**SSC0108 – Prática em Sistemas Digitais**

MEF03 – Final

|  |  |
| --- | --- |
| Nome | N.º USP |
| Bernardo Maia Coelho | 12542481 |
| Eduardo Figueiredo Freire Andrade | 11232820 |
| Gabriel Ribeiro Fonseca de Freitas | 12542651 |

Obs 1: Utilize este arquivo como relatório de laboratório, inserindo as informações a partir da próxima página.

Obs 2: Este Lab é em trio, deverá ser convertido em **PDF** e entregue via Moodle.

Obs 3: **Não** serão aceitos **outros** **formatos**.

Obs 4: Nome do arquivo deverá ser MEF03\_nUSP.pdf onde nUSP é o Número USP do integrante do grupo responsável pela entrega.

**Atividades**

1. Com base nos diagramas e tabelas de transição de estados implemente as máquinas que resolvem o problema da Máquina de Refrigerante e do Elevador usando a integração VHDL com o LogiSIM Evolution.
2. As máquinas de estados devem seguir rigorosamente as regras estabelecidas na Aula8.
3. Faça a Integração com a máscara da DE0-CV e utilize capturas que julgar necessário para mostrar o funcionamento das máquinas.
4. A chave S9 deve ser utilizada como clock da máquina e a chave S0 como Master Reset da máquina. A função Master reset leva a máquina em questão para seu estado inicial ou de repouso.
5. A forma como utilizar os LEDs e displays para demonstrar o funcionamento da máquina é Livre.
6. Apresente o código VHDL utilizado em ambas as máquinas no relatório.
7. O seu arquivo final \*.circ deve de ser adicionado em um drive e o caminho deve estar disponível no relatório para eventuais dúvidas de funcionamento .
8. Deverá ser submetido no Moodle apenas 1 relatório por grupo, nomeando da seguinte maneira: MEF\_03\_xxxx.pdf onde xxxx é o número USP do líder do grupo.
9. Utilize este arquivo para preencher seu relatório.

