Universidade Federal da Fronteira Sul - Campus Chapecó

Trabalho Final de Circuitos Digitais: Relatório - Batalha Naval

Aluno: Douglas Kosvoski

Matrícula: 1911100022

CCR: Circuitos Digitais 2º Fase - Vespertino

Professores: Luciano L. Caimi e Adriano S. Padilha

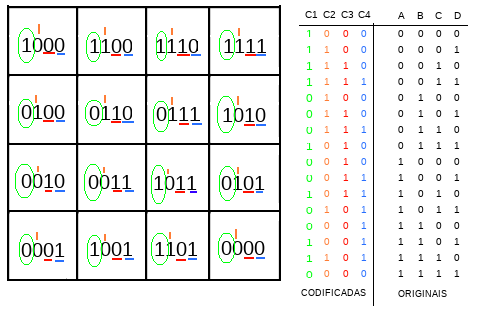
* Apresentação

O trabalho final da disciplina consiste em codificar e implementar simulações do jogo [Batalha Naval](https://pt.wikipedia.org/wiki/Batalha_naval_(jogo)), jogo de tabuleiro de dois jogadores, no qual os jogadores têm de adivinhar em que quadrados estão os *navios* do oponente. Seu objectivo é derrubar os barcos do oponente adversário, ganha quem derrubar todos os navios adversários primeiro, formado por uma matriz 4x4 (16 posições, tendo cada posição uma codificação associada) com dois barcos dispostos nela, tendo a solução implementada em diversas plataformas, entre elas estão: [TinkerCad](https://www.tinkercad.com/), [Logisim](http://www.cburch.com/logisim/), [Quartus Web Edition](https://www.intel.com/content/www/us/en/programmable/downloads/download-center.html) e [Altera DE1](https://www.intel.com/content/www/us/en/programmable/solutions/partners/partner-profile/terasic-inc-/board/altera-de1-board.html).

Os comandos se baseiam em 2 conjuntos de chaves codificadas para os barcos (4 *Keys* para cada barco), 1 conjunto de chaves sem codificação para o adversário (4 *Keys*), bem como um *switch* de disparo (o disparo é realizado na posição real da matriz de acordo com o *input* do adversário). Tendo como retorno *LED’s* de coloração verde (em nível lógico alto) para quando o disparo acertar o alvo e *LED’s* de tom avermelhado (em nível lógico alto) para quando o alvo escapar ileso do disparo.

* Descrição da Solução

A solução encontrada e utilizada em todo o desenvolvimento do trabalho consiste em criar uma codificação qualquer (que já não estivesse sendo utilizada por outros grupos) para cada posição da matriz e em seguida a transformar em expressões lógicas utilizando [Algebra de Boole](https://en.wikipedia.org/wiki/Boolean_algebra) levando em consideração não a codificação como um todo para as expressões, mas sim, a partir da posição de cada caracter.



Percebe se que, cada dígito da posição da matriz está associada com uma sequência de módulos (conjunto de codificações) e com uma tabela-verdade.

Exemplo:

A posição 1001 na matriz para a ser considerada, a partir da codificação, a posição 0011, de acordo com o seguinte passo a passo:

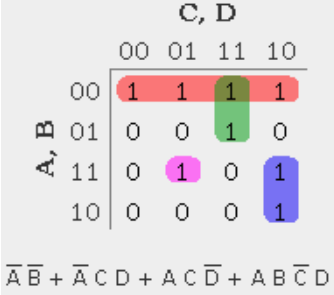
Todos os dígitos de 1001, ou seja, A,B, C e D participam em todos os módulos de codificação, por tanto, A B C e D entram no primeiro codificador passam por diversas portas lógicas e seus resultados nos retornam o valor C1 (primeiro dígito da posição codificada, dígito 0 da posição 0011), e assim por diante também com os outros codificadores, nos retornando também os valores de C2, C3 e C4. Que quando colocados lado a lado nos mostram a posição 0011 (codificada) que originalmente foi inserida 1001, ou seja entra 1001 no circuito e sai 0011. E esse procedimento remete a qualquer posição inserida para os barcos, lembrando que apenas as posições dos barcos são codificadas, no que se refere as chaves do usuário apenas são feitas comparações com as chaves codificadas dos barcos.

