Trabalho Parcial I

Sistemas Digitais

Prof. Iaçanã Ianiski Weber

Desenvolvido pelo Aluno Leonardo Chou da Rosa

O algoritmo desenvolvido tem como objetivo determinar se uma coordenada fornecida ao circuito está localizada dentro de uma ‘sala’. Uma sala é definida como uma região de um mapa circulada por ‘paredes’:

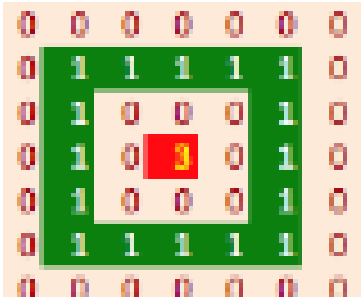

Figura 1: uma coordenada destacada em vermelho sendo circulada por uma sala

Características notáveis de uma sala:

- Contém paredes que formam um quadrado/retângulo ao redor de uma área específica
- As paredes são representadas por um ‘1’ no mapa, e os ‘0’s são regiões sem paredes
- Uma sala não pode ter o mesmo tamanho de outra sala no mapa
- O quadrado/retângulo tem que estar “fechado”, ou seja, não pode ter buracos na parede

Com isso em mente, o algoritmo deve seguir algumas etapas:

- Receber e guardar as coordenadas sendo avaliadas
- Percorrer o mapa no eixo x e eixo y para determinar se a coordenada está cercada por paredes
- Percorrer a parede para determinar se não há buracos na sala
- Calcular e guardar as dimensões da sala
- Comparar o valor calculado com os valores definidos da sala

Para realizar isso, é necessário desenvolver uma máquina de estados que exerce cada etapa do circuito em ordem, terminando a execução caso houver alguma falha. A máquina de estados que eu desenvolvi segue o padrão a seguir:

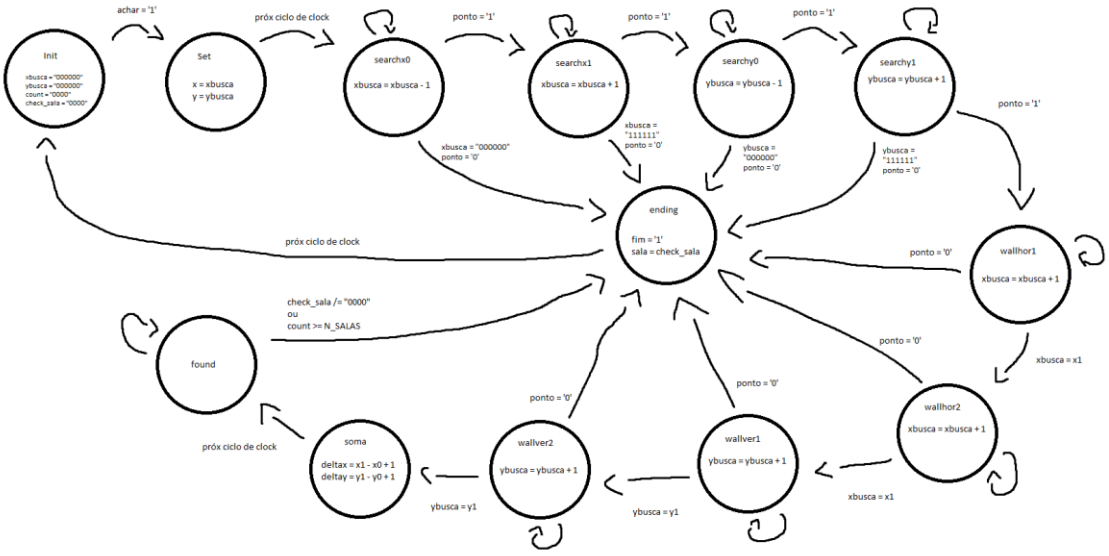

Figura 2: Diagrama da máquina de estados

A máquina de estados contém 13 estados que alternam depende de certas condições:

- Init -> estado que inicia as variáveis e determina o começo da execução do circuito
- Set -> estado que recebe as coordenadas de pesquisa e armazena elas em dois sinais
- Searchx0 -> estado que percorre o mapa horizontalmente para a esquerda em busca de uma parede; quando uma parede é encontrada, o valor é armazenado no sinal x0
- Searchx1 -> estado que percorre o mapa horizontalmente para a direita em busca de uma parede; quando uma parede é encontrada, o valor é armazenado no sinal x1
- Searchy0 -> estado que percorre o mapa verticalmente para cima em busca de uma parede; quando uma parede é encontrada, o valor é armazenado no sinal y0
- Searchy1 -> estado que percorre o mapa verticalmente para baixo em busca de uma parede; quando uma parede é encontrada, o valor é armazenado no sinal y1
- Wallhor1 -> estado que percorre a parede de cima horizontalmente em busca de buracos na parede
- Wallhor2 -> estado que percorre a parede de baixo horizontalmente em busca de buracos na parede
- Wallver1 -> estado que percorre a parede da esquerda verticalmente em busca de buracos na parede
- Wallver2 -> estado que percorre a parede da direita verticalmente em busca de buracos na parede
- Soma -> estado que acham os valores de delta x e delta y (delta x = x1 – x0 + 1; delta y = y1 – y0 + 1)
- Found -> estado que compara os valores de delta x e delta y com as dimensões das salas
- Ending -> estado que encerra a execução do programa e reseta as variáveis (para iniciar uma nova testagem)

As condições que fazem elas alternarem de estado são:

- Init -> quando a entrada ‘achar’ receber o valor de ‘1’, o próximo estado vira o estado set
- Set -> após os valores serem armazenados (ciclo de clock), o estado muda para o estado searchx0
- Searchx0 -> quando uma parede é encontrada (ponto = ‘1’), o estado é trocado para o estado searchx1; caso a variável de busca chegue ao final do mapa (xbusca = “000000”) e não tem uma parede nessa coordenada (ponto = ‘0’), o próximo estado é o estado ending (não tem uma sala); enquanto essas condições não são satisfeitas, permanece no estado searchx0.
- Searchx1 -> quando uma parede é encontrada (ponto = ‘1’), o estado é trocado para o estado searchy0; caso a variável de busca chegue ao final do mapa (xbusca = “111111”) e não tem uma parede nessa coordenada (ponto = ‘0’), o próximo estado é o estado ending (não tem uma sala); enquanto essas condições não são satisfeitas, permanece no estado searchx1.
- Searchy0 -> quando uma parede é encontrada (ponto = ‘1’), o estado é trocado para o estado searchy1; caso a variável de busca chegue ao final do mapa (ybusca = “000000”) e não tem uma parede nessa coordenada (ponto = ‘0’), o próximo estado é o estado ending (não tem uma sala); enquanto essas condições não são satisfeitas, permanece no estado searchy0.
- Searchy1 -> quando uma parede é encontrada (ponto = ‘1’), o estado é trocado para o estado wallhor1; caso a variável de busca chegue ao final do mapa (ybusca = “111111”) e não tem uma parede nessa coordenada (ponto = ‘0’), o próximo estado é o estado ending (não tem uma sala); enquanto essas condições não são satisfeitas, permanece no estado searchy1.
- Wallhor1 -> se houver um buraco na parede (ponto = ‘0’), o estado é trocado para o estado ending (não é uma sala); se a parede inteira for percorrida e não houve buracos (xbusca = x1), o estado é trocado para o estado wallhor2; enquanto nenhuma condição for satisfeita, fica no estado atual.
- Wallhor2 -> se houver um buraco na parede (ponto = ‘0’), o estado é trocado para o estado ending (não é uma sala); se a parede inteira for percorrida e não houve buracos (xbusca = x1), o estado é trocado para o estado wallver1; enquanto nenhuma condição for satisfeita, fica no estado atual.
- Wallver1 -> se houver um buraco na parede (ponto = ‘0’), o estado é trocado para o estado ending (não é uma sala); se a parede inteira for percorrida e não houve buracos (ybusca = y1), o estado é trocado para o estado wallver2; enquanto nenhuma condição for satisfeita, fica no estado atual.
- Wallver2 -> se houver um buraco na parede (ponto = ‘0’), o estado é trocado para o estado ending (não é uma sala); se a parede inteira for percorrida e não houve buracos (ybusca = y1), o estado é trocado para o estado soma; enquanto nenhuma condição for satisfeita, fica no estado atual.
- Soma -> o estado troca para o estado found após a soma de delta x e delta y (após um ciclo de clock)
- Found -> se as dimensões de uma sala forem iguais às dimensões de delta x e delta y (check_sala /= “0000”; encontrou o número da sala), ou se o programa percorreu todas as salas não achou uma sala com dimensões iguais às dimensões de delta x e

delta y (count >= N_SALAS), o próximo estado é o estado ending; enquanto nenhuma das duas condições forem satisfeitas, fica no estado found.

