UNIVERSIDADE FEDERAL DO CARIRI
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ARTHUR DE SOUZA RAMALHO
DANTE DINIZ PAES BARRETO

PROF. RAMON SANTOS NEPOMUCENO

2º PROJETO DE CIRCUITOS DIGITAIS
MATCHING GAME

SUMÁRIO

1. EXPLICAÇÃO DO JOGO

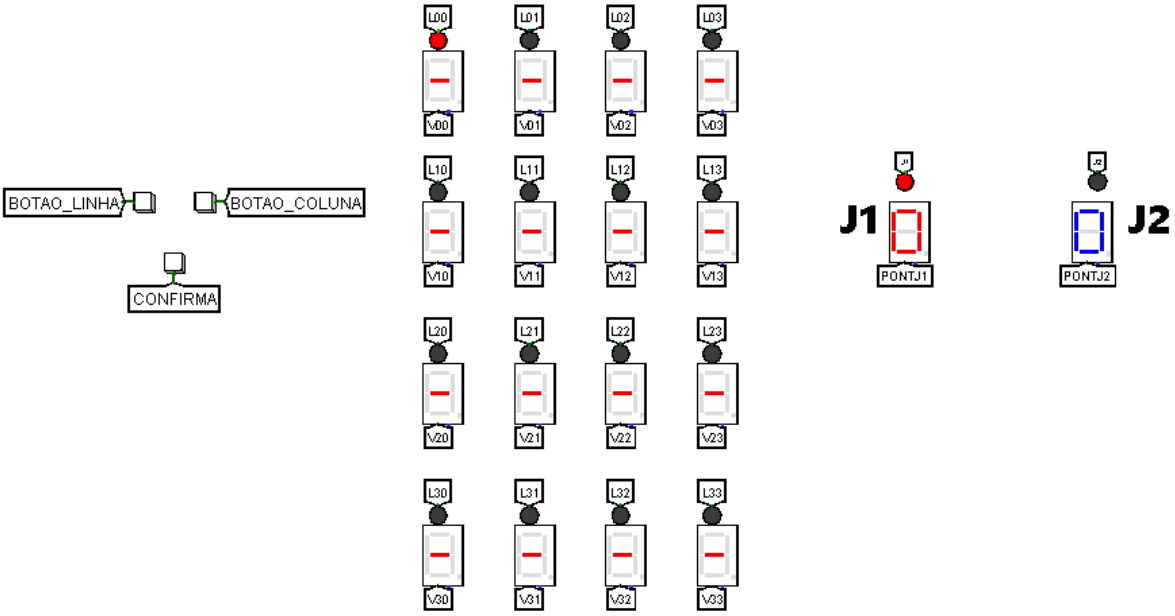

Figura 00: Imagem principal do jogo.

Matching Game ou jogo da memória consiste em dois jogadores tentarem formar pares de dois números iguais que estarão inicialmente ocultos. O tabuleiro do jogo é do tamanho 4x4, contendo 16 números e 8 pares de números iguais. A medida que os pares forem sendo acertados, os números serão revelados. O desafio do jogo consiste em usar a memória para acertar mais pares que o seu oponente, obtendo uma pontuação mais alta e consequentemente ganhando o jogo.

O jogo é dividido em turnos alternados entre os dois jogadores. O jogador 1 começa o jogo fazendo sua primeira escolha e logo em seguida a segunda escolha, após confirmar, ocorre a comparação do primeiro número escolhido com o segundo, se caso os dois forem iguais, então o jogador 1 recebe um ponto, caso contrário, a pontuação se mantém. Em ambos os casos, a vez da jogada é passada para o jogador 2, que repete todo o processo de escolhas dos números,

por fim, acertando o par ou não, a vez retorna ao jogador 1 e assim continua até todos os pares terem sido formados.

2. DESAFIOS DO PROJETO

Para criarmos o circuito do jogo, tivemos que resolver pequenos problemas, são eles:

1. Como montar o tabuleiro?
2. Como colocar os pares no tabuleiro?
3. Como selecionar em que posição será feita a jogada?
4. Como trocar de jogadores a cada turno?
5. Como exibir cada número na hora certa?
6. Como comparar os dois números do par?
7. Como deixar os pares acertados sempre visíveis?
8. Como aumentar a pontuação de cada jogador individualmente?
9. Como impedir do jogador escolher um par já acertado?

3. O PROJETO

3.1. O TABULEIRO

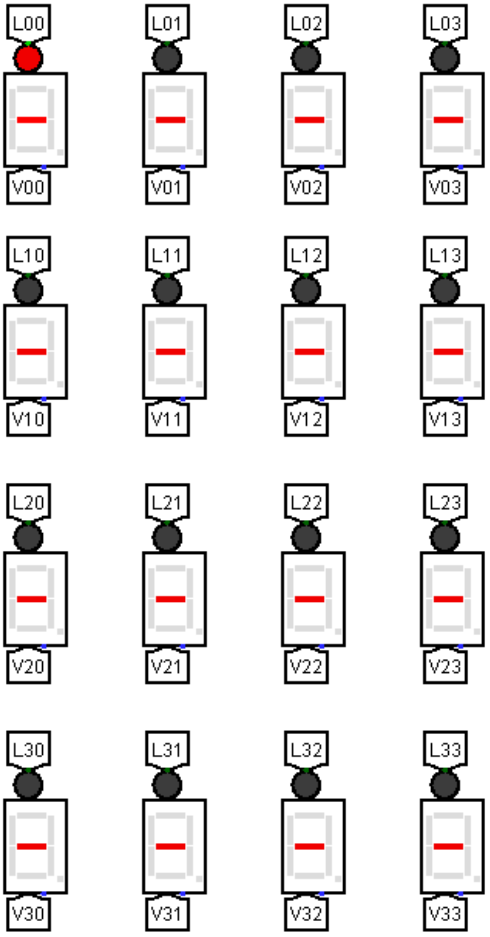

Figura 01: Tabuleiro do jogo.

O tabuleiro 4x4 é formado por 16 números, cada número é mostrado por um display hexadecimal que recebe o valor do número pelo túnel Vab que está conectado abaixo do display de led, onde a é o número da linha e b o número da coluna, cada um indo de 0 a 3. Além do display, há também um led logo acima para indicar a posição que será selecionada pelo jogador. O circuito para variar esta seleção será explicada mais para frente.