* Procedimento interno de cada módulo de codificação

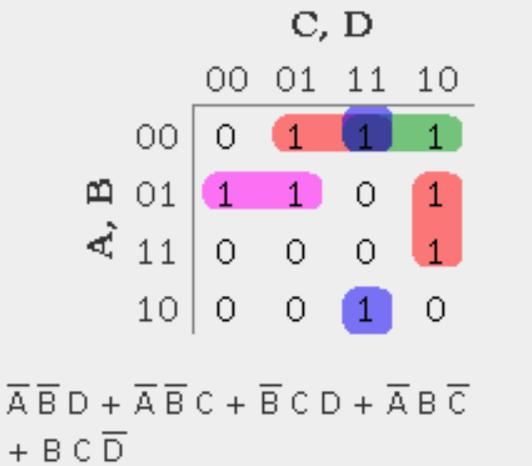
Ao pegarmos respectivamente todos os dígitos A, B, C e D de cada posição da matriz e fazermos uma tabela-verdade por coluna temos a tabela anterior, agora por [Soma de Produtos](https://pt.wikipedia.org/wiki/Mintermo_e_Maxtermo), sequencialmente pegamos todos os valores de nível lógico alto, ou seja, 1 e aplicamos o [Mapa de Karnaugh](https://en.wikipedia.org/wiki/Karnaugh_map), com a ajuda do programa Logisim para então fazermos as simplificações necessárias.

Temos por tanto, os mapas de Karnaugh dos módulos de codificação juntamente com a expressão lógica resultante, vemos respectivamente C1, C2, C3 e por fim C4.

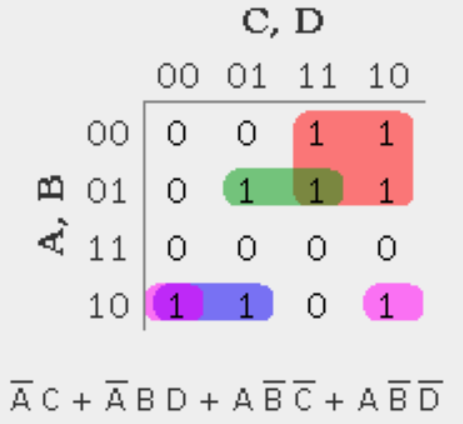
Mapa C1 - minitermos {0,1,2,3,7,10,13,14}



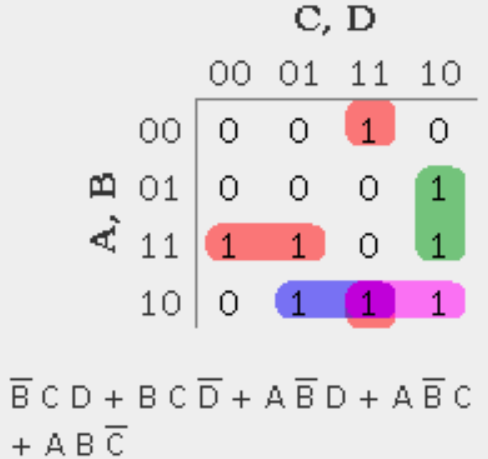
Mapa C2 - minitermos {1,2,3,4,5,6,11,14}