- Ending -> o estado troca para o estado init após os valores de count, ybusca, xbusca e check_sala forem zerados (após um ciclo de clock)

Máquina de estados em prática:

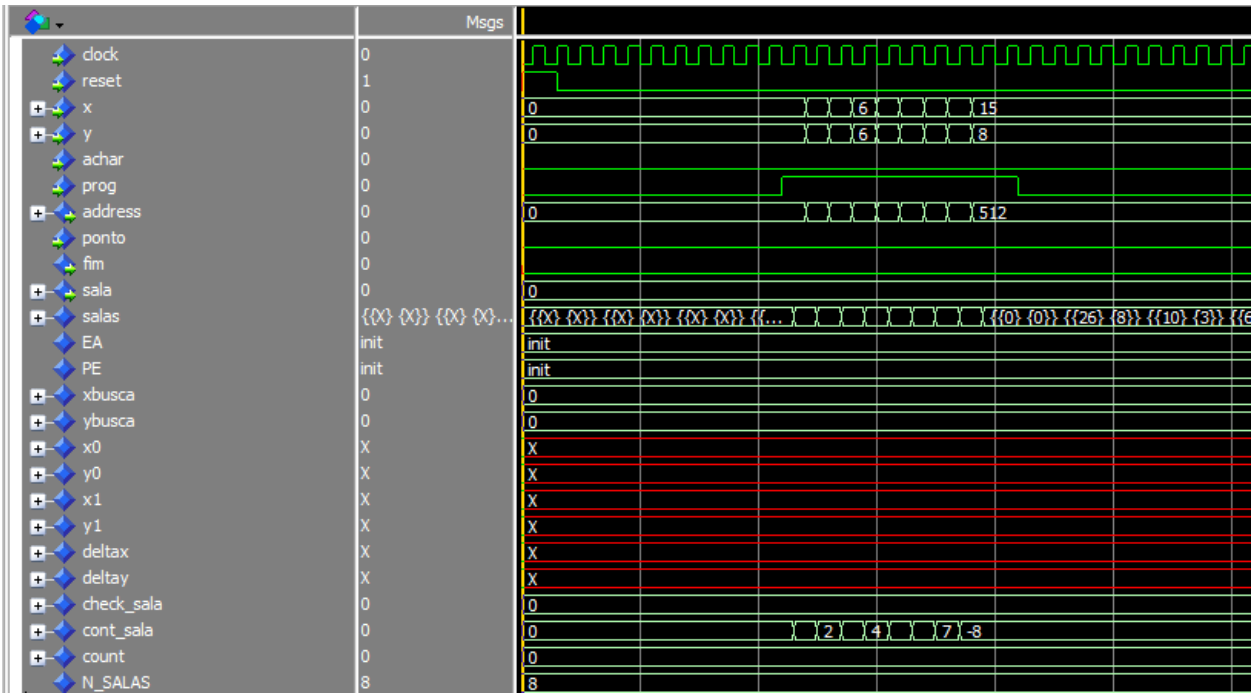

Figura 3: estado init na simulação

Os valores da sala estão sendo definidos no estado init, inicializados pela entrada ‘prog’.