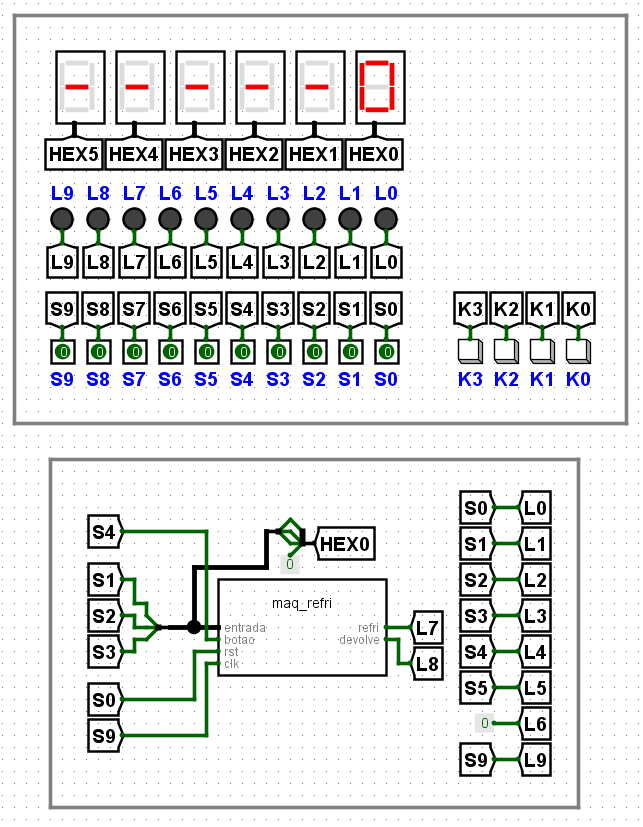
**Respostas**

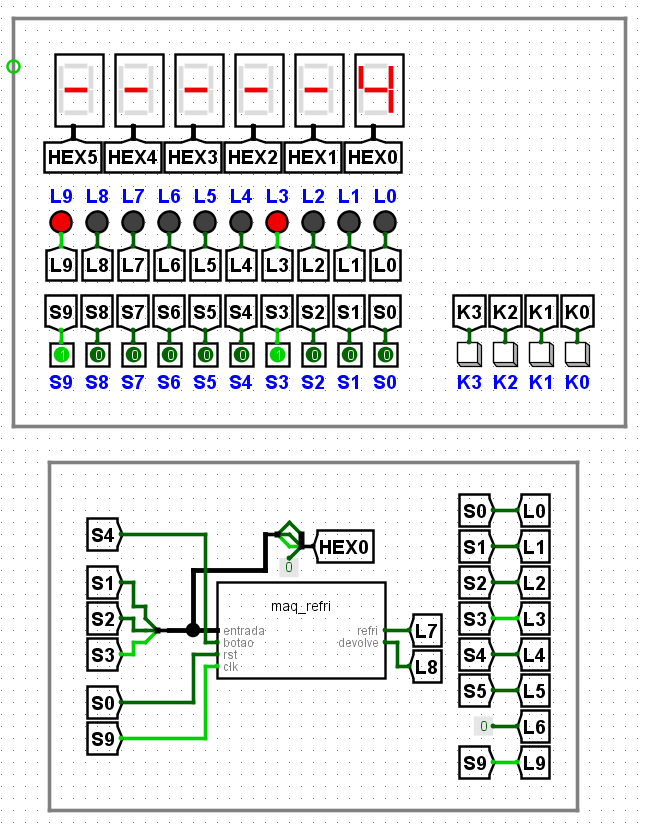
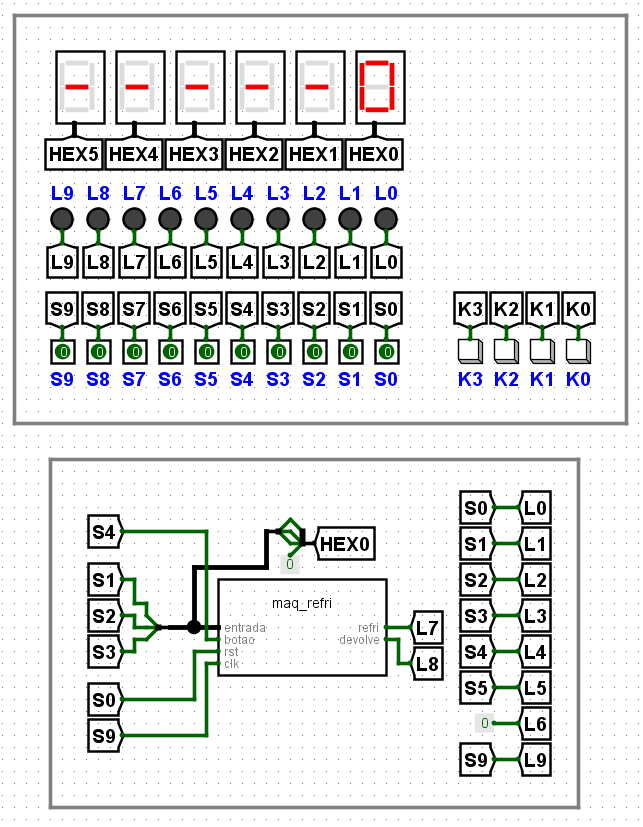
# MÁQUINA DE REFRIGERANTE

O funcionamento interno da máquina de refrigerante leva em consideração uma entrada de 3 bits a qual comunica que tipo de moeda está sendo inserida. A cada clock, considera-se que uma nova moeda, e apenas uma, pode ser inserida; sendo o valor de entrada igual a 0, (000)2, quando nenhuma moeda for inserida. A tabela a seguir mostra o código de cada moeda.

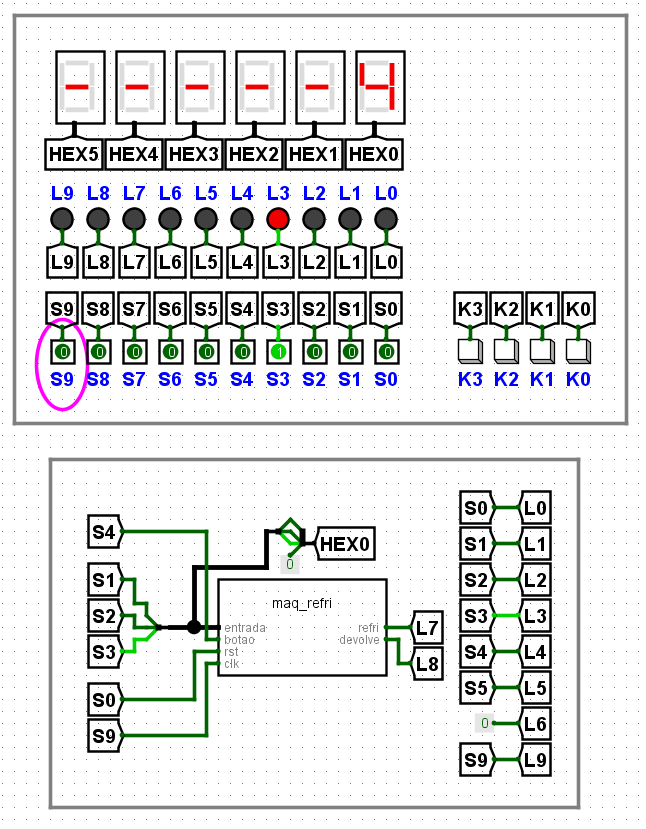
|  |  |  |
| --- | --- | --- |
| MOEDA | CÓDIGO | EM DECIMAL |
| R$ 0,10 | 010 | 2 |
| R$ 0,25 | 011 | 3 |
| R$ 0,50 | 100 | 4 |
| R$ 1,00 | 101 | 5 |
| R$ 0,00 | QUALQUER OUTRA ENTRADA | 0, 1, 6 e 7 |

O circuito armazena o valor de entrada em um sinal inteiro cujo valor pode assumir de 0 a 63. Cada unidade desse sinal é equivalente a 5 centavos, o que foi estabelecido como meio de economizar bits, uma vez que a máquina não aceita moedas de 1 centavo. Desse modo, um real, o preço de uma latinha de refrigerante, é equivalente a 20 unidades. O resto da lógico funciona da maneira como especificado na proposta do projeto.