3.2. PARES NO TABULEIRO

Para o jogo funcionar, cada display deve ter um valor que seja igual a um único outro display, para isso usaremos uma constante.

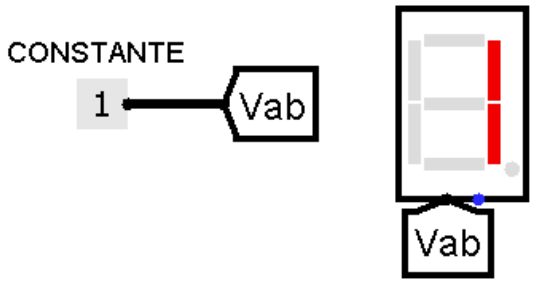

Figura 02: Constante.

A constante determina o valor que será mostrado no display. Para o tabuleiro inteiro, é preciso repetir o mesmo processo 16 vezes.

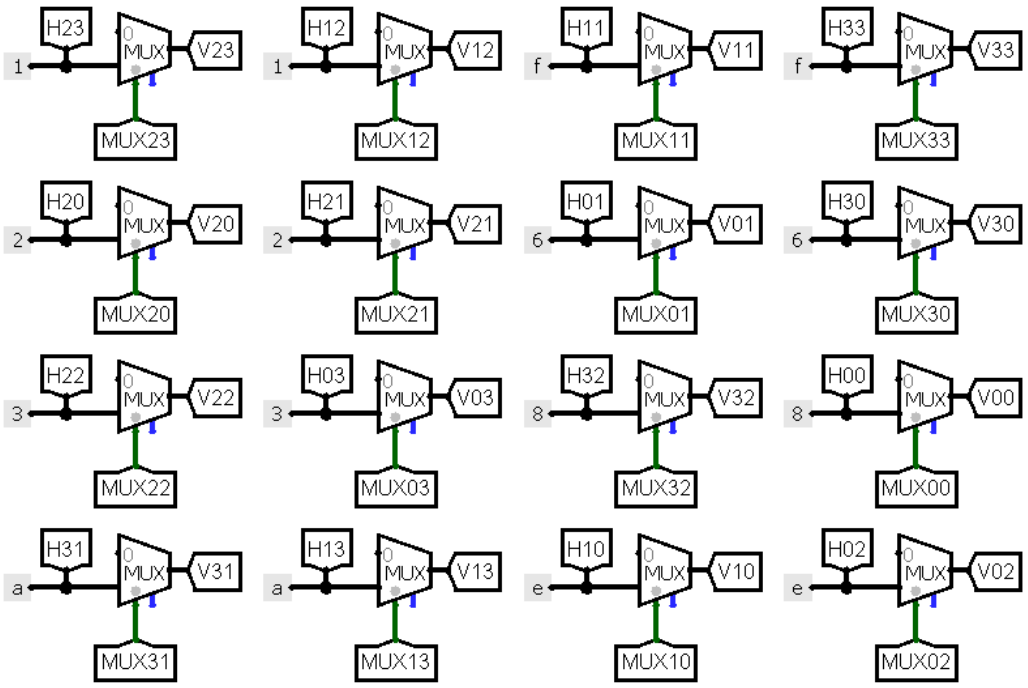

Figura 03: Valores das constantes do tabuleiro embaralhados.

Note que na figura 03 há o uso de multiplexadores que servirão para controlar o momento de revelar os valores no display. Além disso, os túneis MUXab para cada display correspondente à linha a e coluna b ativam ou desativam a exibição do valor da constante. Quando MUXab é 1, o valor é revelado, enquanto MUXab é 0 o valor não é mostrado. Os túneis Hab referentes à linha a e coluna b servirão mais à frente para comparar os valores nos turnos e verificar as condições de acerto do par.

3.3. SELEÇÃO DA JOGADA

Agora que temos o tabuleiro e os valores nos displays, precisamos fazer o circuito capaz de selecionar o número em qualquer posição que o jogador quiser. Para isso é necessário uma interface para o jogador.

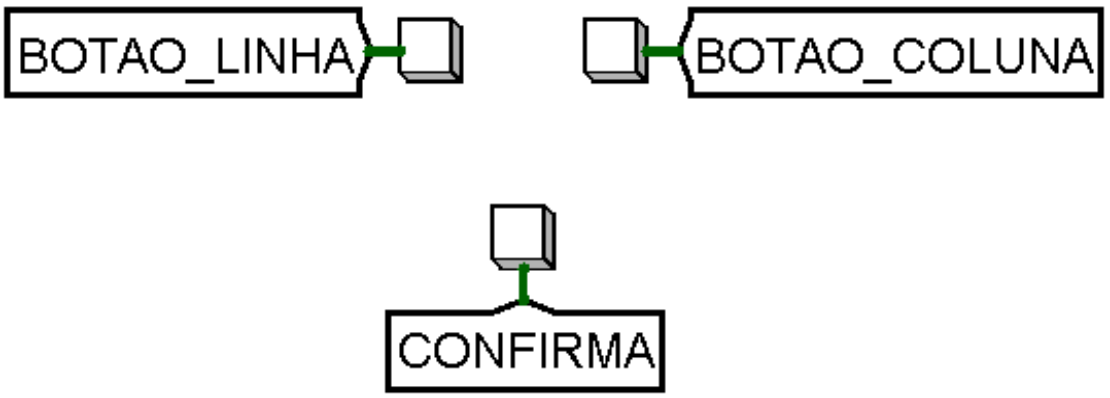

Figura 04: Botões de linha, coluna e confirmação.

O botão conectado ao túnel BOTAO_LINHA vai ser responsável por mover o led de seleção linha por linha. Já o botão conectado ao túnel BOTAO_COLUNA avança o led de seleção coluna por coluna. Além disso, quando o led estiver na última linha, apertar o botão BOTAO_LINHA fará com que o led retorne a primeira linha, analogamente, apertar o botão BOTAO_COLUNA quando o led está na última coluna fará com que o led retorne a primeira coluna. O terceiro botão conectado ao túnel CONFIRMA faz com que a posição seja de fato escolhida e o jogo passe para o próximo estado. Os estados que dividem o jogo serão explicados em uma próxima sessão.

O circuito que muda a posição do led quando os botões são apertados é um contador que conta de 0 a 3, pois o tabuleiro tem apenas 4 linhas e 4 colunas.