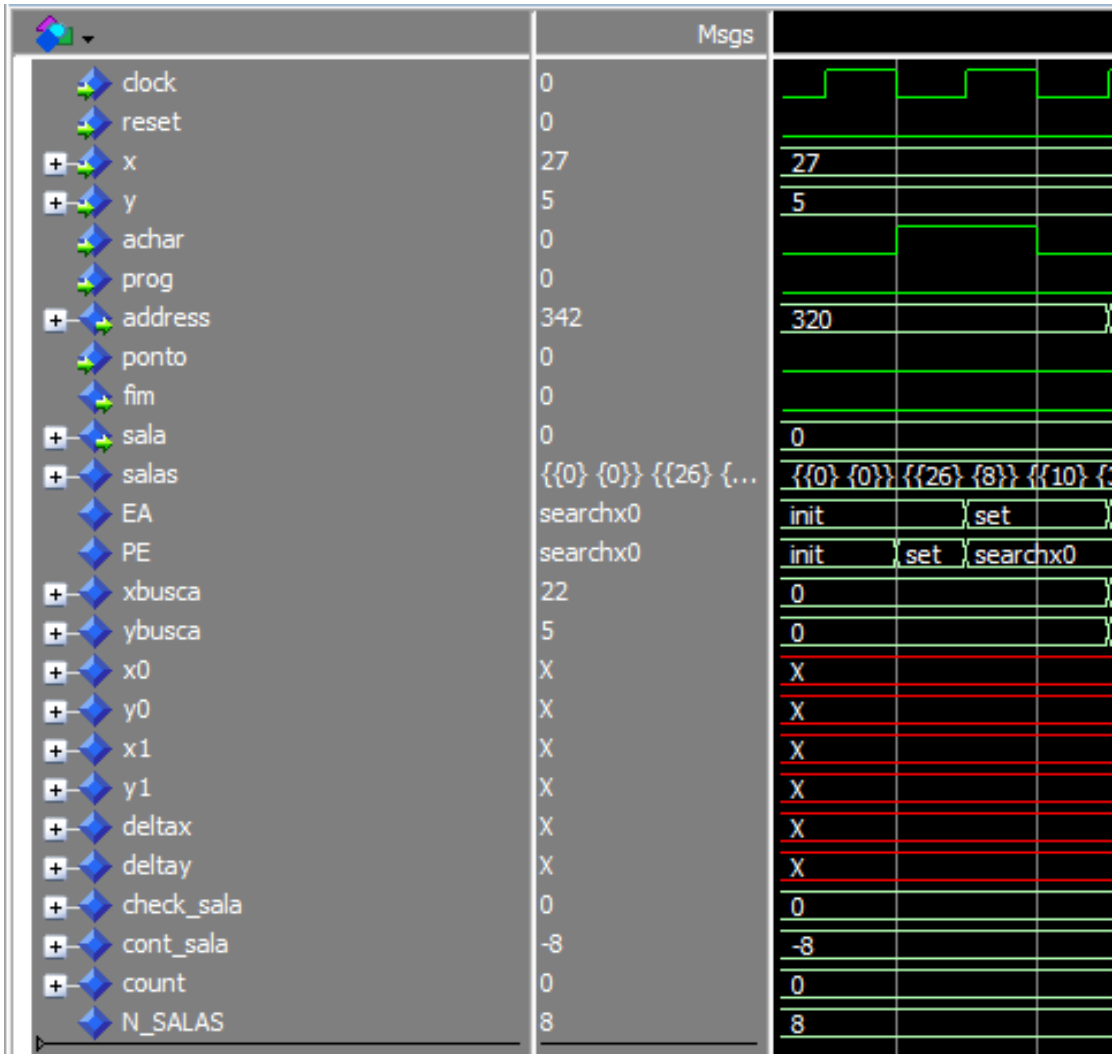

Figura 4: estado set na simulação

Quando o a entrada ‘achar’ é ligada, o próximo estado é o estado set, que coloca os valores de x e y (27 e 5, respectivamente) nos sinais de busca (xbusca e ybusca)