*Figura I. Circuito da máquina de refrigerante conectada à mascara DE0-CV.*

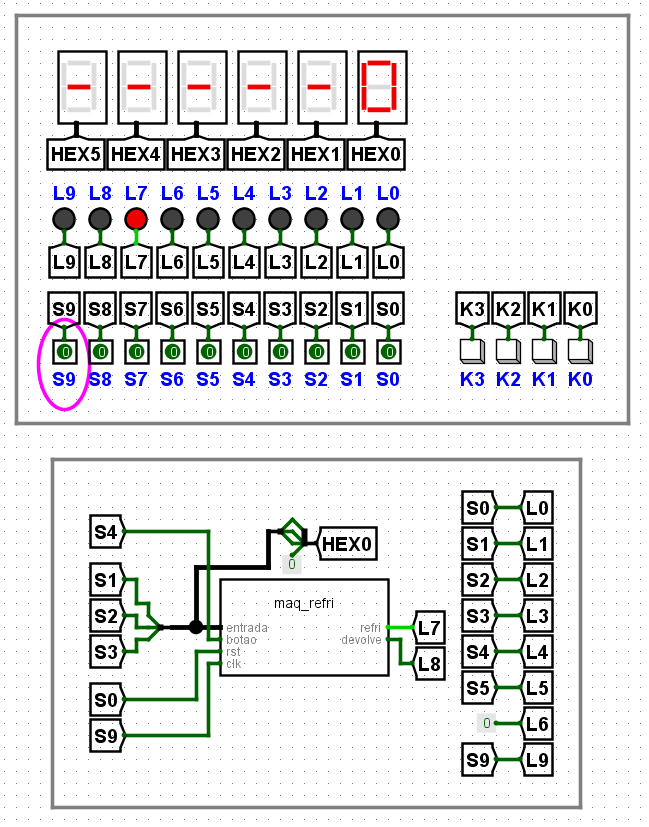
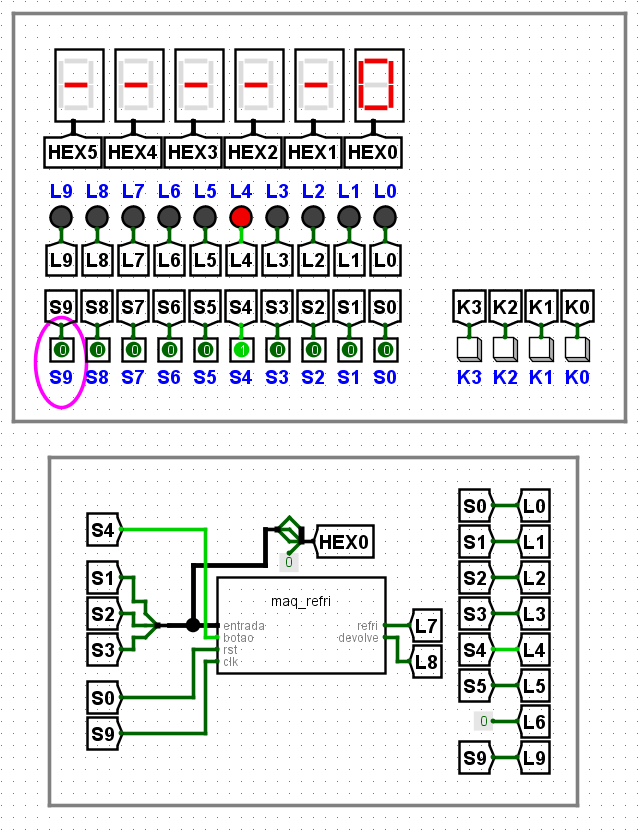
As imagens a seguir demonstram o funcionamento do circuito em ordem cronológica.