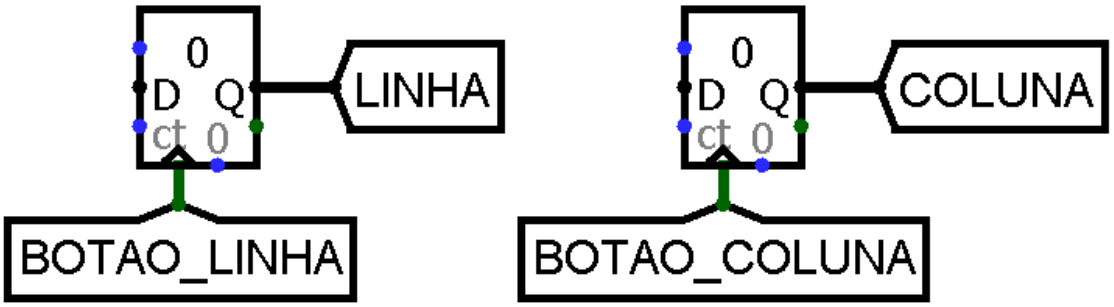

Figura 05: Contadores de linha e coluna.

Quando o botão é apertado, seja o botão de linha ou coluna, o túnel BOTAO_LINHA ou BOTAO_COLUNA passará o valor 1, isso atualiza o contador que soma uma unidade ao túnel LINHA ou COLUNA.

Por sua vez, o túnel LINHA ou COLUNA passam o valor que está no contador e são comparados com o valor da linha e coluna usando-se um comparador.

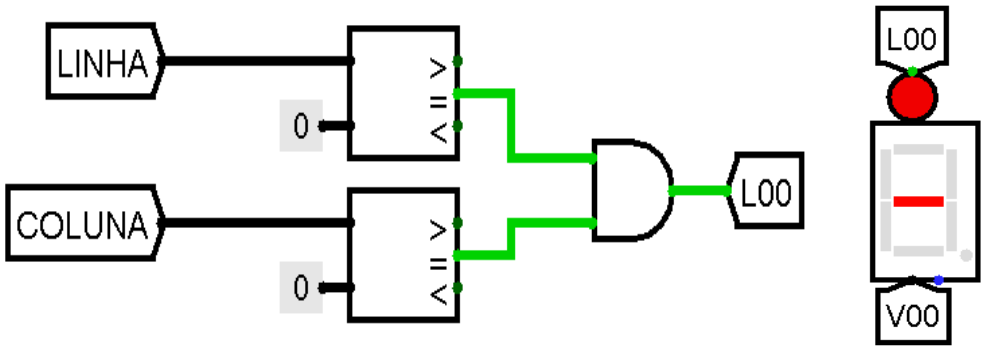

Figura 06 e 07: Led da linha 0 e coluna 0 aceso.

Com esse circuito, o led só acenderá quando o valor de LINHA e o valor de COLUNA forem simultaneamente iguais a linha e coluna respectivas da posição selecionada. Por isso é necessário o uso da porta lógica AND que recebe as duas comparações nas entradas.

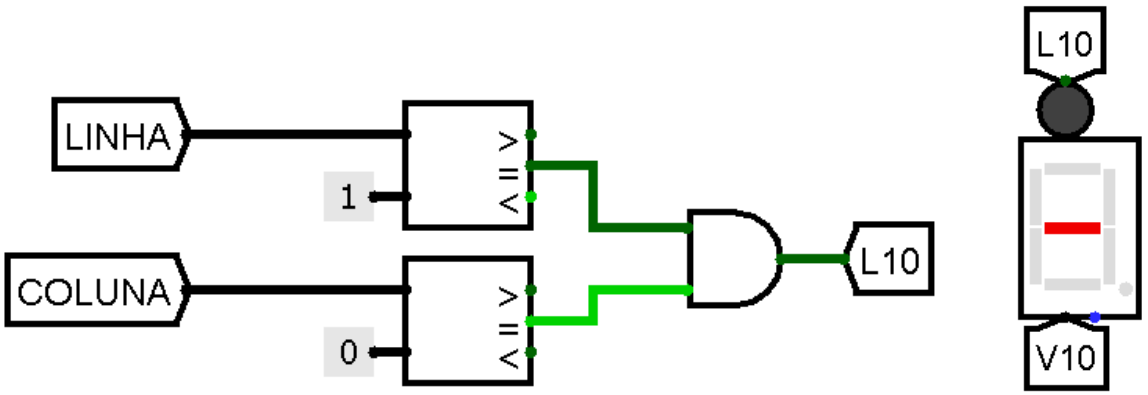

Figura 08 e 09: Led da linha 1 e coluna 0 apagado.

3.4. ESTADOS DO JOGO

Agora que temos o tabuleiro, os valores e os botões que permitem mudar a posição do número que será selecionado e confirmado, é preciso a implementação da lógica de troca de fases para fazer o jogo avançar. Para a mudança de estados, utilizaremos o conceito de máquina de estados, que basicamente é um contador em que cada número representa um estado.

O jogo será composto por 4 estados, isso significa que o contador deve contar de 0 a 3. A cada turno do jogo começa com todos os pares ocultos, depois o primeiro display é revelado, e então o segundo, e por fim uma fase de transição para contagem da pontuação. O contador tem que representar as 4 fases do jogo:

- ESTADO_0: Ambos os displays ocultos, é nesta fase que o jogador fará sua primeira escolha;
- ESTADO_1: O primeiro display está revelado, é nesse momento que o jogador fará a sua segunda escolha;
- ESTADO_2: Ambos os displays estão revelados, o jogador já fez a escolha do par e deve apertar o botão CONFIRMA novamente e passar para a próxima fase;
- ESTADO_3: Fase transitória que serve apenas para verificar a igualdade do par e aumentar a pontuação em caso de acerto.

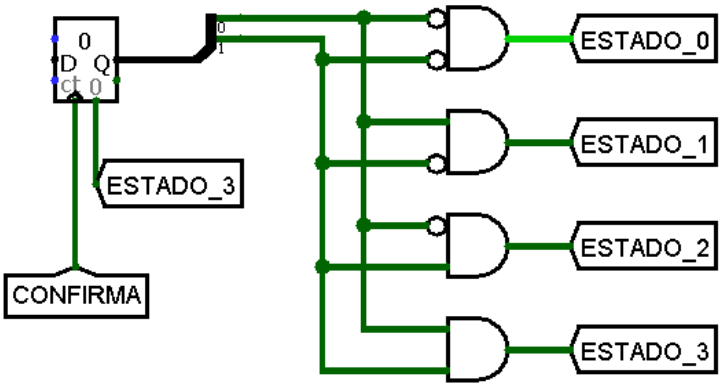

Figura 10: Máquina de estados.