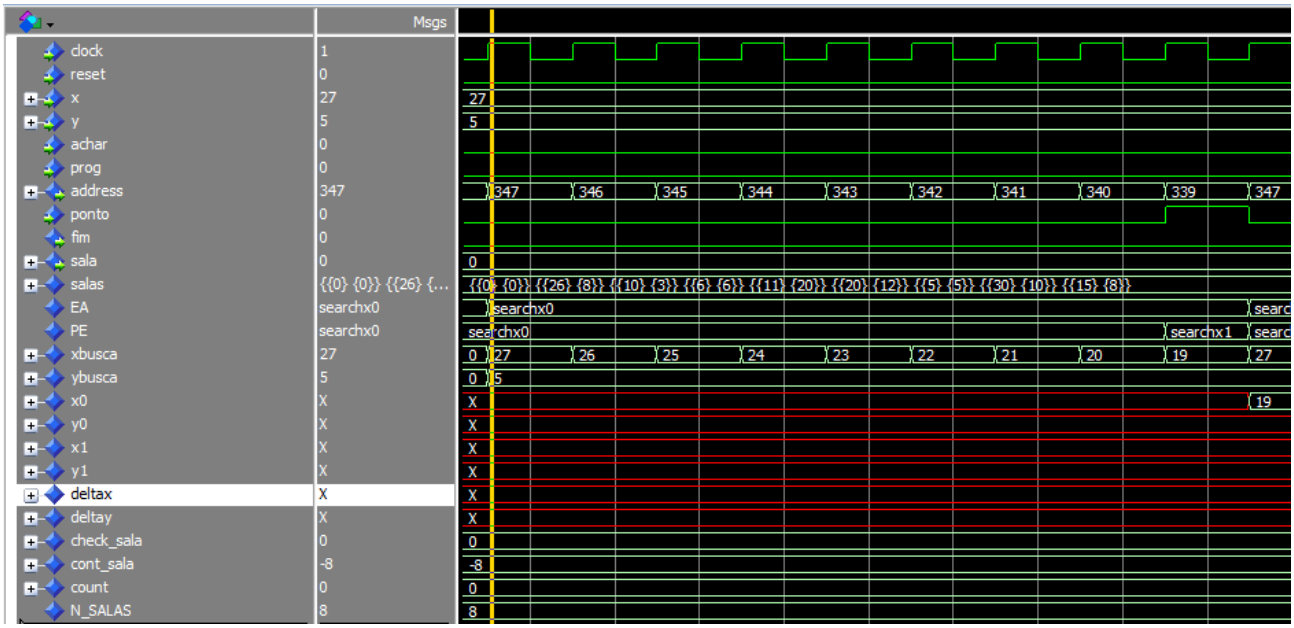

Figura 5: estado searchx0 na simulação

Quando o estado searchx0 é inicializado, observa-se que os sinais de xbusca e ybusca são 27 e 5 (definidos pelo estado set). O sinal xbusca então é decrementado até o sinal de ponto virar 1 (xbusca = 19), e esse valor é armazenado na variável x0. O estado então muda para o estado searchx1, pois uma parede foi encontrada.

Figuras 6 e 7: estado searchx1 na simulação

O estado searchx1 funciona da mesma maneira que o estado searchx0, mas ao invés de decrementar o valor de xbusca, o valor é incrementado. Quando uma parede é encontrada, o valor é guardado no sinal x1 (-20). O estado então muda para o estado searchy0.

