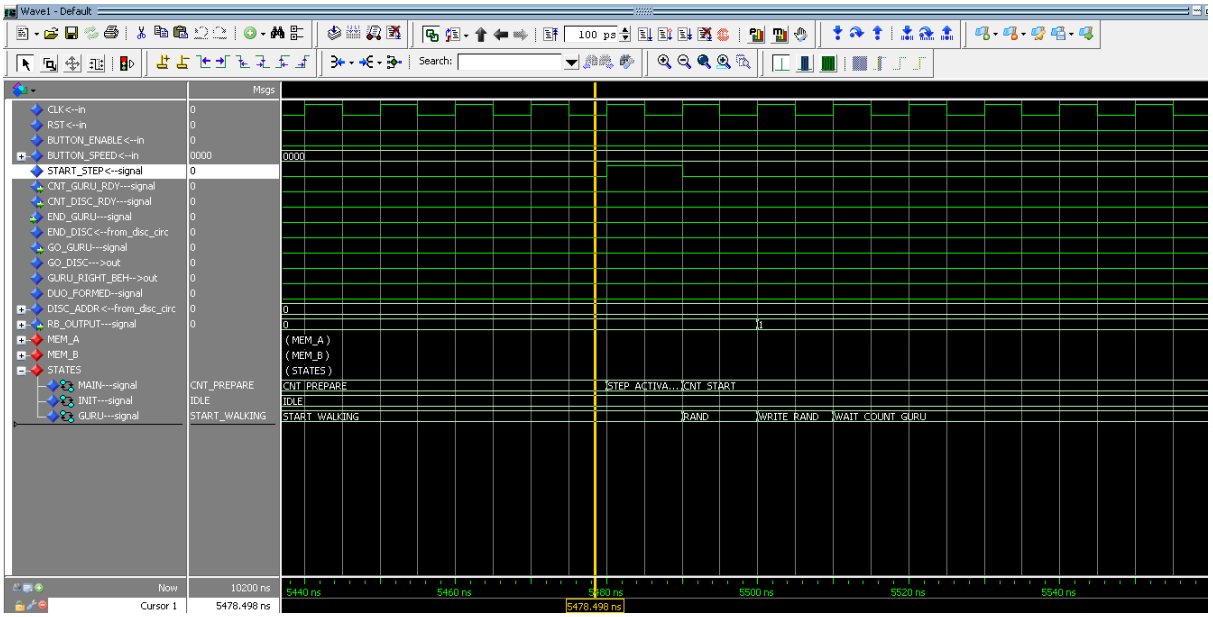
# Vinícius de Barros Silva  10335913

**a) Para a FSM Guru**

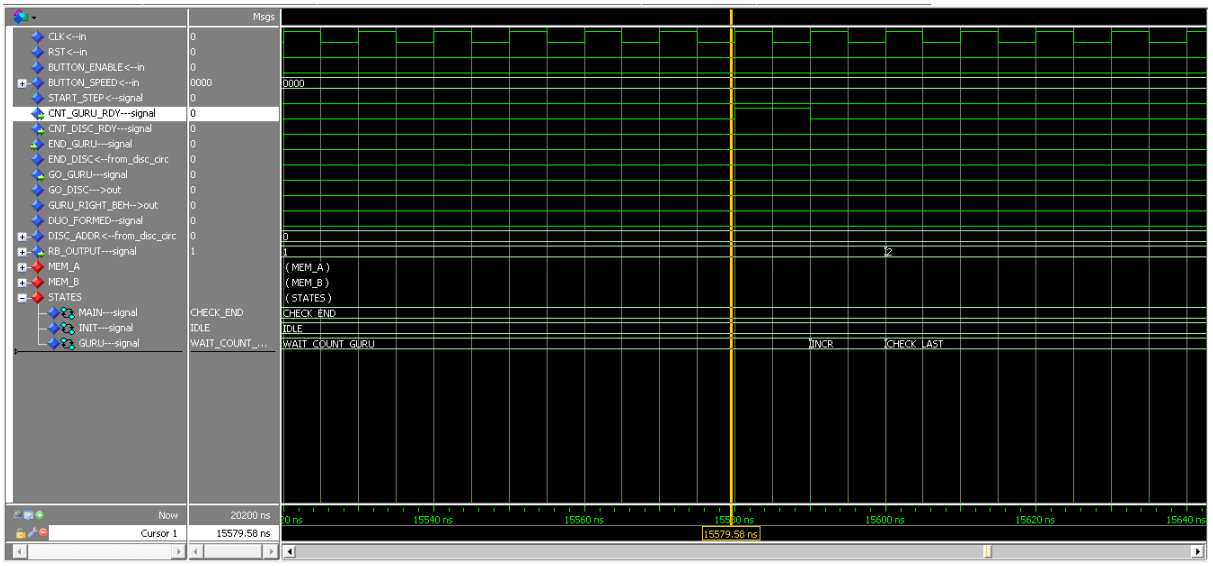
**1c) captura: imagem do Wave onde fique evidente os 3 estados de espera da FSM Guru e de suas relações com os sinais respectivos que os liberam**
**1t) texto: para cada evento acima, explicar o ocorrido de acordo com a interação do circuito com o ambiente externo.**