Na figura 10, o botão CONFIRMA faz com que o valor do contador seja atualizado em mais um, passando para o próximo estado. Quando o contador está em zero, ou seja 00_2 na base binária, o estado será ESTADO_0, é por isso que a porta AND está com as duas entradas barradas. O mesmo ocorre quando o contador está em um, ou 01_2 em binário, representando o ESTADO_1, com apenas a entrada do segundo bit negada. 10_2 para o ESTADO_2 e 11_2 para o estado ESTADO_3. Além disso, quando o jogo está no ESTADO_3, o contador é automaticamente resetado, voltando ao valor zero e consequentemente ao ESTADO_0.

3.5. GUARDANDO OS VALORES DAS ESCOLHAS

Para que o jogo funcione corretamente, é preciso guardar os valores da linha e coluna da primeira escolha e da segunda escolha em registradores. A imagem a seguir é a lógica usada para essa função.

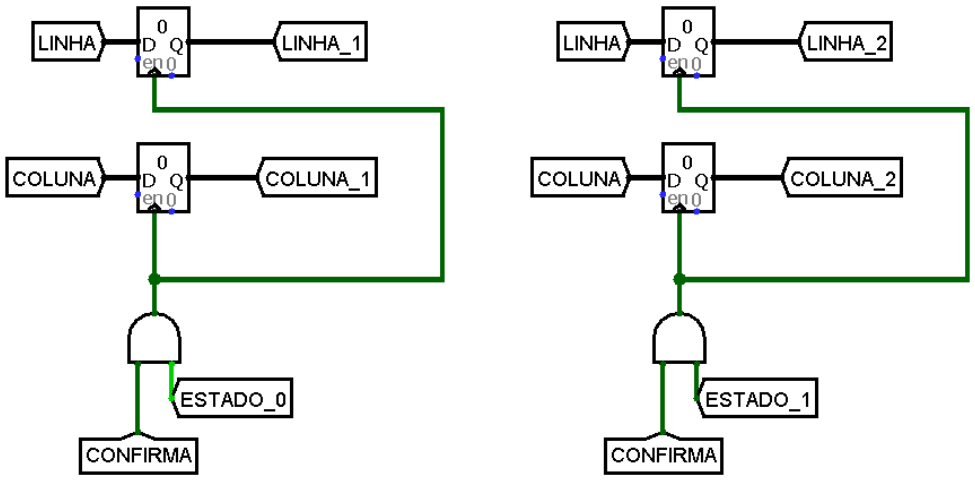

Figura 11: Registradores de linhas e colunas.

Na figura 11 está a lógica que guarda a primeira escolha(à esquerda) e a segunda escolha(à direita). Na parte esquerda desta imagem há um registrador mais acima que armazena o valor da linha atual em passa esse valor para o túnel LINHA_1. Logo abaixo há outro registrador responsável por guardar o valor da coluna atual e passar o mesmo valor para o túnel COLUNA_1. Todo esse processo ocorre quando está no ESTADO_0 e o botão CONFIRMA é apertado, ou seja, na transição entre o ESTADO_0 e o ESTADO_1.

Na parte mais à direita da figura acontece o mesmo processo que na esquerda com a primeira escolha. No entanto, agora os valores LINHA e COLUNA são armazenados em outros registradores e passados para os túneis LINHA_2 e COLUNA_2 respectivamente, indicando que é a posição do segundo número do par. Adicionalmente, o segundo par só será guardado nos registradores quando o estado do jogo é o ESTADO_1 e o botão CONFIRMA é pressionado, como é mostrado no uso da porta lógica AND na parte de baixo da figura 11. Os valores gravados nos quatro registradores serão usados em outras partes do circuito do jogo.

3.6. MOSTRANDO OS DISPLAYS

No jogo temos três situações que os displays deverão ser mostrados, são elas:

- Situação 1: O display será mostrado quando já tiver sido acertado;

- Situação 2: O jogo esteja no ESTADO_1 e o display em questão tenha sido a primeira escolha do jogador;
- Situação 3: O jogo esteja no ESTADO_2 e o display em questão tenha sido a segunda escolha do jogador.

Para que o display seja exibido caso se encaixe na primeira situação, é preciso usarmos um registrador. O registrador ACERTADO_00 na figura 12 guarda o valor 0 caso o display da linha 0 e coluna 0 ainda não tenha sido acertado em rodadas anteriores. Da mesma forma, guarda o valor 1 se o display da linha 0 e coluna 0 já tiver sido acertado. Para os outros displays usaremos a mesma lógica, ACERTADO_ab, sendo a o número da linha e b o número da coluna.

Para a segunda situação, o display será revelado quando as três condições forem simultaneamente verdadeiras. O estado atual do jogo será o ESTADO_1, a LINHA_1 e COLUNA_1 serem respectivamente iguais a linha e coluna do display que será aceso.

Analogamente, na terceira situação, teremos três condições que devem ser satisfeitas. O estado atual do jogo deve ser ESTADO_2, a LINHA_2 e COLUNA_2 devem ser respectivamente iguais a linha e coluna do display que será aceso.

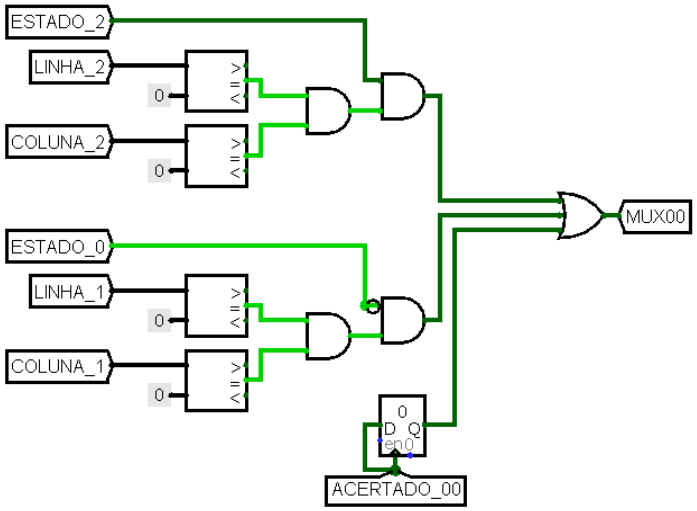

Figura 12: Exibição dos displays.

