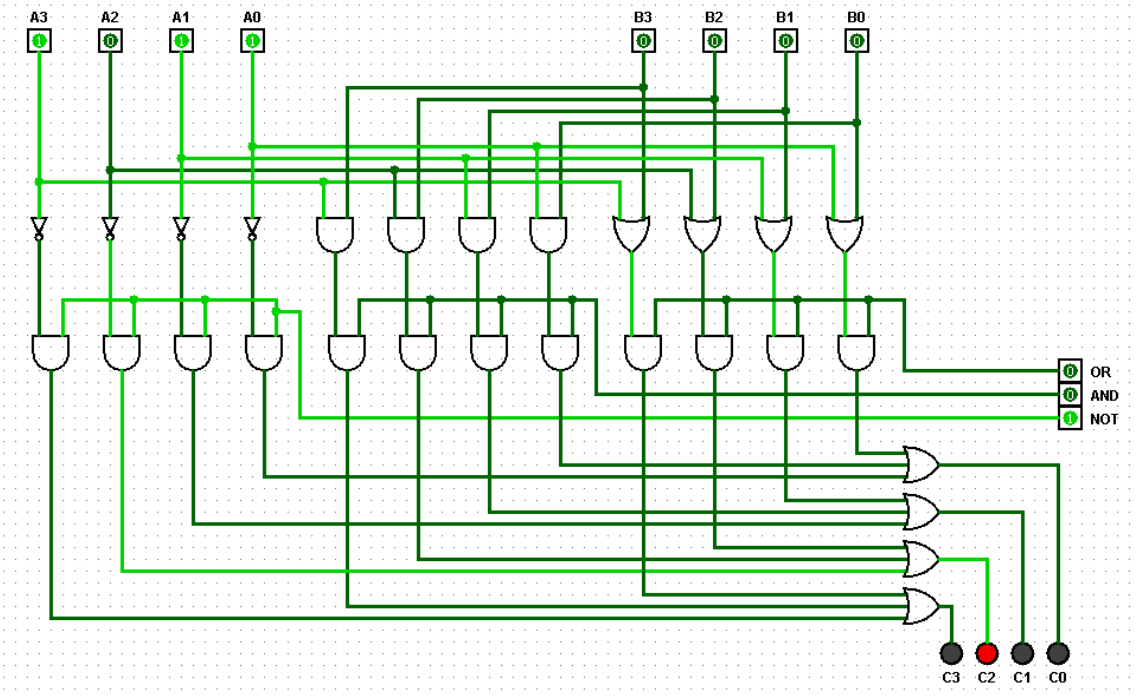

Figura 34. ALU com NOT A (1011).

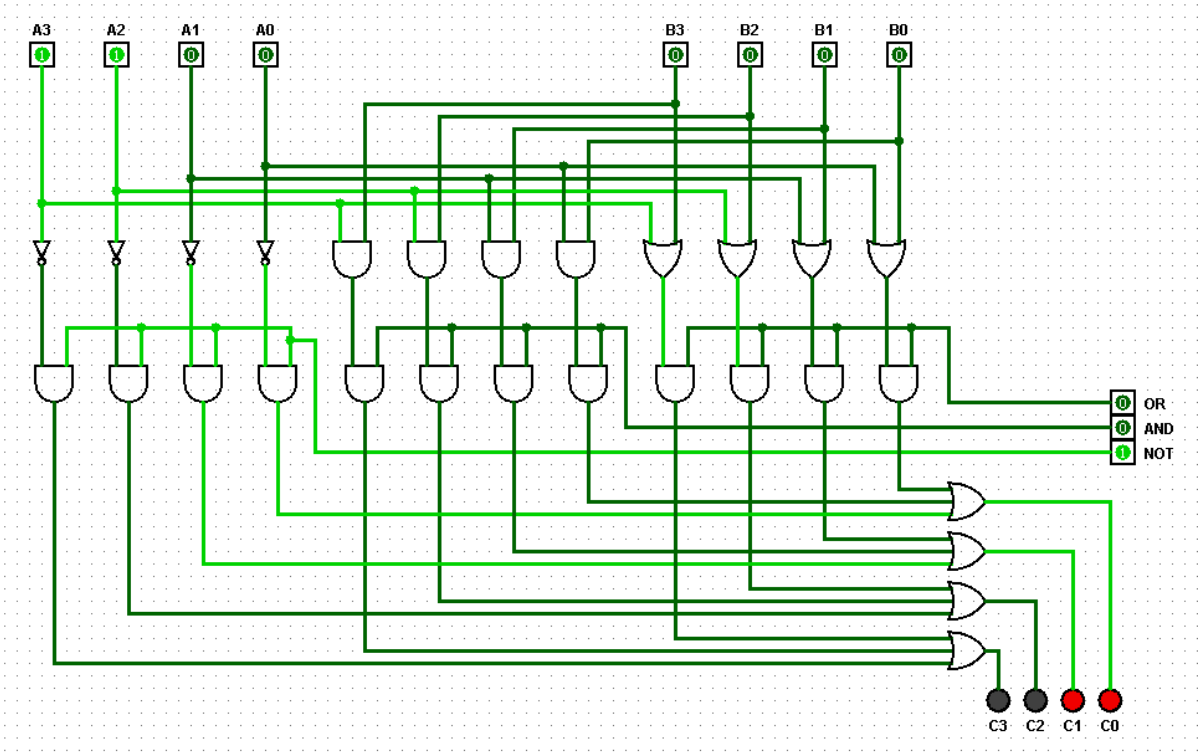

Figura 35. ALU com NOT A (1100).

7. Conclusão

Esse trabalho acabou por aplicar vários dos conhecimentos por nós aprendidos durante as aulas ministradas pelo Docente Regente João Gluz, além da aplicação prática usando o software *Logisim* para poder apresentar visualmente os diagramas conceituais dos circuitos trabalhados.

Em sumarização, todo o desenvolvimento dessa atividade acabou por nos auxiliar no melhor aprendizado da matéria, além de realmente pôr em prática e testar a manipulação de binários como um todo.