START_WALKING - START STEP




(Imagem 1)
Após a inicialização do tabuleiro, a FSM_GURU recebe o sinal START_STEP = 1 isso faz com que a máquina mude para o estado RAND para receber o número aleatório para determinar a posição inicial do guru, para o usuário nada acontece, o tabuleiro está em branco nesse momento.

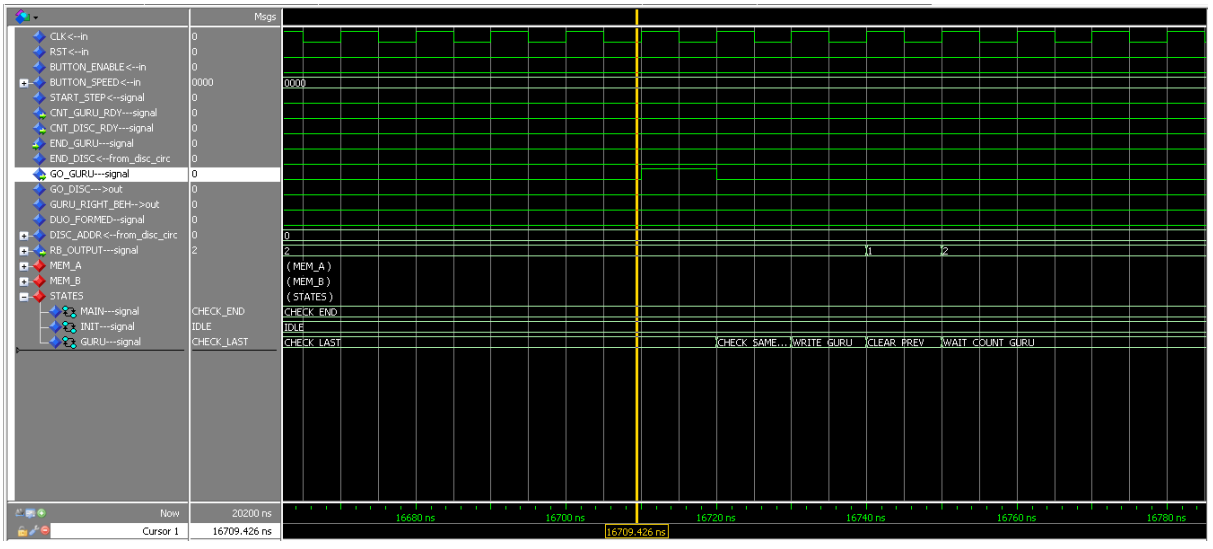
WAIT_COUNT_GURU - CNT_GURU_RDY




(Imagem 2)

Nesse estado a FSM_GURU aguarda o sinal CNT_GURU_RDY para poder dar um passo com o guru, quando CNT_GURU_RDY = 1, teremos a mudança para o estado INCR onde incrementamos o contador para mudar a posição do guru. Do ponto de vista do usuário, até esse momento o guru apareceu na posição aleatória determinada.