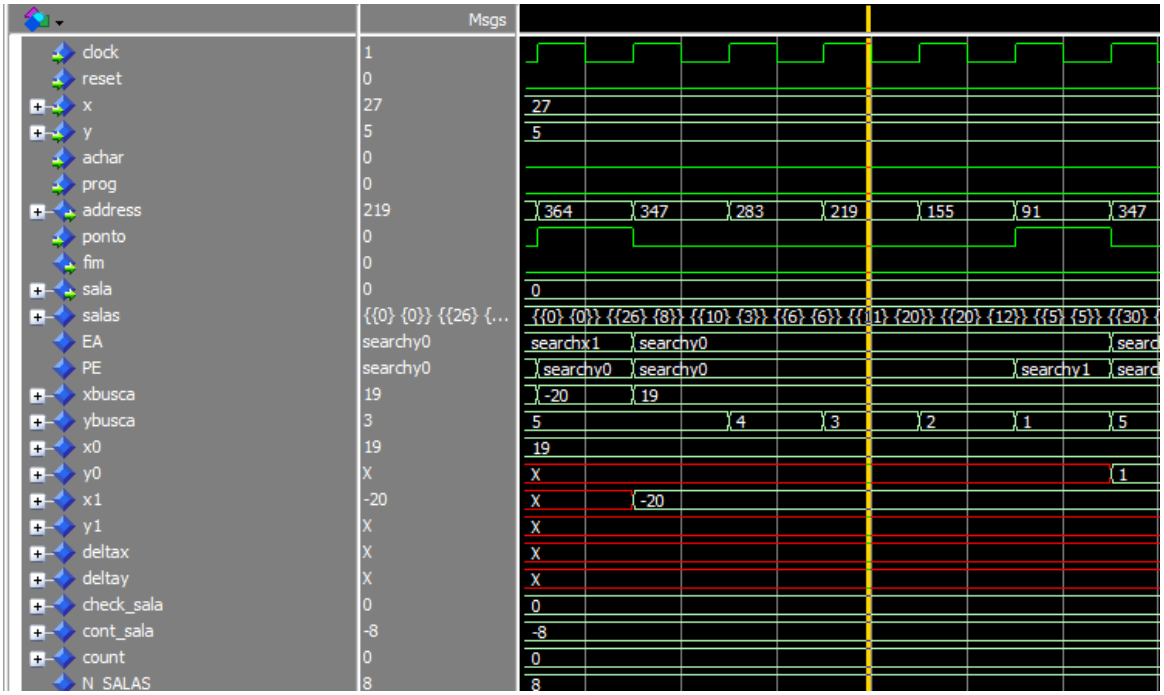

Figura 8: estado searchy0 na simulação

O estado searchy0 funciona da mesma forma que os estados searchx0 e searchx1, mas para o eixo y. O estado searchy0 decrementa o sinal ybusca até encontrar uma parede, onde o valor é guardado no sinal y0 (1).

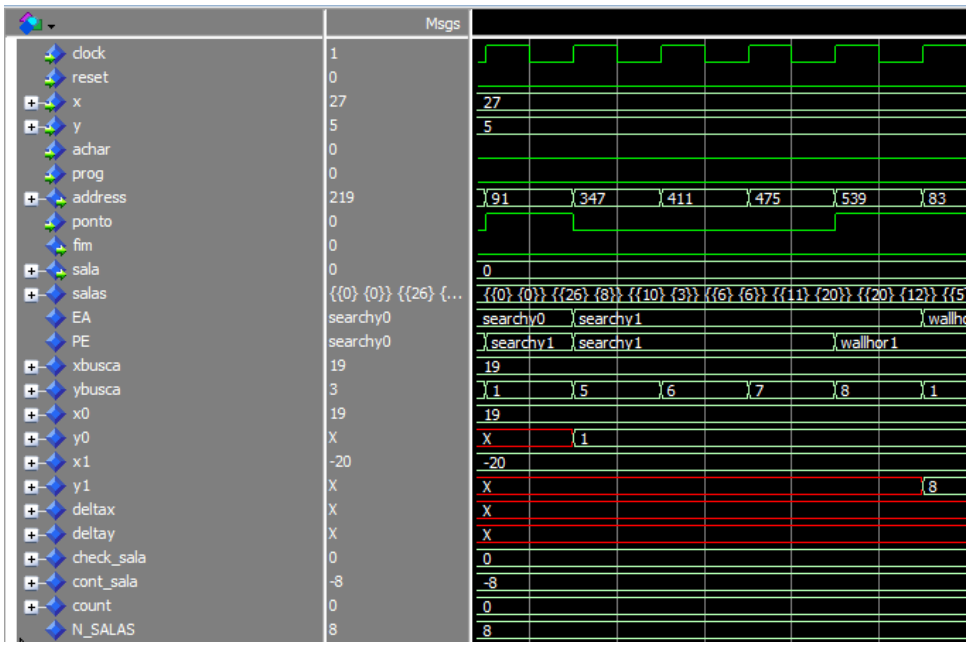

Figura 9: estado searchy1 na simulação

O estado serachy1 decrementa o valor de ybusca e guarda esse valor no sinal y1 (8). Após o sinal ser encontrado, ele vai para o estado wallhor1.