Mapa C3 - minitermos {2,3,5,6,7,8,9,10}



Mapa C4 - minitermos {3,6,8,9,10,11,12,13,14}

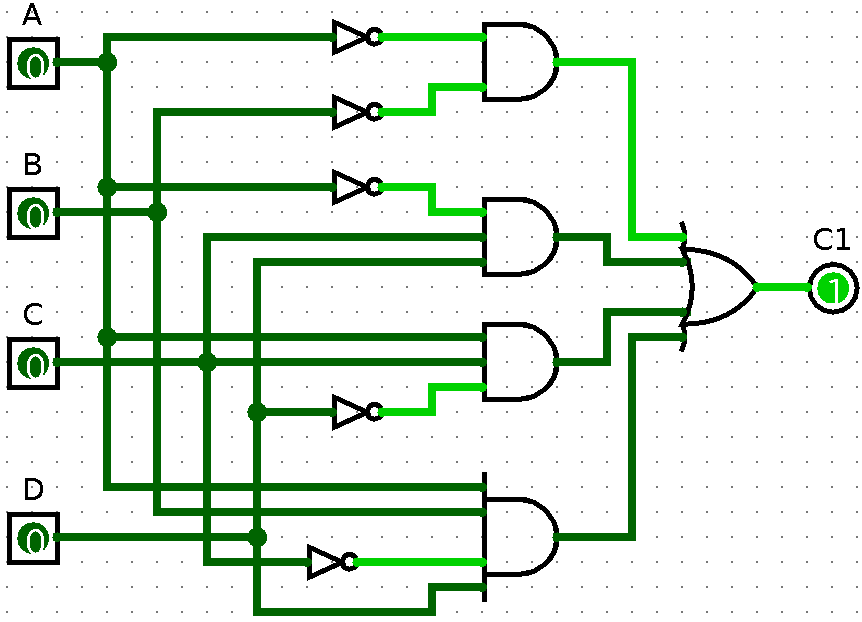
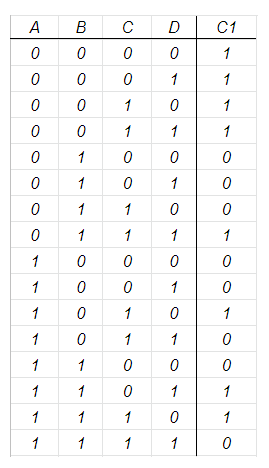


Visto que agora temos as Tabelas-Verdades, Mapas de Karnaugh com as Simplificações e Equações Lógicas, podemos aplicar as equações nas diversas plataformas. Vamos começar por aplicar no Programa *Logisim, em seguida TinkerCad e por fim mas não menos importante Quartus juntamente com a placa Altera DE1.*

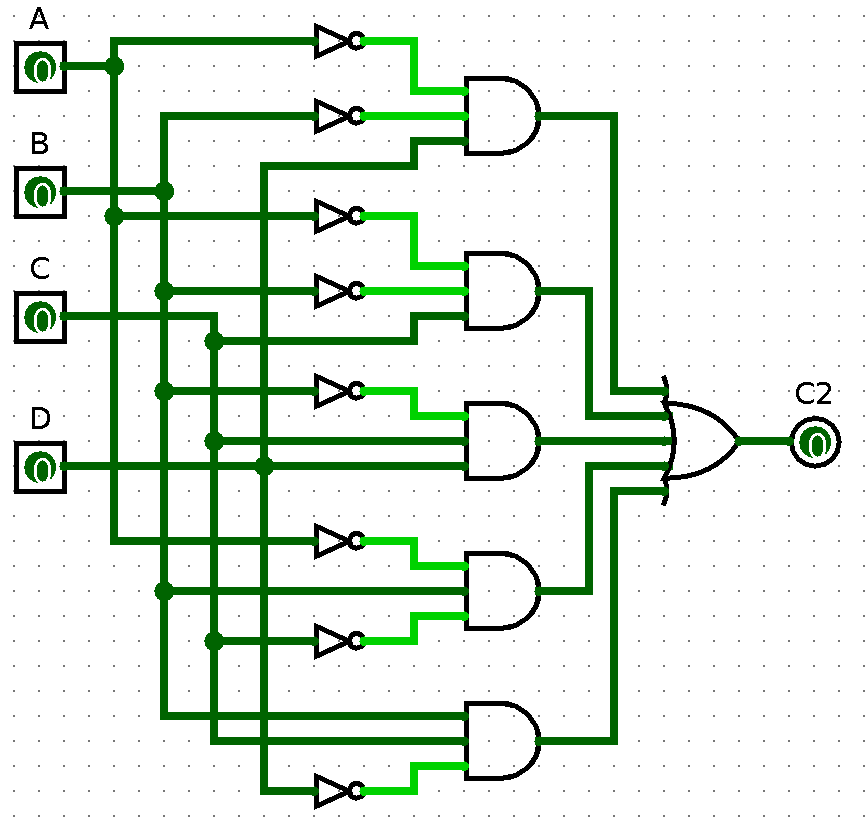
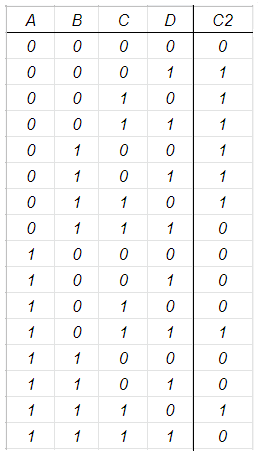
*Logisim*