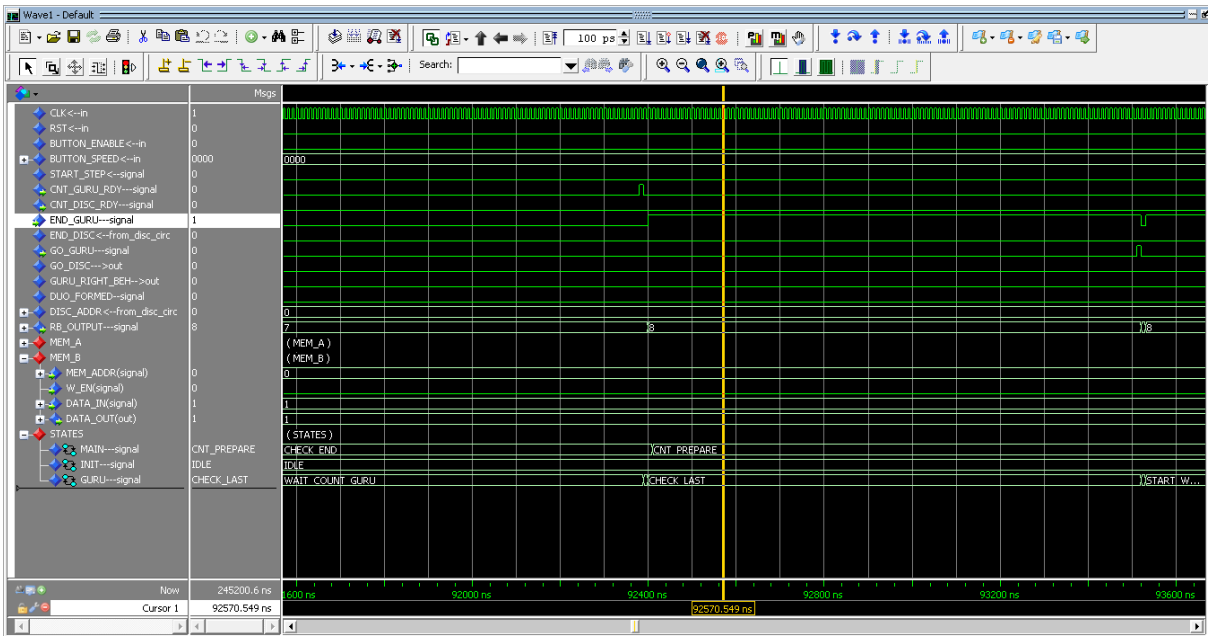
CHECK_LAST - GO_GURU e END_GURU




(Imagem 3)

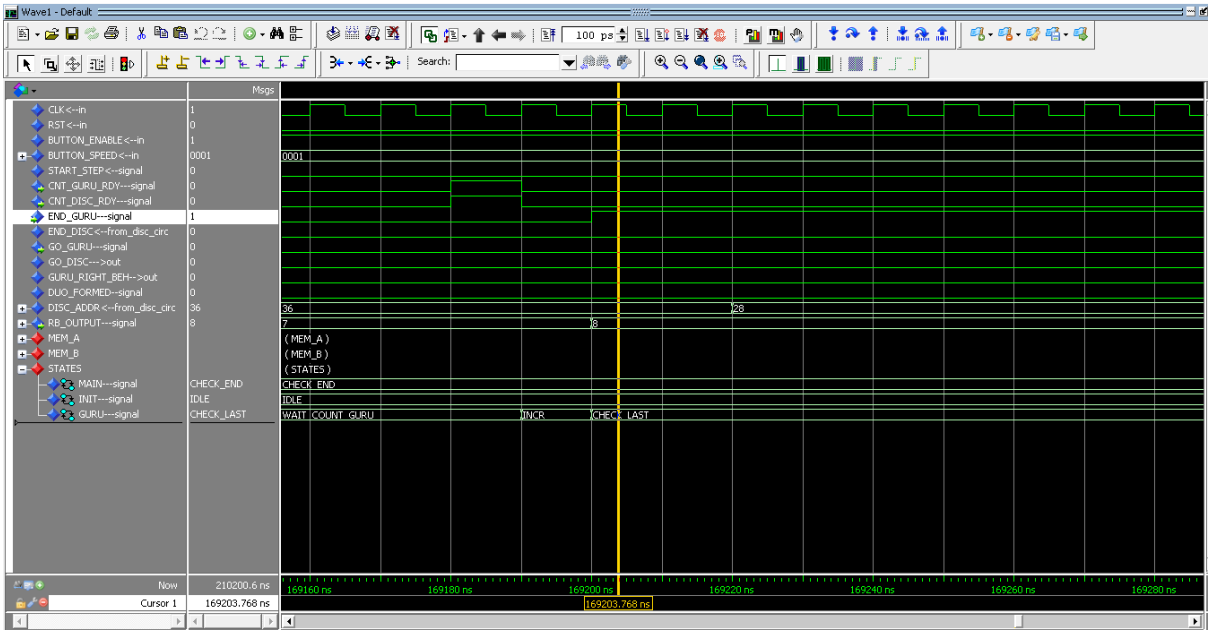
Aqui a FSM_GURU aguarda o sinal GO_GURU =1 para junto ao sinal END_GURU decidir qual o próximo estado, isto é, se escreve o guru na posição incrementada, se escreve duo caso haja encontro ou se o guru utrapassou a borda do tabuleiro e uma nova rodada deve iniciar. Do ponto de vista do usuário, essa transição de estado trata dos passos do guru, se ele andou uma casa, se encontrou o discípulo ou se passou do tabuleiro.