Na figura 12 está representada apenas a lógica para revelar o display da linha 0 e coluna 0, no entanto é preciso fazer a mesma lógica para cada um dos outros displays do tabuleiro. O túnel MUXab acenderá o display da linha a e coluna b, revelando o valor da constante, quando receber o valor 1. Para receber o valor 1, é preciso que pelo menos uma das três situações se encaixe, ou seja, ou o número

já foi acertado, ou é a primeira ou segunda escolha, por isso é preciso o uso da porta lógica OR. A porta lógica OR de três entradas recebe o valor 1 quando qualquer uma das situações seja satisfeita.

3.7. ACERTANDO OS PARES

Agora que temos como escolher o par no tabuleiro, temos que implementar a lógica responsável pela condição de acerto e o acréscimo de pontos. Vale lembrar que a comparação que verifica a igualdade do valor das duas escolhas e a contagem de pontos ocorrem durante o ESTADO_3, o estado transitório e o último de cada rodada. Para isso usaremos multiplexadores, que recebem 16 entradas, cada uma sendo um número de uma posição única do tabuleiro. Para que se possa escolher qual valor passar, teremos que usar o número da linha e da coluna que foram escolhidas, tais valores estão armazenados em LINHA_1 e COLUNA_1 para a primeira escolha e LINHA_2 e COLUNA_2 para a segunda escolha.

Para selecionar a primeira escolha do jogador, usamos LINHA_1 e COLUNA_1 para escolher qual dentre os 16 valores será usado na comparação com a outra escolha. Já o segundo número que o jogador escolheu será selecionado usando LINHA_2 e COLUNA_2. Vale o lembrete de que para se escolher um valor entre 16 é preciso de 4 bits de seleção. Para a primeira escolha são usados os 2 bits de LINHA_1 e os 2 bits de COLUNA_1, totalizando os 4 bits de seleção necessários. Da mesma forma, ocorre com a segunda escolha e os bits de LINHA_2 e COLUNA_2.

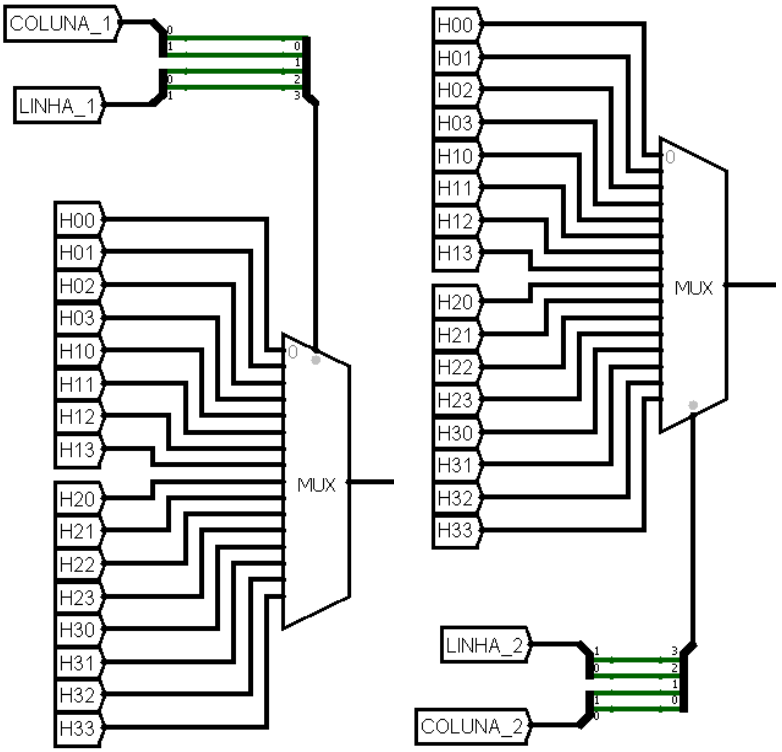

Figura 13 e 14: Multiplexadores das escolhas.

Os túneis H00 até o H33 representam os valores das constantes de cada linha, de L00 até L33. Cada túnel Hab, sendo a o número da linha e b o número da coluna, está diretamente ligado ao valor da constante, como mostra a figura 15.

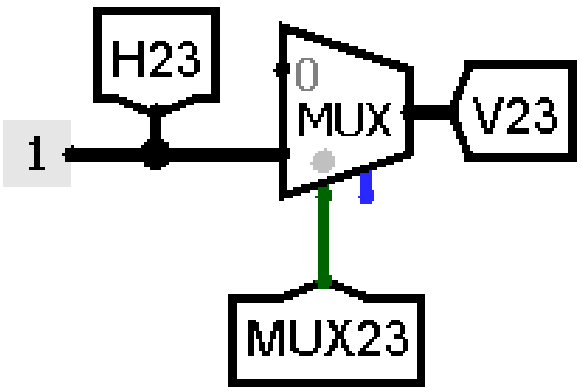

Figura 15: O túnel que leva o valor da constante ao multiplexador.

Agora, para verificarmos a igualdade dos valores das constantes escolhidas, usaremos o comparador.