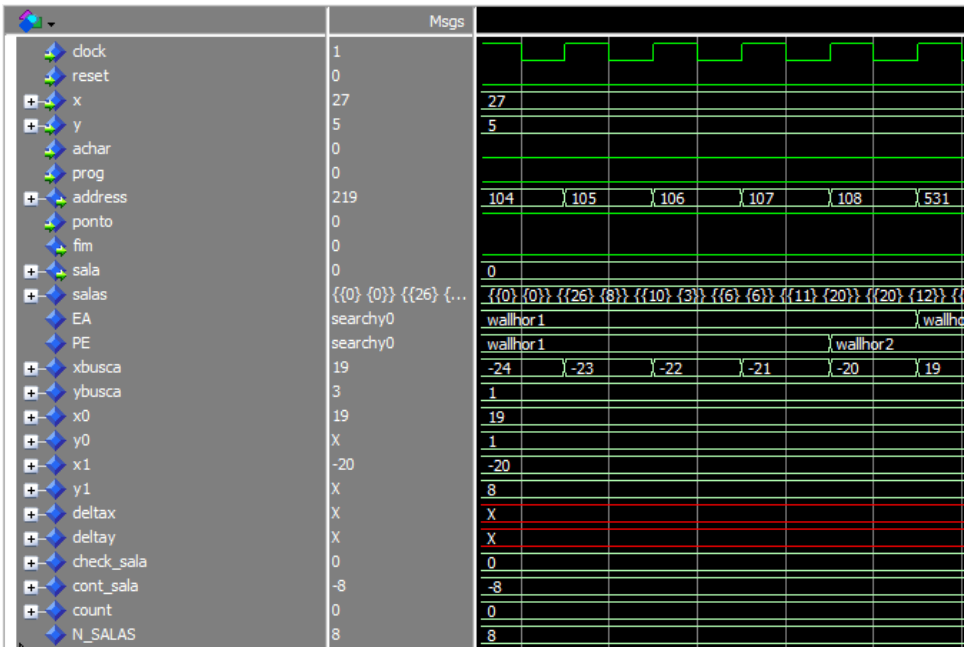

Figura 10: estado wallhor1 na simulação

O estado wallhor1 percorre a parede horizontalmente começando na coordenada (x0, y0) e termina na coordenada (x1, y0). Na simulação, quando o valor de xbusca foi decrementado até o valor de x1 (-20) e não houve uma instância de buraco (ponto = '0'), o estado muda para o estado de wallhor2, onde o mesmo é executado, mas para a coordenada (x0, y1).