*O programa Logi*sim se mostra extremamente fácil e intuitivo de se utilizar, em questão de segundos a partir da escolha de como será gerado o circuito, seja por meio da inserção da tabela-verdade ou da equação lógica, e da escolha das variáveis de entrada e saída, é apresentado de prontidão o circuito resultante (pessoalmente escolhi inserir individualmente cada tabela-verdade para todos os codificador, tanto para me assegurar do resultado final, quanto para escolher em qual arquivo cada codificador seria salvo, no meu projeto também escolhi por salvar cada módulo em um arquivo separado).

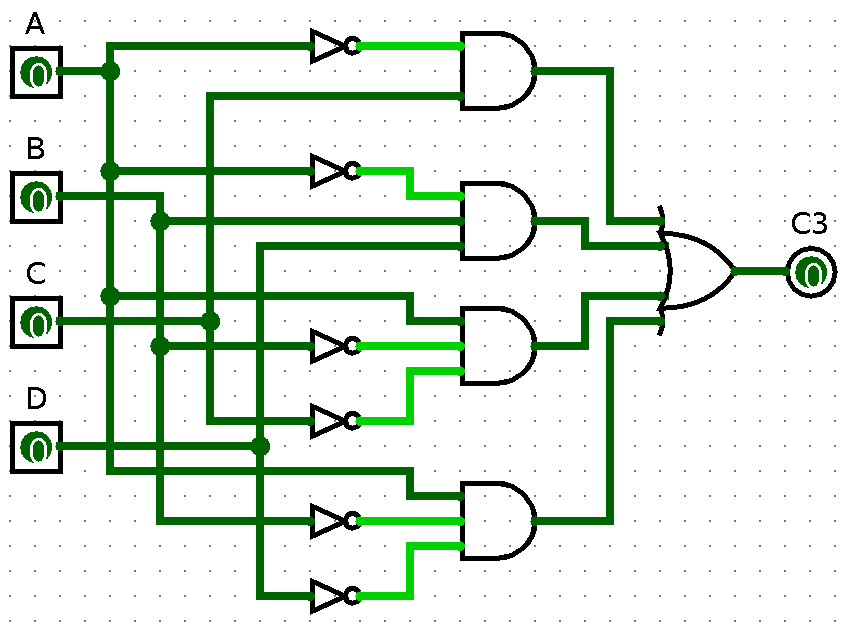
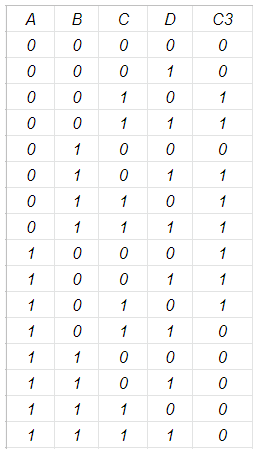
Ao inserimos a tabela C1 temos codificador C1