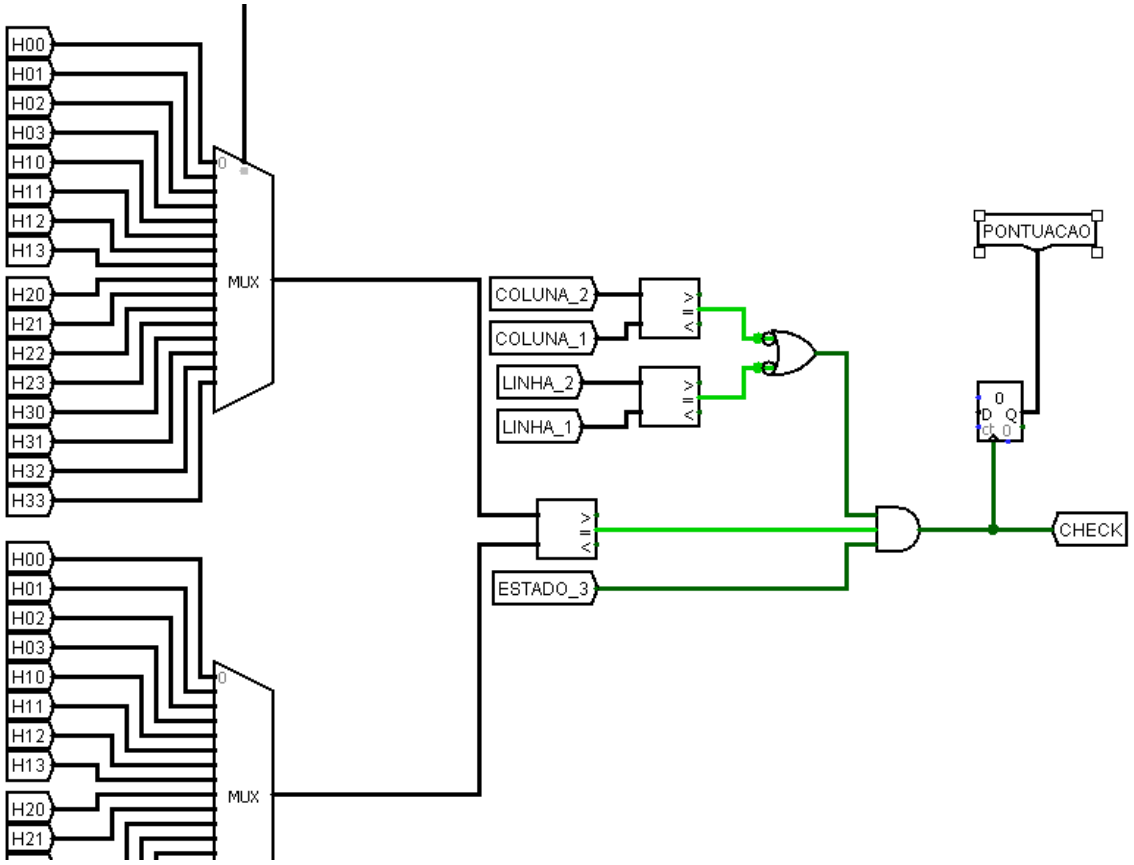

Figura 16: Comparação das escolhas e pontuação.

Na figura 16 temos os dois multiplexadores passando os valores das escolhas do primeiro e segundo display, estes dois valores serão comparados e se caso forem números iguais, o comparador passará 1 como saída. O comparador é apenas uma das três entradas do AND que será responsável pela condição de acerto do par. Para que o jogador ganhe um ponto, é necessário não só combinar dois números de valores iguais, mas também é preciso que esses números estejam em posições diferentes e o jogo está no ESTADO_3, o estado transitório em que acontece o acréscimo da pontuação. É obrigatório fazer a verificação das posições pois se nas duas jogadas o jogador escolher exatamente a mesma linha e coluna, é óbvio que os valores das constantes serão iguais e a pontuação subiria. Portanto, para que seja evitada esta situação, a LINHA_1 e LINHA_2 devem ter valores diferentes, mas caso a linha for a mesma nas duas escolhas, então COLUNA_1 e COLUNA_2 deverão ser diferentes. Pode ocorrer da COLUNA_1 e COLUNA_2 serem iguais, então LINHA_1 e LINHA_2 devem ser diferentes. O circuito capaz de resolver este problema está mostrado na figura 17 a seguir.

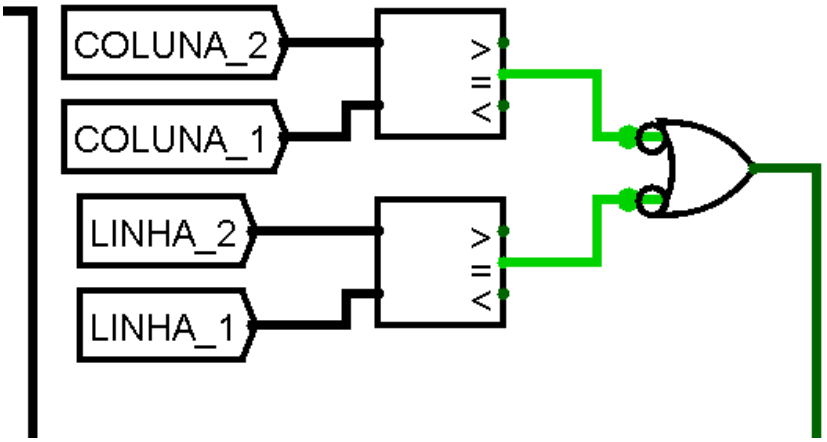

Figura 17: Posições diferentes das escolhas.

Quando todas as três condições forem satisfeitas, ou seja, quando as três entradas da porta AND recebem o valor 1, então a saída será 1. Esta saída envia o valor 1 para o túnel CHECK que serve para tornar o número acertado e exibi-lo de forma contínua. Além disso, a saída, quando em 1, atualiza o valor do contador que aumenta em uma unidade a pontuação. Até agora, como não há troca de jogadores, a pontuação é só uma, porém, quando for implementada a troca de jogadores, a pontuação deverá ser distribuída conforme os acertos de cada jogador.

Para que um par fique com a condição de acertado e seja exibido para sempre, é preciso que ele seja ou a primeira escolha ou a segunda, e ao mesmo tempo, receber o valor 1 do túnel CHECK. A lógica capaz de verificar esta situação é a seguir:

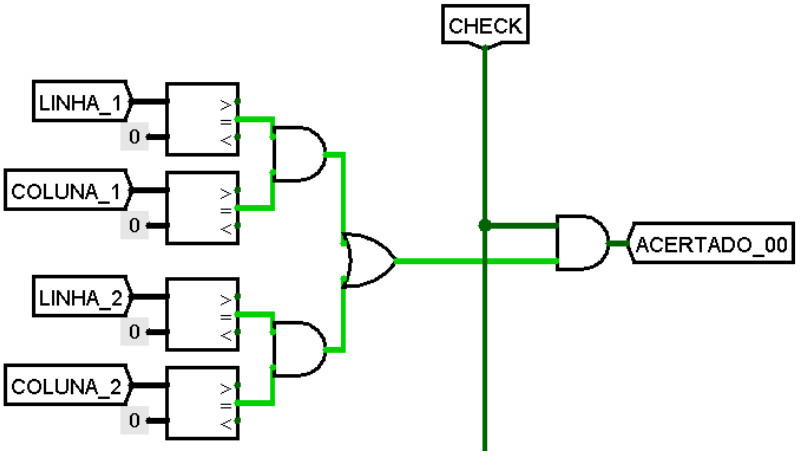

Figura 18: Checagem de acerto.