*Figura III: O circuito durante a borda de subida do 2º clock, quando uma moeda de 50 centavos foi inserida (código 100).*



*Figura II: O circuito após um clock sem nenhum outro input. Como de se esperar, o circuito interpretou como se uma moeda de 0 centavos fosse inserida e o valor interno não mudou.*

*Figura IV: Circuito após o clock. No 3º clock, o código da moeda não foi alterado, indicando que uma segunda moeda de 50 centavos foi inserida. Agora, o valor acumulado na máquina atingiu 1 real, mas como o botão não foi pressionado, nada acontece de imediato.*



*Figura V: Circuito após o 4º clock. O botão foi apertado e nenhuma moeda nova foi inserida. Porém, a entrega do refrigerante não pode ocorrer nesse ciclo de clock.*

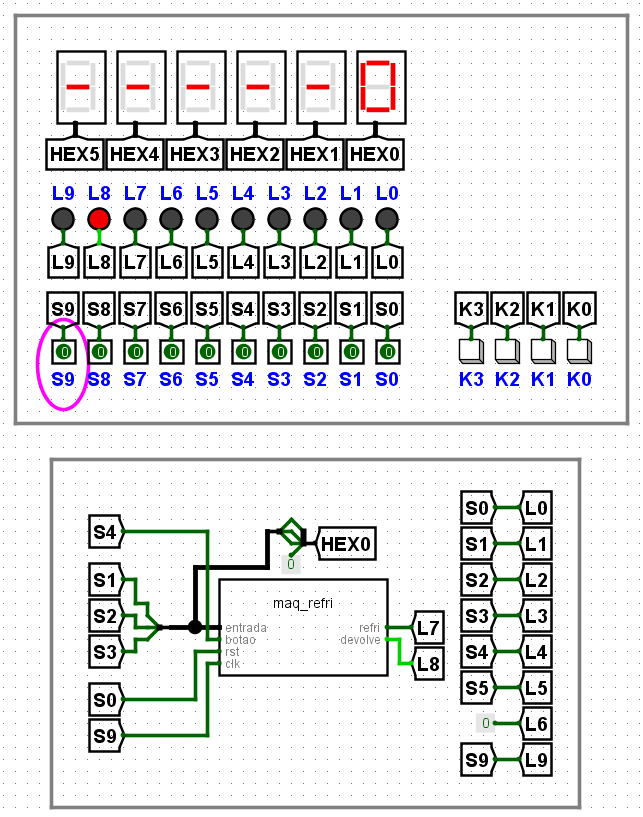
*Figura VI: Circuito após o 5º clock. A entrega do refrigerante foi efetuada.*

Diagram, schematic

Description automatically generated

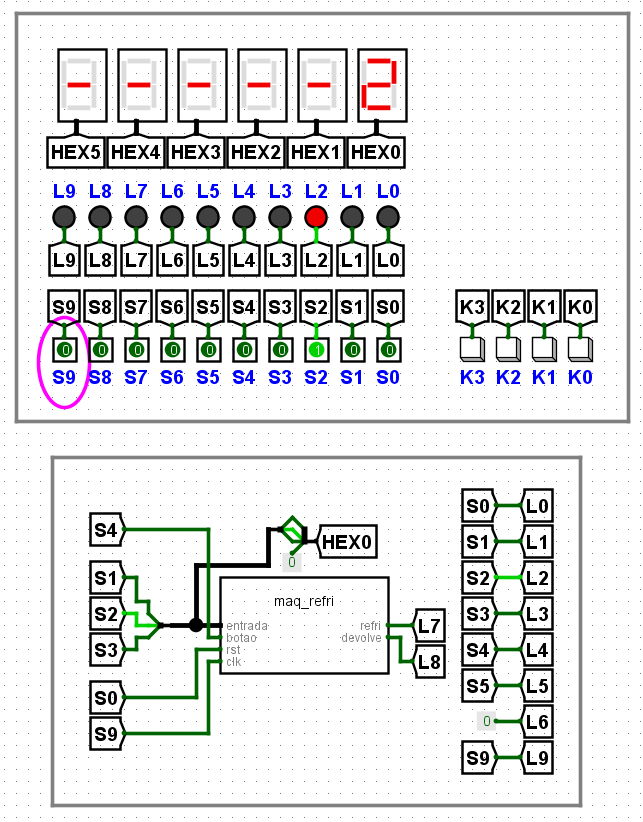
*Figura VII: Circuito após o 6º clock. Uma moeda de 25 centavos foi inserida.*