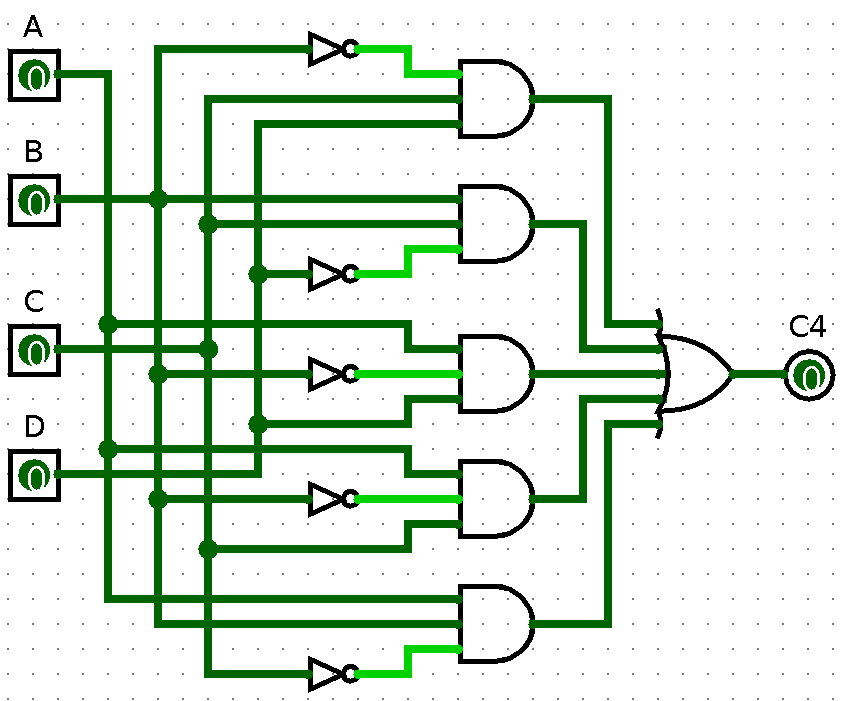
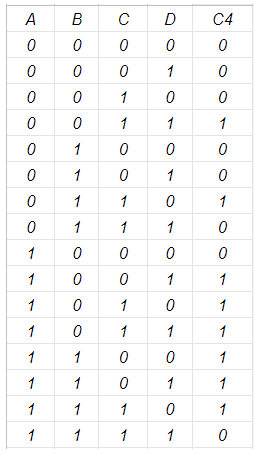
Ao inserimos a tabela C2 temos o codificador C2