**2c) captura: imagem dos eventos em SIM-2 e SIM-3 com os estados de finalização da iteração e o sinais que os determinam.**



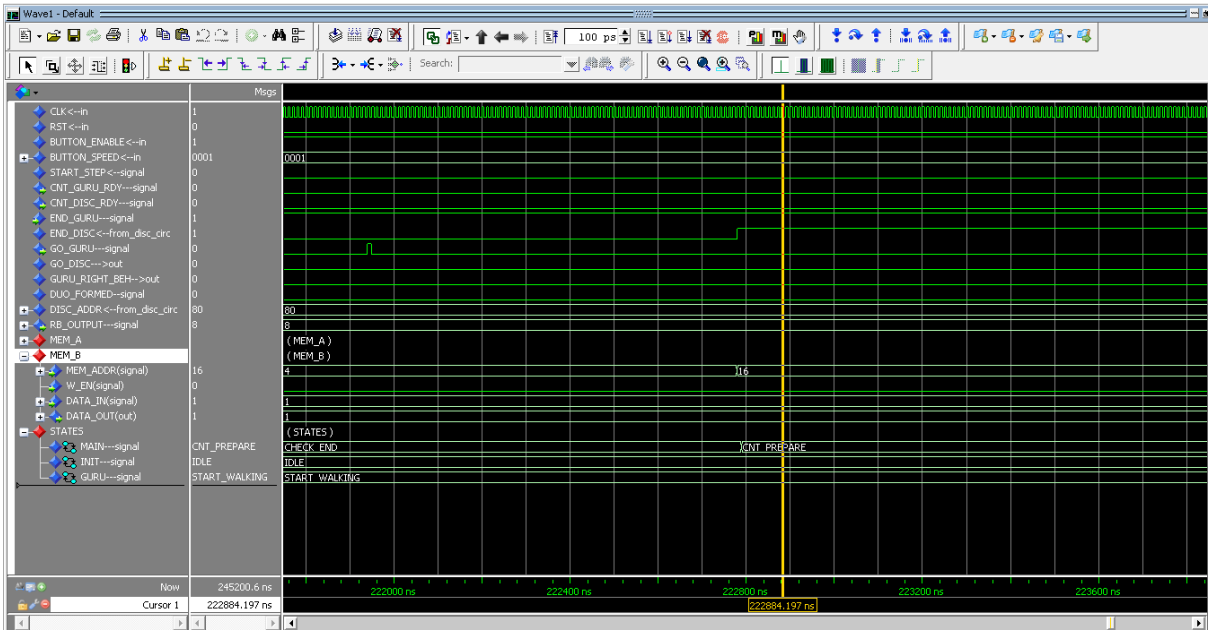

(Imagem 4)

Momento em que o GURU incrementa para 8 e assim END_GURU = 1 e a FSM GURU passa ao estado LAST indicando que houve overflow, isto é, o guru passou pela borda do tabuleiro sem o discípulo ter sido ativado e uma nova rodada deve iniciar.