Diagram, schematic

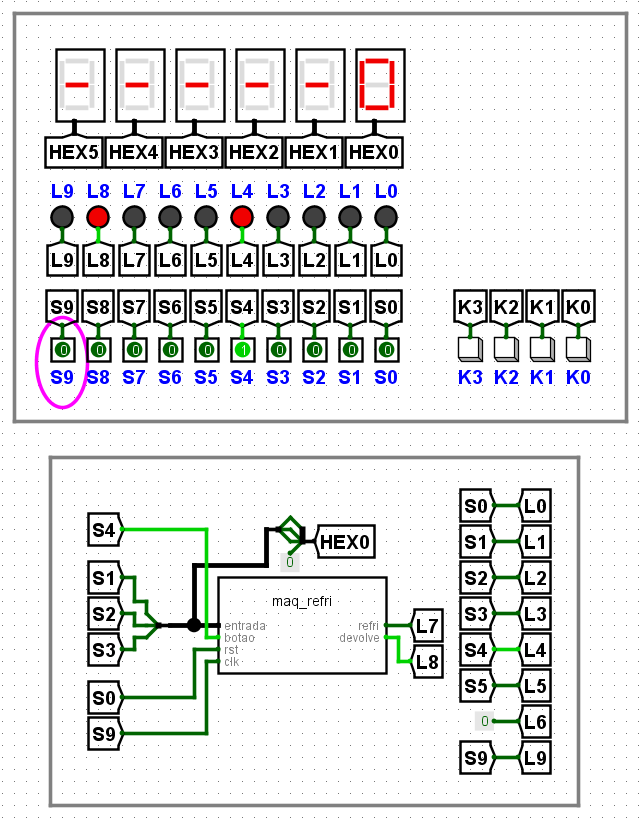
Description automatically generated

*Figura VIII: Circuito após o 7º clock. Uma moeda de um real foi inserida, ultrapassando o valor de um refrigerante, mas nenhuma ação será feira acerca disso nesse clock.*

*Figura IX: Circuito após o 8º clock. Todo o dinheiro inserido foi devolvido.*



*Figura X: Circuito após o 9º clock. Uma moeda de 10 centavos foi inserida.*



*Figura XI: Circuito após o 11º clock. O botão foi apertado e, após dois ciclos de clock, um para a mudança de estado e outro para o resultado, o dinheiro foi devolvido.*

O código VHDL utilizado para gerar o circuito acima está disponível logo abaixo:

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**numeric\_std**.all;**

**entity** maq\_refri **is**

-- entrada: 0: nada, 2: 10 cent, 3: 25 cent, 4 : 50 cent, 5: 1 real

**port** **(**

entrada **:** **in** std\_logic\_vector**(**2 **downto** 0**)** **:=** "000"**;**

botao**,** rst**,** clk **:** **in** std\_logic**;**

refri **:** **out** std\_logic **:=** '0'**;**

devolve **:** **out** std\_logic **:=** '0'

**);**

**end** maq\_refri**;**

**architecture** arch\_refri **of** maq\_refri **is**

--Estados possiveis da maquina

**type** state\_t **is** **(**refri\_idle**,** refri\_give**,** refri\_reset**);**

**signal** current\_state **:** state\_t **:=** refri\_idle**;**

**signal** next\_state **:** state\_t **:=** refri\_idle**;**

**signal** grana\_total **:** integer **range** 0 **to** 63 **:=** 0**;**

**begin**

--Processo que imprime informacoes relevantes da maquina no console

impr**:process(**current\_state**,** next\_state**,** grana\_total**,** botao**)** **is**