Ao inserimos a tabela C3 temos o codificador C3

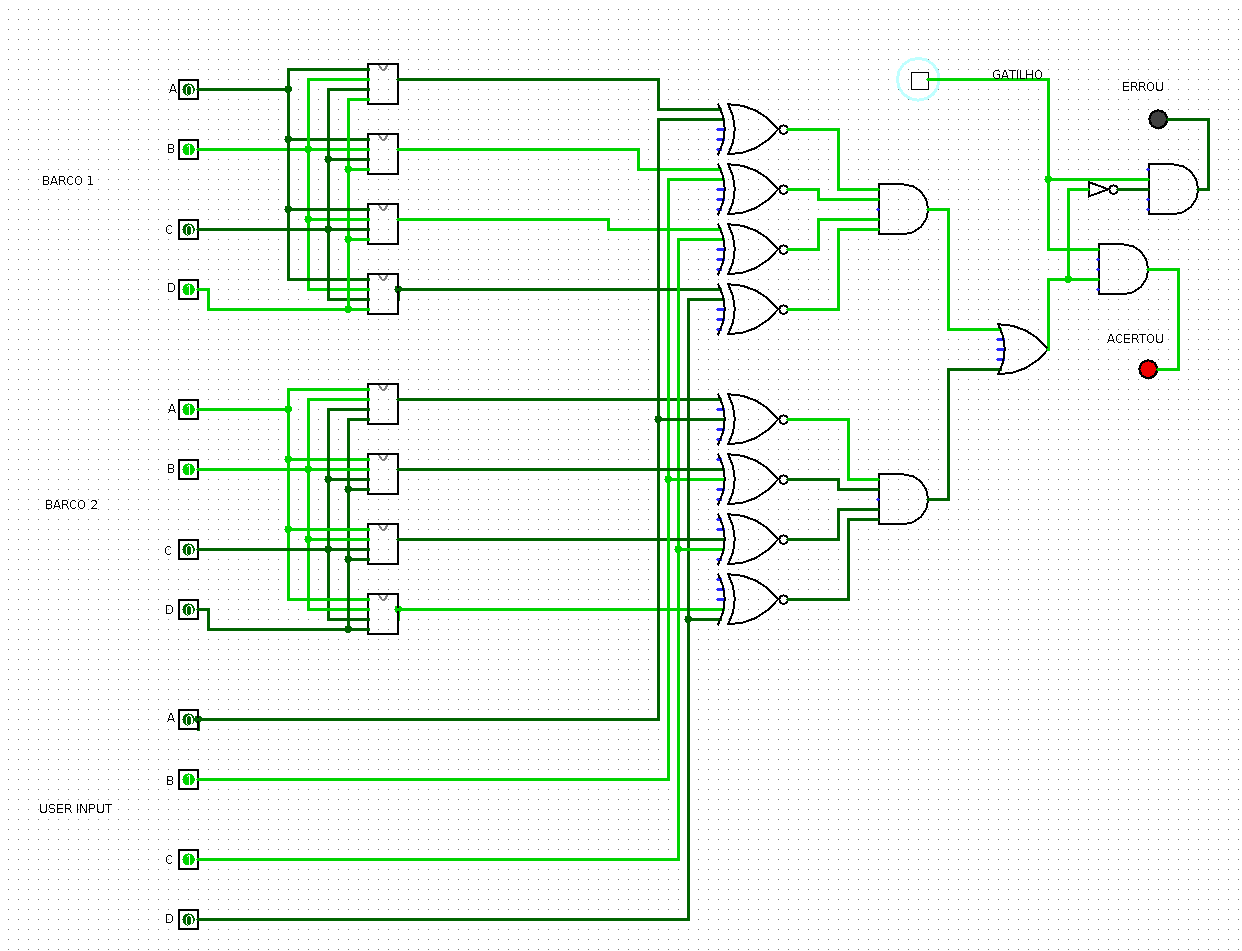


Ao inserimos a tabela C4 temos o codificador C4



E com cada módulo (C1, C2, C3 e C4) podemos então junto com o *INPUT* do adversário formar a arquivo/estrutura *Main* (Junção dos módulos de codificação, entrada do adversário e comparador entre o resultado da codificação e *inputs).*

*Main*

**

Explicação:

BARCO 1 possui 4 entradas, cada uma com 2 opções de entrada (nível lógico baixo ou alto), assim como o BARCO 1, o BARCO 2 possui 4 entradas binárias (0 ou 1), essas 4 entradas correspondem ao A, B, C e D individuais dos barcos, que entram em cada um dos codificadores que foram importados a partir do circuito gerado das tabelas-verdades anteriores.