(Imagem 5)

Momento em que END_GURU = 1 e espera a FSM_DISC chegar ao fim




(Imagem 6)

Momento em que END_DISC = 1 e a FSM_DISC chega ao fim da rodada após passar pela casa 4. Importante notar que nessa simulação o guru termina antes do discípulo, então o FSM_GURU aguarda o discípulo terminar para uma nova rodada.

**2t) texto: Inclua o trecho do código VHDL da FSM Main onde estes eventos são determinados e explicar o ocorrido**

```
when CHECK_END      =>  if (end_of_guru = '1') AND (end_of_disc = '1') AND (duo_formed = '1') AND (en_disc = '1') then
                            NEXT_STATE <= HIT_POINT;
                        elsif ((end_of_guru = '1') AND (end_of_disc = '1') AND (duo_formed = '0') AND (en_disc = '1'))
                              OR ((end_of_guru = '1') AND (en_disc = '0') AND (duo_formed = '0'))then
                            NEXT_STATE <= CNT_PREPARE;
                        else
                            NEXT_STATE <= CHECK_END;
                        end if;
```

Trecho de código que determina o término da rodada.

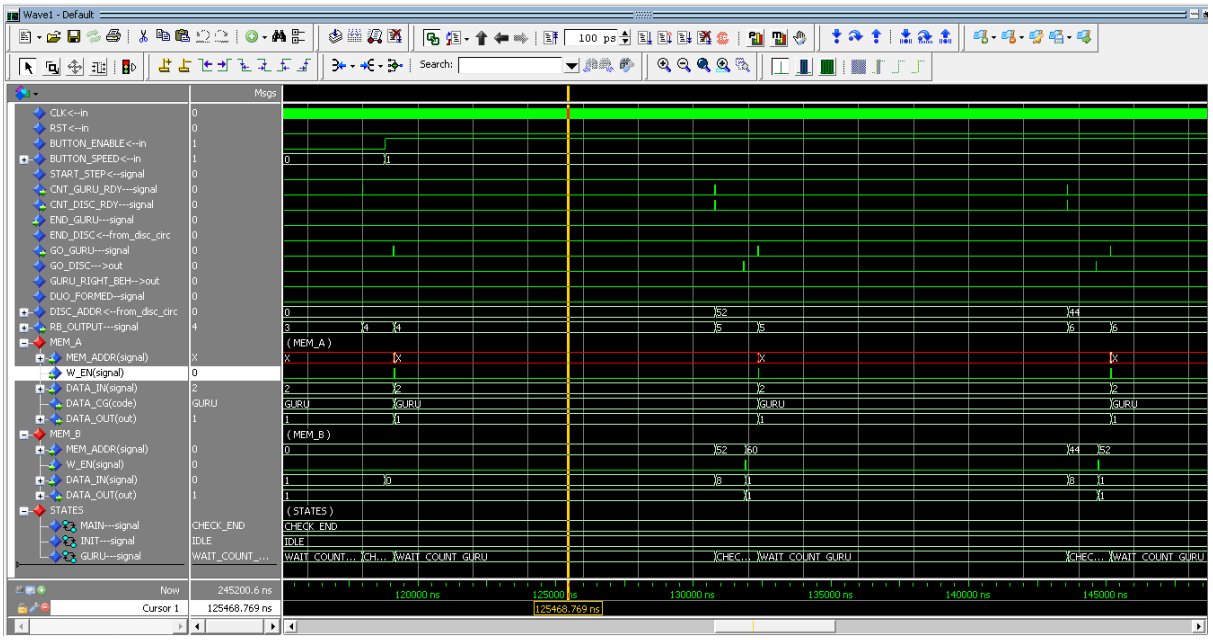
Para SIM-2, o guru termina e teremos (end_of_guru = '1') AND (en_disc = '0') AND (duo_formed = '0') assim para esse caso o próximo estado é CNT_PREPARE para uma nova rodada.

para SIM-3, o guru termina indicando END_GURU = 1 e a FSM_GURU aguarda enquanto o discípulo não termina,quando o discípulo envia o sinal de término END_DISC = 1 e DUO_FORMED =0 teremos (end_of_guru = '1') AND (end_of_disc = '1') AND (duo_formed = '0') AND (en_disc = '1'), assim a FSM MAIN troca do estado CHECK_END para CNT_PREPARE para nova rodada.