**begin**

**if** **(**current\_state'**EVENT)** **then**

**report** "----------> CHANGE IN current\_state: " **&** state\_t'**image(**current\_state**);**

**elsif** **(**next\_state'**EVENT)** **then**

**report** "----------> CHANGE IN next\_state: " **&** state\_t'**image(**next\_state**);**

**elsif** **(**grana\_total'**EVENT)** **then**

**report** "----------> CHANGE IN GRANA: " **&** integer'**image(**grana\_total**);**

**elsif** **(**botao'**EVENT)** **then**

**report** "----------> CHANGE IN BOTAO: " **&** std\_logic'**image(**botao**);**

**end** **if;**

**end** **process;**

--Processo que cuida da logica combinacional da maquina de refrigerante

comb**:process(**clk**)** **is**

**begin**

**if** **falling\_edge(**clk**)** **then**

next\_state **<=** current\_state**;**

**case** current\_state **is**

**when** refri\_idle **=>**

devolve **<=** '0'**;**

refri **<=** '0'**;**

**if** botao **=** '1' **then**

**if** grana\_total **=** 20 **then**

next\_state **<=** refri\_give**;**

**else**

next\_state **<=** refri\_reset**;**

**end** **if;**

**else**

**if** grana\_total **>** 20 **then**

next\_state **<=** refri\_reset**;**

**else**

next\_state **<=** refri\_idle**;**

**end** **if;**

**end** **if;**

**when** refri\_give **=>**

devolve **<=** '0'**;**

refri **<=** '1'**;**

next\_state **<=** refri\_idle**;**

**when** refri\_reset **=>**

devolve **<=** '1'**;**

next\_state **<=** refri\_idle**;**

**end** **case;**

**end** **if;**

**end** **process** comb**;**

--Processo que cuida da logica da memoria da maquina de refrigerante

mem**:process(**rst**,** clk**)** **is**

**begin**

**if** **rising\_edge(**clk**)** **then**

**if** rst **=** '1' **then**

current\_state **<=** refri\_idle**;**

**else**

current\_state **<=** next\_state**;**

**end** **if;**

**if** next\_state **=** refri\_reset **or** next\_state **=** refri\_give **or** rst **=** '1' **then**

grana\_total **<=** 0**;**

**else**

**case** entrada **is**

**when** "010" **=>**

grana\_total **<=** grana\_total **+** 2**;**

**when** "011" **=>**

grana\_total **<=** grana\_total **+** 5**;**

**when** "100" **=>**

grana\_total **<=** grana\_total **+** 10**;**

**when** "101" **=>**

grana\_total **<=** grana\_total **+** 20**;**

**when** **others** **=>**

**null;**

**end** **case;**

**end** **if;**

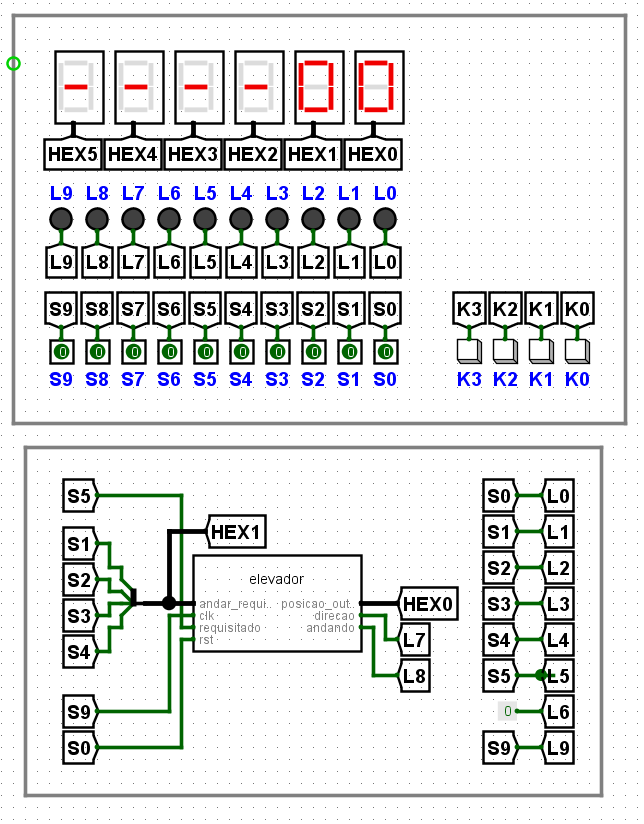
**end** **if;**

**end** **process** mem**;**

**end** arch\_refri**;**

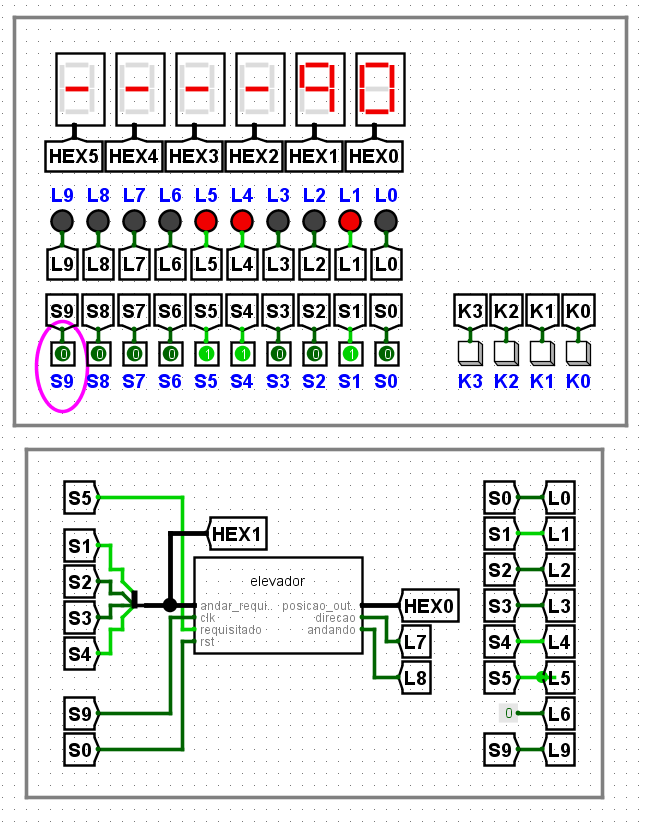
# ELEVADOR

O elevador possui dois sinais internos do tipo vetor de 4 bits. Eles guardam em que andar o elevador está e para qual andar ele está indo. Quando a variável de entrada “requisitado” é ativada, recebe ‘1’, o andar destino é atualizado com o andar disponível na entrada. Caso requisitado permaneça em 0, valor do andar da entrada é ignorado. O elevador ou sobe, ou desce ou fica parado. A saída “direcao” indica para qual direção o elevador estava ou está indo e a saída “andando” indica se o elevador está em movimento ou não.