Na figura 18 está exemplificado apenas um display, no entanto o projeto precisa de uma lógica dessa para todos os 16 displays do tabuleiro. A cada acerto que o jogador fizer, combinando um par igual, dois números ficam acertados e serão exibidos para sempre.

3.8. TROCA DE JOGADORES

Agora iremos implementar a lógica de troca de jogadores. Vamos usar o conceito de máquina de estados, usado anteriormente na mudança de estados do jogo. O jogo da memória é jogado por dois jogadores, isso significa que o contador será de apenas um bit. Quando o contador estiver em 0, a vez de jogada será do jogador 1, quando o contador estiver em 1, será a vez do jogador 2. Além disso, toda vez que o jogo passar pelo ESTADO_3, o contador atualiza, e assim o jogador muda.

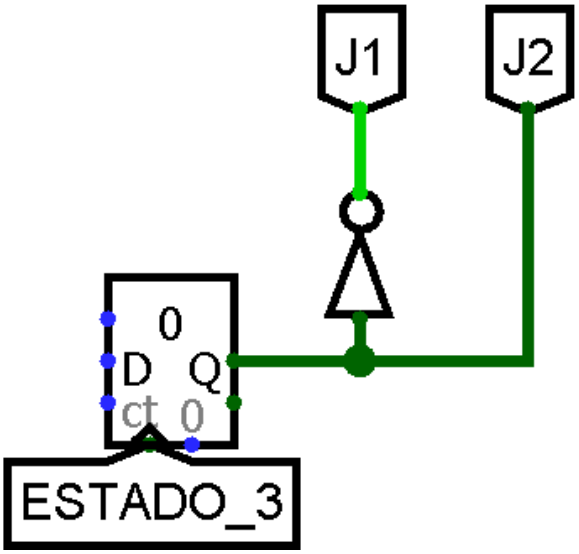

Figura 19: Troca de jogadores.

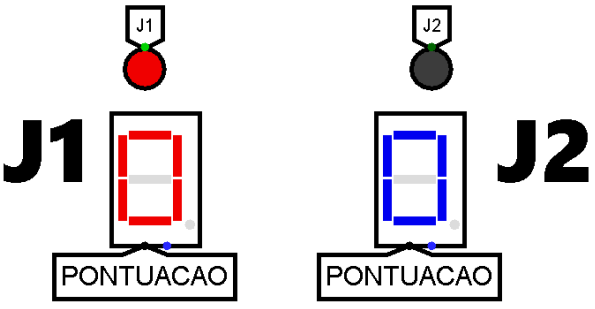

Figura 20: Led dos jogadores.

Na figura 20 estão os leds que indicam a vez do jogador. O led acima do display recebe o valor do contador da figura 19. Se o valor passado pelo túnel J1 ou J2 for 0, então o led estará apagado. Se o valor passado pelo túnel for 1, o led ficará aceso, indicando de quem é a vez de jogar. Na figura 20, o turno é do jogador 1. O display e o led do jogador 1 são vermelhos, já o display e o led do jogador 2 são ambos azuis, apenas por questão de diferenciação.

3.9. PONTUAÇÃO

Na figura 16, quando o par é acertado, o contador acrescenta em 1 a pontuação, que passa pelo túnel PONTUACAO o valor até o display dos jogadores, como mostrado na figura 20. No entanto, essa lógica tem um problema, a pontuação é acrescentada de forma igual aos dois jogadores, independente de quem for a vez. Para consertamos isto, usaremos uma porta lógica AND.

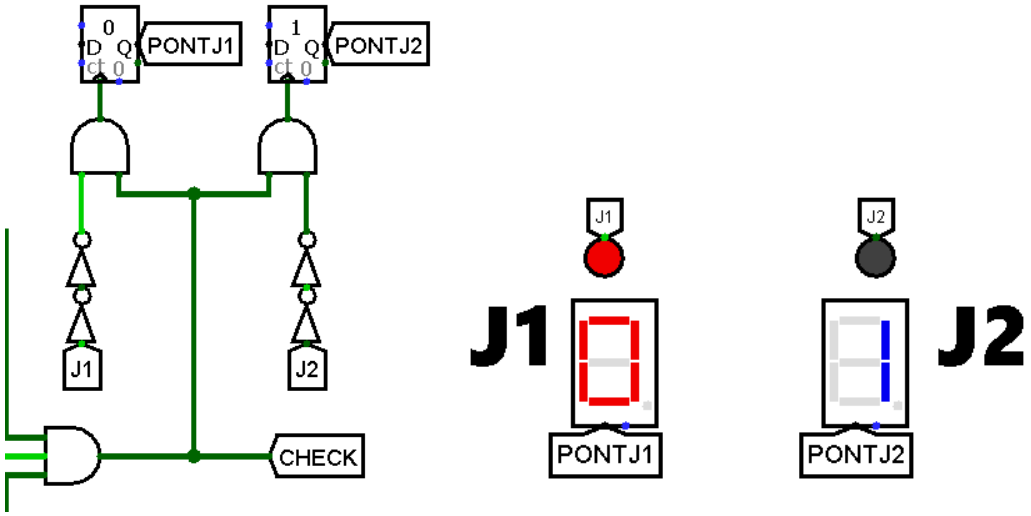

Figura 21 e 22: Acréscimo da pontuação de cada jogador.

Dessa forma, temos um contador para cada jogador, que pode armazenar valores diferentes conforme os acertos de cada jogador. O contador da esquerda passa o valor armazenado pelo túnel PONTJ1 até o display de pontuação do primeiro jogador, como na figura 22. Em paralelo, o contador da direita passa o valor armazenado pelo túnel PONTJ2 até o display do segundo jogador.