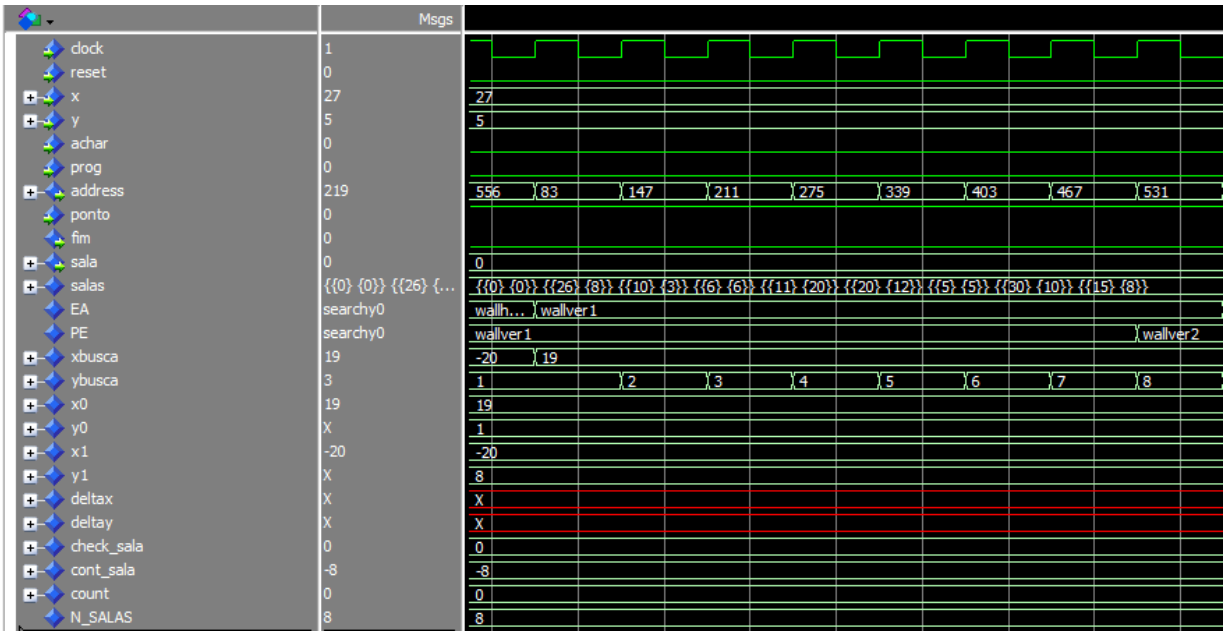

Figura 11: estado wallver1 na simulação

O estado wallver1 faz a mesma funcionalidade do que os estados wallver2, wallhor1 e wallhor2, mas ele percorre a parede verticalmente começando na coordenada (x0, y0). Quando o valor de ybusca é igual ao valor de y1(8), e não há buracos (ponto = '0'), o estado é trocado para o estado wallver2, onde o mesmo é executado para a coordenada (x1, y0).

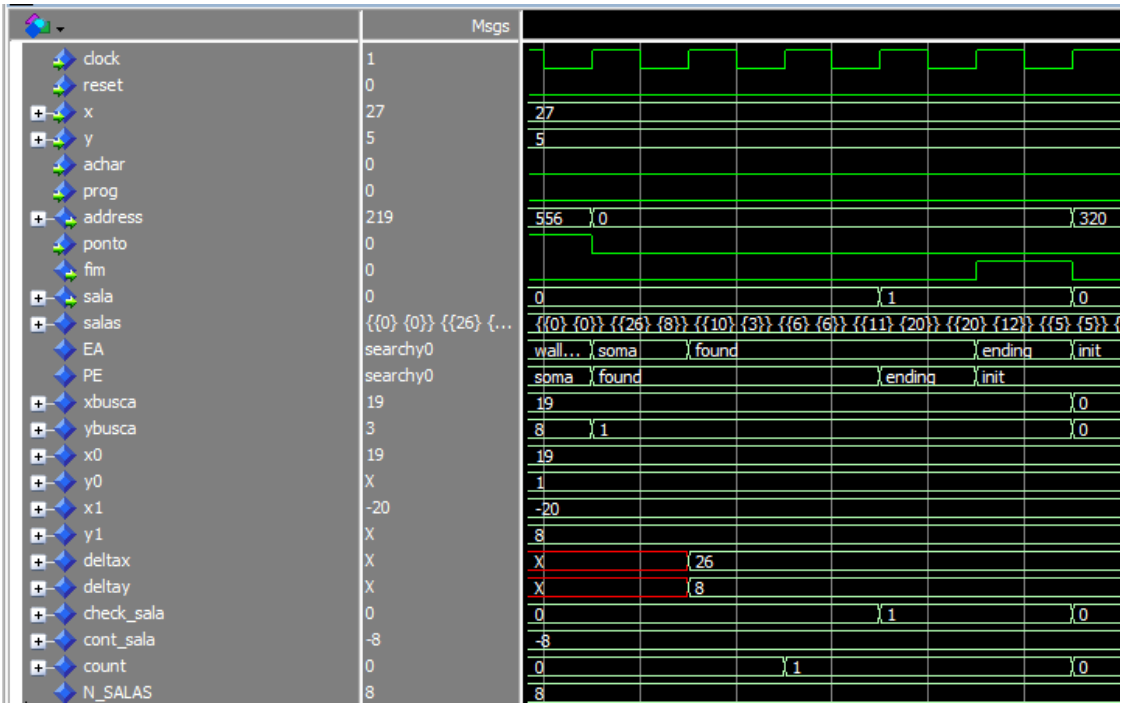

Figura 12: estados soma, found e ending na simulação

Após o circuito determinar que não há buracos na parede, ele muda para o estado soma, onde os valores de deltax e deltay são calculados (26 e 8, respectivamente). Após a soma, eles são guardados no sinal deltax e deltay e o estado muda para o estado found.

No estado found, o valor de count é incrementado até uma sala for achada, ou até não houver mais salas. No caso da simulação, o count incrementou uma vez (1) e os valores de deltax e deltay foram comparados com os valores x e y da sala 1 (26 e 8). O algoritmo determinou que os valores de deltax e deltay são iguais aos valores da sala 1. Portanto, a coordenada recebida no início da execução pertence dentro da sala 1 do mapa.

Após a sala for encontrada, o próximo estado é o estado ending, que reseta as variáveis e começa o processo novamente.