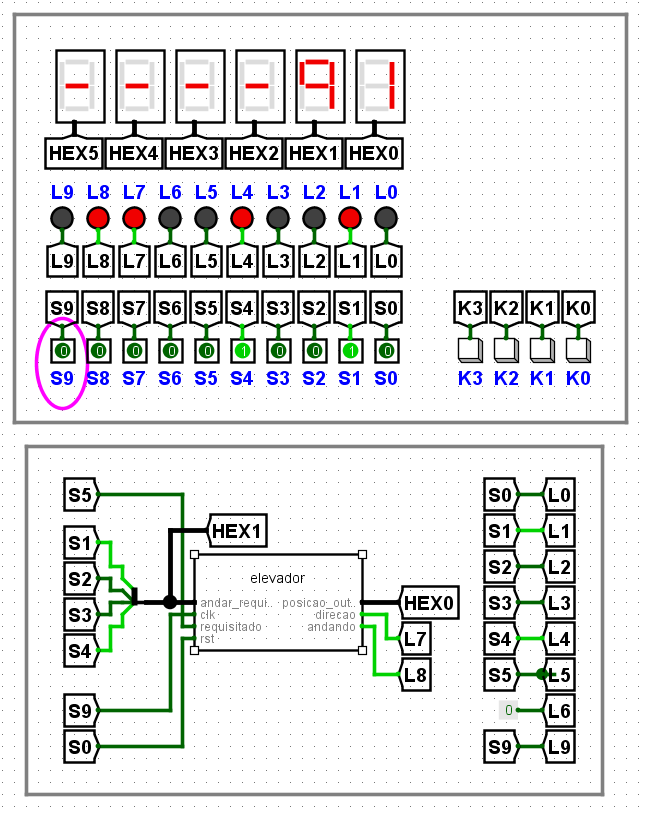
*Figura XII: Circuito do elevador conectado à máscara DE0-CV.*

Diagram, schematic

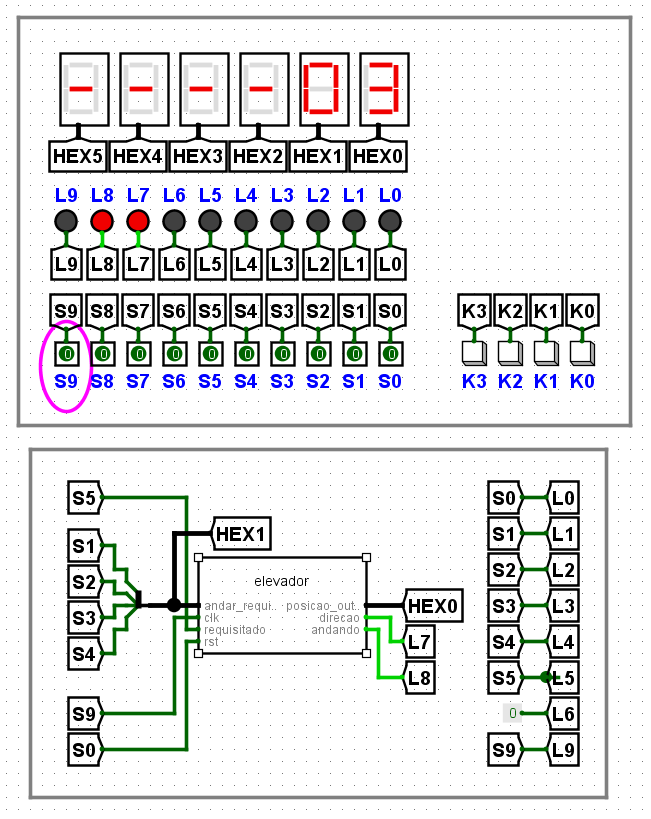
Description automatically generated

*Figura XIV: Circuito após o 4º clock. O andar número 9 foi requisitado.*

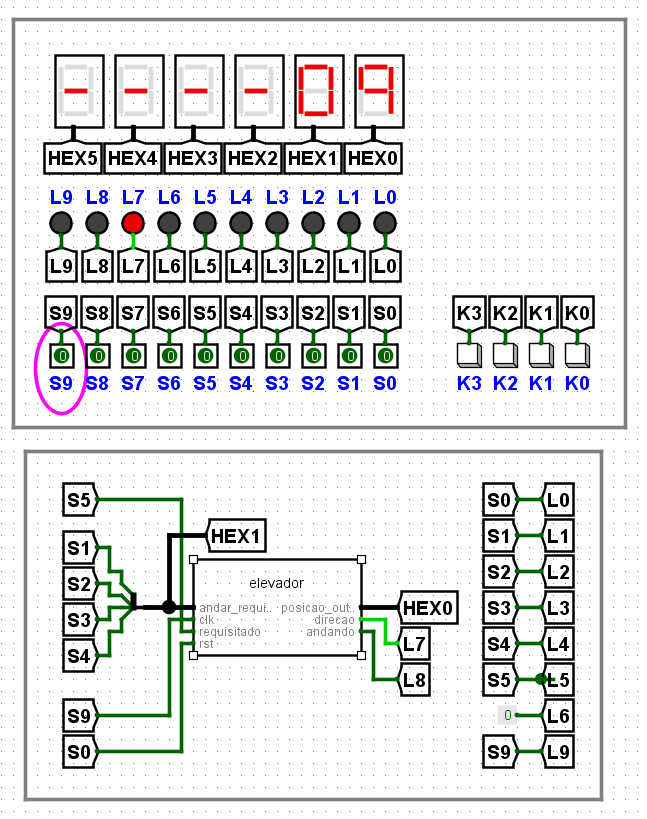
*Figura XIII: Circuito após o 3º clock. Como o esperado, o elevador permaneceu inerte durante os primeiros clocks, quando nenhum input foi fornecido.*