**b) Para a interação com o Circuit Disciple**

**3c) captura: imagem dos eventos em SIM-3 onde o sinal enable é ativado. Indicar o primeiro efeito que isto causa nos sinais provenientes do circuito do discípulo.**

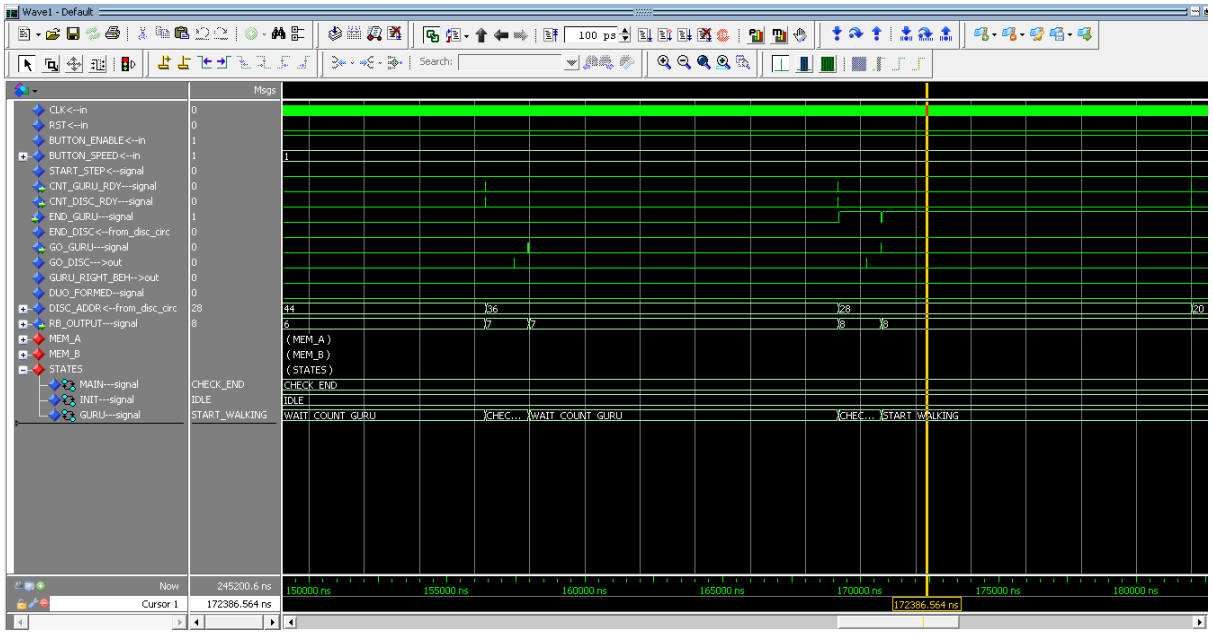
**3t) Explique a correlação entre o enable e o sinal do discípulo.**




(Imagem 7)