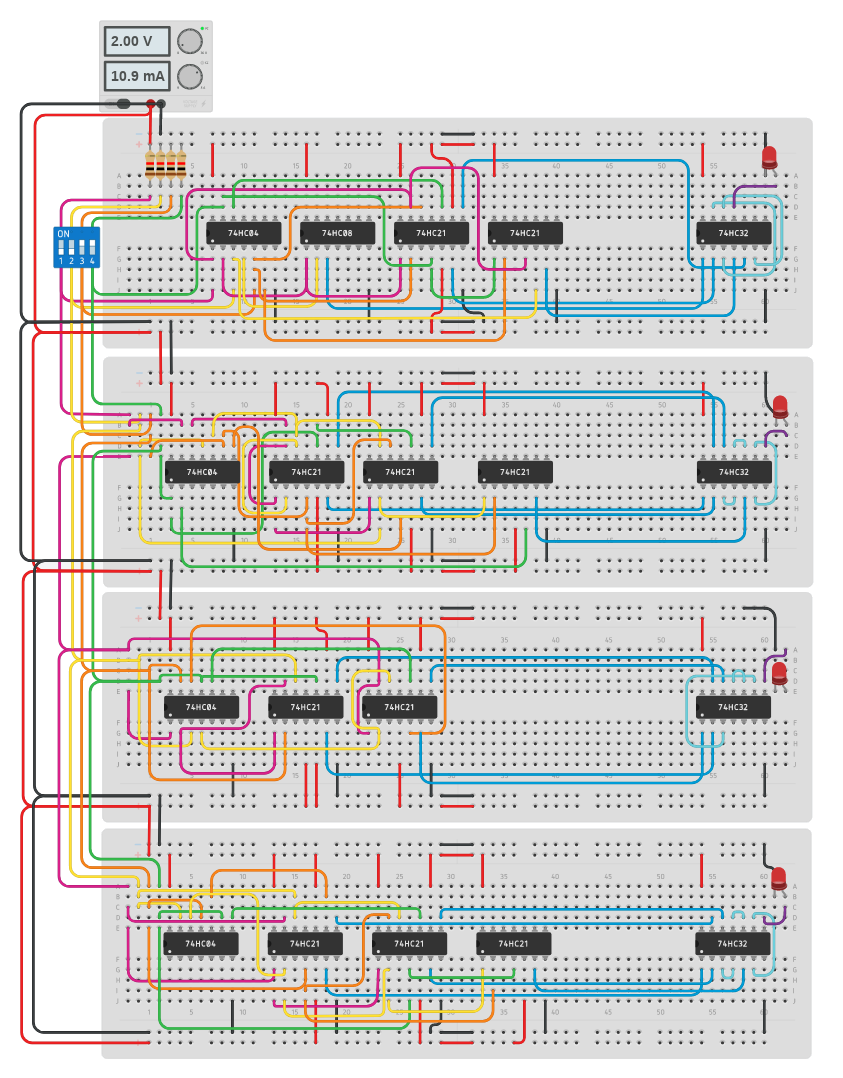
Ainda em relação ao barcos a saída desses codificadores são ligadas em um XNOR (XOR negada) juntamente com a entradas do usuário (que tentará adivinhar as posições dos barcos por meio de quatro entradas de valores ou 0 ou 1) (C1 xnor A, C2 xnor B, C3 xnor C, C4 xnor D) o resultado dessas 4 portas XNOR é então enviado para uma AND, a AND é responsável por assegurar que as entradas do usuário estejam completamente corretas e não parcialmente. O mesmo procedimento de codificar as entradas do BARCO 1 e comparar com as entradas do usuário por meio de XNOR e AND,acontecem com o BARCO 2.

Por seguinte, temos uma porta OR que verifica se tanto o usuário adivinhou as posições do BARCO 1 quanto do BARCO 2, a mesma porta OR envia o sinal lógico agora para uma AND que analisará se o botão de Gatilho foi acionado, isso é de extrema importância, visto que, esse sistema é assíncrono, ou seja, independe de CLK (clock), o gatilho é por tanto nossa entrada manual de sinal de clock, responsável por dar prosseguimento no circuito. Se o resultado da OR for 1 (nível lógico alto) e o botão de disparo for acionado, reflete que o usuário descobriu a posição de nosso barco e por consequência aciona um LED de retorno ao usuário sinalizando seu acerto, porém se a saída da porta lógica OR for 0 (nível lógico baixo) e o gatilho for acionado um outro LED será acionado, agora sinalizando um disparo sem vítimas, enquanto o disparo não for acionado nenhum LED será aceso.

# Observação

Na imagem acima o comparador do retorno do codificador com a entrada do usuário está representada por uma XNOR de 5 entradas, porém devido a minha falta de familiaridade com o software acabei não encontrando uma XNOR de 2 entradas, por isso, quero deixar claro que, se implementado na vida real o desempenho do circuito não estaria garantido, visto que as entradas sem utilização poderiam vir a afetar seriamente o comportamento da porta lógica, acarretando em um resultado errôneo e impreciso, um ajuste temporário poderia ser ligar a entradas sem utilização da XNOR no GND visto que o nível lógico baixo é o elemento neutro da porta e não afetaria seu resultado.

O mesmo acontece na porta OR (que compara se qualquer um dos barcos foi acertado) e na porta AND (que verifica o acionamento do disparo), tendo respectivamente que suas entradas sem utilização fossem ligadas em nível lógico baixo (OR) e alto (AND).

Aqui temos os circuito (Módulos C1, C2, C3 e C4) implementados na plataforma TinkerCad.**

Vemos Também que as entradas A e B em nível lógico baixo e C e D em nível lógico alto nos retornam todos os *Leds* acesos, ou seja, condiz com a tabela-verdade e equações anteriormente apresentadas.