*Figura XV: Circuito após o 6º clock. O estado foi alterado no 5º ciclo de clock e no 6º o elevador começou a se mover.*



*Figura XVI: Circuito após o 8º clock. Notar-se-á que o movimento do elevador se mantém, não importando, enquanto o requisitado está desligado, o valor assumido pela entrada.*



*Figura XVII: Circuito após o 15º clock. O elevador chegou ao andar desejado e encerrou seu movimento.*

O código VHDL utilizado para gerar o circuito acima está disponível logo abaixo:

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**use** ieee**.**numeric\_std**.all;**

**entity** elevador **is**

**port** **(**

andar\_requisitado**:** **in** std\_logic\_vector**(**3 **downto** 0**);**

--requisitado indica se tem alguem chamando o elevador

clk**,** requisitado**,** rst**:** **in** std\_logic**;**

--posicao atual do elevador

posicao\_output**:** **out** std\_logic\_vector**(**3 **downto** 0**)** **:=** "0000"**;**

--direcao: '0' indica descendo, '1' indica subindo

direcao**:** **out** std\_logic **:=** '0'**;**

--andando: '0' indica parado, '1' indica que esta andando

andando**:** **out** std\_logic **:=** '0'

**);**

**end** elevador**;**

**architecture** arch\_elevador **of** elevador **is**

**type** state\_t **is** **(**up**,** down**,** stop**);**

--signal andar : integer range 0 to 15 := 0;

**signal** current\_state **:** state\_t **:=** stop**;**

**signal** next\_state **:** state\_t **:=** stop**;**

**signal** destino **:** std\_logic\_vector**(**3 **downto** 0**)** **:=** "0000"**;**

**signal** posicao **:** std\_logic\_vector**(**3 **downto** 0**)** **:=** "0000"**;**

**constant** subindo **:** std\_logic **:=** '1'**;**

**constant** descendo **:** std\_logic **:=** '0'**;**

**begin**

impr**:process(**current\_state**,** next\_state**,** destino**,** posicao**)** **is**

**begin**

**if** **(**current\_state'**EVENT)** **then**

**report** "----------> CHANGE IN current\_state: " **&** state\_t'**image(**current\_state**);**

**elsif** **(**next\_state'**EVENT)** **then**

**report** "----------> CHANGE IN next\_state: " **&** state\_t'**image(**next\_state**);**

--elsif (destino'EVENT) then

--report "----------> CHANGE IN DESTINO: " & unsigned'image(unsigned(destino));

--elsif (posicao'EVENT) then

--report "----------> CHANGE IN POSICAO: " & unsigned'image(unsigned(posicao));

**end** **if;**

**end** **process** impr**;**

comb**:process(**clk**)** **is**

**begin**

**if** **rising\_edge(**clk**)** **then**

next\_state **<=** current\_state**;**

posicao\_output **<=** posicao**;**

**case** current\_state **is**

**when** stop **=>**

andando **<=** '0'**;**

**if** posicao **/=** destino **then**

**if** posicao **<** destino **then**

next\_state **<=** up**;**

**else**

next\_state **<=** down**;**

**end** **if;**

**end** **if;**

**when** up **=>**

**if** posicao **=** destino **then**

next\_state **<=** stop**;**

**end** **if;**

andando **<=** '1'**;**

direcao **<=** subindo**;**

**when** down **=>**

**if** posicao **=** destino **then**

next\_state **<=** stop**;**

**end** **if;**

andando **<=** '1'**;**

direcao **<=** descendo**;**

**end** **case;**

**end** **if;**

**end** **process** comb**;**

mem**:process(**clk**)** **is**

**begin**

**if** rst **=** '1' **then**

current\_state **<=** stop**;**

-- andando <= '0';

posicao **<=** "0000"**;**

destino **<=** "0000"**;**

**elsif** **falling\_edge(**clk**)** **then**

**if** requisitado **=** '1' **then**

destino **<=** andar\_requisitado**;**

**end** **if;**

current\_state **<=** next\_state**;**

**if** next\_state **=** up **then**

posicao **<=** std\_logic\_vector**(**unsigned**(**posicao**)** **+** 1**);**

**elsif** next\_state **=** down **then**

posicao **<=** std\_logic\_vector**(**unsigned**(**posicao**)** **-** 1**);**

**end** **if;**

**end** **if;**

**end** **process** mem**;**

**end** arch\_elevador**;**