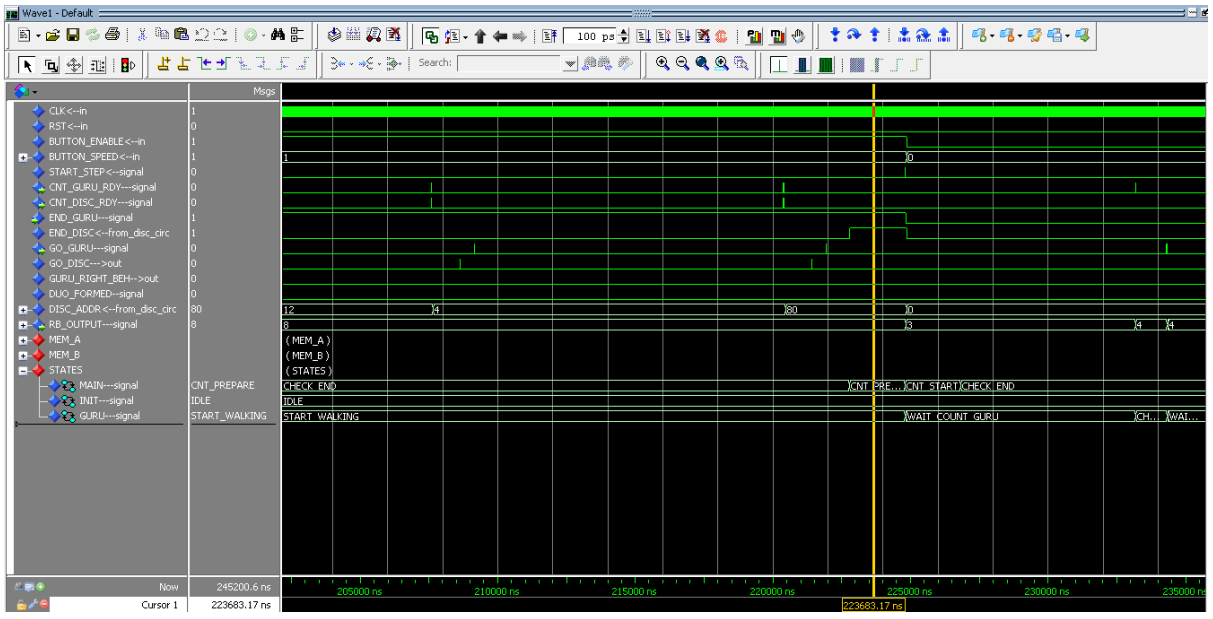
O sinal de Enable é ativo quando o botão é pressionado, isso indica que o jogador deu início ao funcionamento do discípulo. Junto ao sinal de enable o primeiro efeito é a alteração do valor do sinal de velocidade do discípulo , nesse caso BUTTON_SPEED = 1. Com o enable ativado e a velocidade definida, o sinal CNT_DISC_RDY pode ser disparado de acordo com a velocidade para computar os passos do discípulo, como também, o sinal go_disc agora entra em ação para o estado “CHECK_LAST” do discípulo tomar a decisão de próximo estado da máquina de estados do discípulo, junto ao sinal END_OF_DISC.

**4c) captura: imagem do evento SIM-3, com a chegada do sinal end_of_disciple e a geração do end_of_guru. Identifique o valor dos respectivos endereços na memória que causaram a ativação destes sinais.**




(Imagem 8)

O endereço de memória que causou o disparo de END_GURU foi “8”



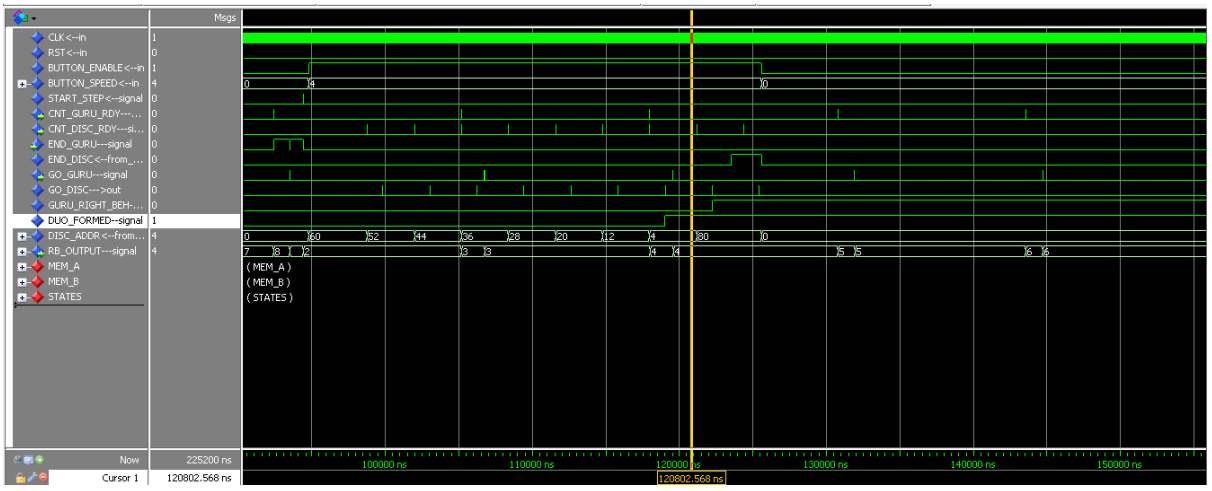

(Imagem 9)