Na figura 21 está a lógica responsável por resolver este problema. Com a porta lógica AND, garantimos que a pontuação só será atualizada quando está na vez do jogador. No caso da mesma figura, quando o par for acertado, apenas o primeiro jogador ganharia um ponto, pois além de receber o valor 1 após a checagem do par, a outra entrada da porta lógica AND, vinda do túnel J1, também é 1. Já o segundo jogador teria apenas uma das entradas do AND sendo 1, e a outra 0, ou seja, a saída também será 0. O uso das duas portas NOT após J1 e J2 é por garantia de que o sinal passe um pouco mais devagar, para que a pontuação possa ser acrescentada corretamente. Testes feitos pela equipe mostraram que sem portas NOT no caminho entre os túneis J1 e J2 e seus respectivos ANDs causam problemas no circuito, e a pontuação aumenta arbitrariamente para ambos os jogadores.

3.10. CONSERTANDO PROBLEMAS

Agora que o jogo está praticamente concluído, basta resolvermos dois problemas, são eles:

1. Como impedir que o jogador escolha um display já acertado?
2. Como impedir que o jogador escolha a mesma posição duas vezes no mesmo turno?

Para isso, devemos impedir que o estado do jogo avance em algumas nessas duas situações. Para o primeiro problema, a solução encontrada foi o uso de um multiplexador que recebe os valores de cada registrador que indicam se um número já foi acertado ou não.

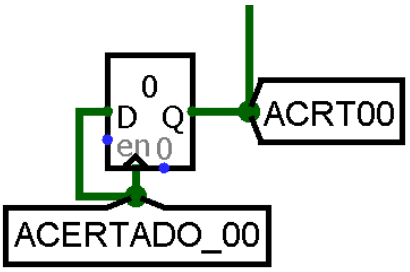

Figura 23: Valor do registrador de acerto.

O túnel nomeado de ACRTab, sendo a o número da linha e b o número da coluna, será usado no multiplexador mostrado na figura abaixo.

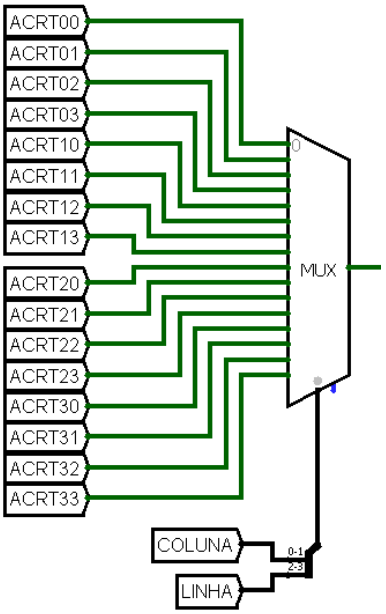

Figura 24: O multiplexador dos números acertados

De acordo com a linha e a coluna que o jogador está posicionado, o multiplexador vai selecionar o número. Caso este número já tenha sido acertado, o valor da saída será 1, caso contrário, o valor da saída será 0.

A solução para o segundo problema é verificar se, durante o ESTADO_1, o estado da segunda escolha, a linha e a coluna sejam respectivamente iguais a LINHA_1 e COLUNA_1. Em outras palavras, se o jogador quer escolher o mesmo número da primeira escolha durante a sua segunda escolha.

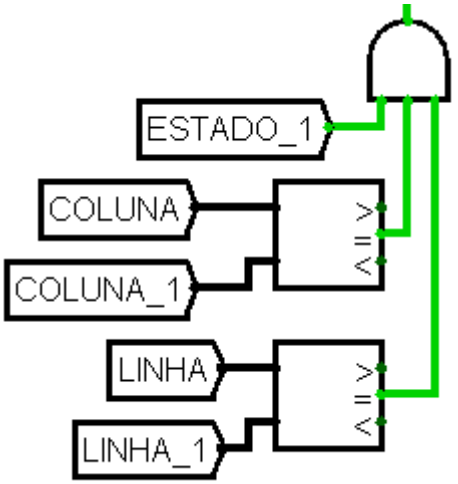

Figura 25: Escolhas iguais.

Por fim, basta juntar os circuitos, como mostrado na figura a seguir:

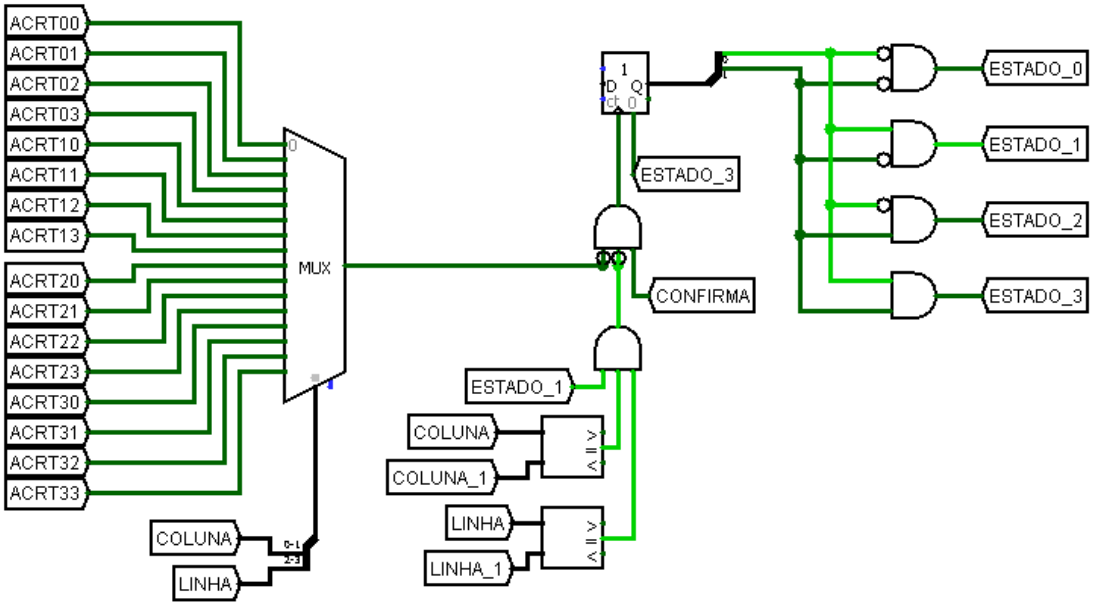

Figura 26: Mudança de estados definitiva.

Agora que atualizamos a máquina de estados, ela só atualiza se satisfazer as três condições. O botão CONFIRMA foi pressionado E o número NÃO foi acertado ainda E a segunda escolha NÃO é igual a primeira escolha. Com isso, o circuito do jogo da memória está concluído.