o endereço de memória que causou o disparo de END_DISC foi “80”

**4t) Considerando o número aleatório da SIM-3 para o guru, faça um cálculo aproximado do instante da partida do discípulo (acionamento do enable e do definição do valor de sys_speed) para que haja o encontro entre o guru e o discípulo, e em que casa (endereço) isto ocorreria. Sugestão: aplique este cálculo no testbench para verificar o sinal de duo_formed.**

Podemos disparar o discípulo junto ao disparo do guru em 94785 ns e utilizar a velocidade 4x para causar o encontro com o guru na casa 4, isso pode ser

observado no wave da simulação gerada com as devidas modificações no testbench.




(Imagem 10)

```
145         ------------------------------------------------------------------------------------
146
147         -- SIM 3 : condicao de jogo sem conflito entre guru e discipulo - deve-se observar toda
148             -- Velocidades iguais para o guru e o discipulo
149
150
151             --wait for 24us;     --tempo para o guru dar alguns passos...
152
153
154             checkWC('0','1','0', 60, 0, 4, 60, 0);   -- durante a caminhada do guru, em 118 us aprox.
155
156                                                 -- o circuito do discipulo aguarda o cnt_disc_rea
157                                                 --  como o discipulo comeca atrasado, nao ha conf
158                                                 -- a partir deste momento o referee se encarrega
159
160             wait for 4us;      --chega no ponto de o discipulo enviar os dados
161
162             checkWC('0','1','0', 52, 60, 4, 52, 8);   --
163
164             wait for 1.1us;      --chega no ponto de o discipulo escrever os dados
165
166             checkWC('0','1','1', 52, 60, 4, 52, 8);   --
167
168             wait for CLK_PERIOD;
169
170             checkWC('0','1','1', 52, 60, 4, 60, 1);   --
171
172             wait for CLK_PERIOD;
173
174             checkWC('0','1','0', 52, 60, 4, 60, 1);   --
175
176             for i in 1 to 6 loop
177
178                 wait for 2.08us;      --chega no ponto de o discipulo enviar os dados
179
180                 checkWC('0','1','0', 52-i*8, 60-i*8, 4 , 52-i*8, 8);   --
181
```

```
181
182             wait for 1.1us;      --chega no ponto de o discipulo escrever os dados
183
184             checkWC('0','1','1', 52-i*8, 60-i*8, 4, 52-i*8, 8);   --
185
186             wait for CLK_PERIOD;
187
188             checkWC('0','1','1', 52-i*8, 60-i*8, 4, 60-i*8, 1);   --
189
190             wait for CLK_PERIOD;
191
192             checkWC('0','1','0', 52-i*8, 60-i*8, 4, 60-i*8, 1);   --
193
194         end loop;
195
196
197         wait for 2.08us;      --chega no ponto de o discipulo enviar os dados
198
199         checkWC('0','1','0', 80, 4, 4, 80, 1);   --
200
201         wait for 1.1us;      --chega no ponto de o discipulo escrever os dados
202
203         checkWC('0','1','1', 80, 4, 4, 4, 1);   --
204
205         wait for CLK_PERIOD;
206
207         checkWC('0','1','0', 80, 4, 4, 4, 1);   --
208
209         wait for 1.2us;      --intervalo de tempo até o discipulo gerar o end_of_disc
210
211
212         checkWC('1','1','0', 80, 4, 4, 80, 1);   --
213
214
215         wait for 2.1us;      --intervalo de tempo até o discipulo gerar o end_of_disc
216
217         checkWC('0','0','0', 0, 0, 0, 0, 0);   --
```