Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática - FACIN

LABORG

Prof. Dr. Rafael Garibotti

AULA SOBRE:

PROJETO DE UM CIRCUITO DIGITAL DE MÉDIA COMPLEXIDADE

CRONÔMETRO

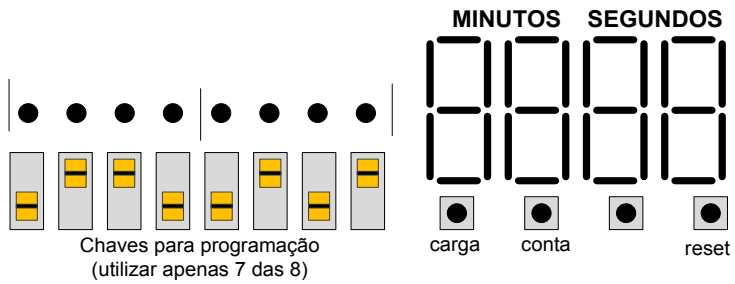
INTRODUÇÃO

- O objetivo deste trabalho é especificar um módulo que deverá ser implementado em hardware pelos alunos, desde a escrita em VHDL do circuito, até a sua prototipação.

- Pressupostos: o cronômetro deve operar sobre a plataforma Nexys2 disponível em laboratório, com todas as restrições que esta plataforma impõe, o que inclui:
  - ✓ Só existe uma fonte de relógio na plataforma, um cristal de 50MHz.
  - ✓ Somente podem ser usados os dispositivos de entrada e saída nativos da placa, quais sejam: para saída de dados, existem 4 mostradores de sete segmentos e 8 diodos emissores de luz (LEDs); para entrada de dados existem apenas 8 chaves deslizantes e quatro botões (push button).

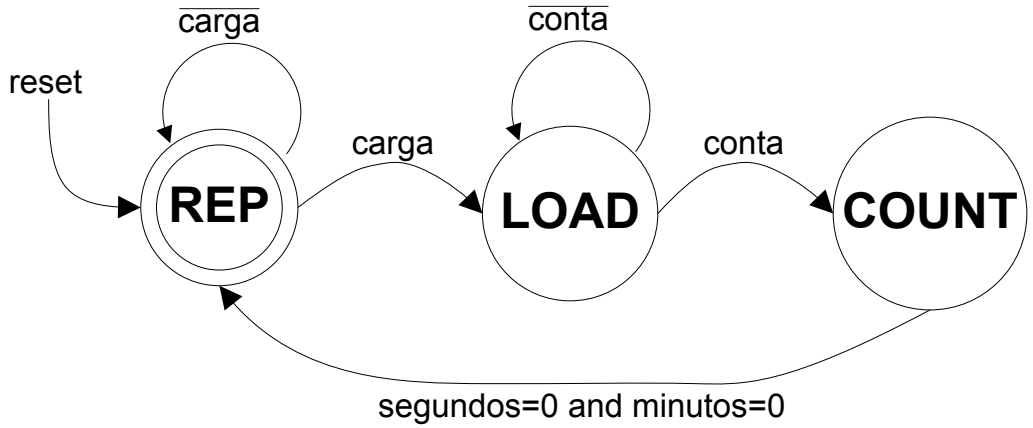
ESPECIFICAÇÃO PRELIMINAR

- Realizar um **cronômetro** com precisão de segundo.



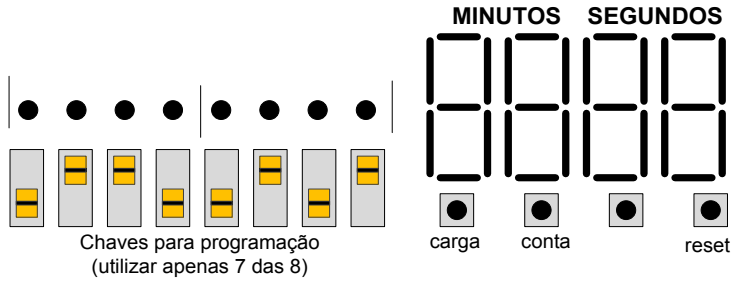
Operação:

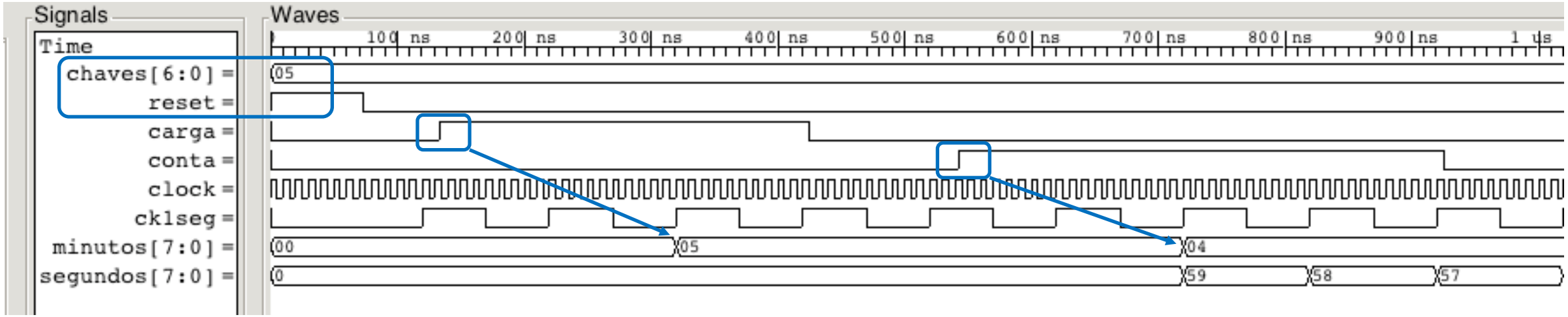
- **Estado REP**
  - ✓ Quando o botão **reset** é pressionado o display deve mostrar 00:00.
  - ✓ Quando o botão **carga** é pressionado muda-se para o estado LOAD.
- **Estado LOAD**
  - ✓ As chaves definem o valor dos minutos.
  - ✓ Quando o botão **conta** é pressionado, muda-se para o estado COUNT.
- **Estado COUNT**
  - ✓ Conta de forma decrescente até chegar a 00:00, voltando para o estado REP ao chegar neste valor.



EXEMPLO DE OPERAÇÃO

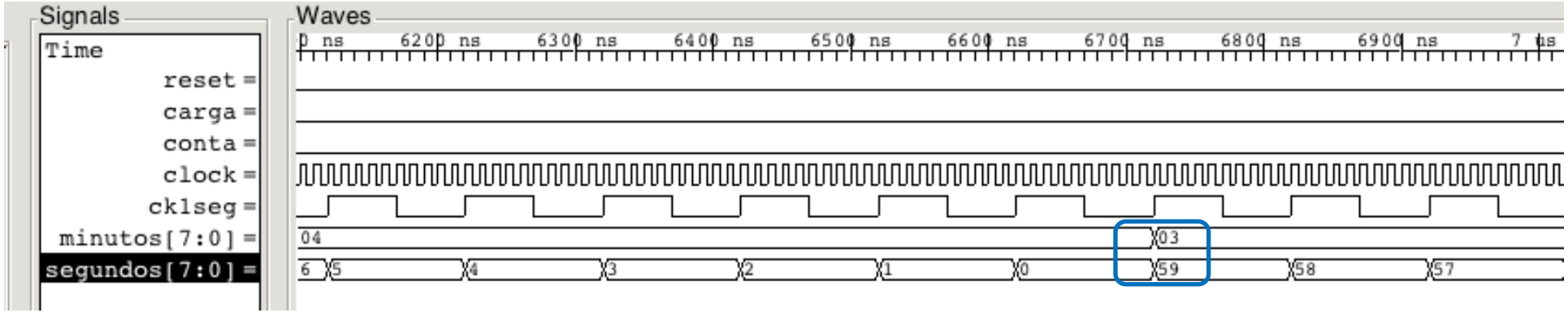
1. Pressionar **reset** e colocar nas chaves o valor 5.
2. Pressionar **carga**. Então os minutos recebem o valor das chaves, 5.
3. Pressionar **conta** para iniciar a contagem.
   - ✓ Na sequência teremos: 4:59 – 4:58 – 4:57...





EXEMPLO DE OPERAÇÃO

4. Troca de minuto → 4:00 para 3:59.



5. Final da Simulação
   - ✓ Ao chegar em 00:00, deve-se parar a contagem e voltar para o estado REP.

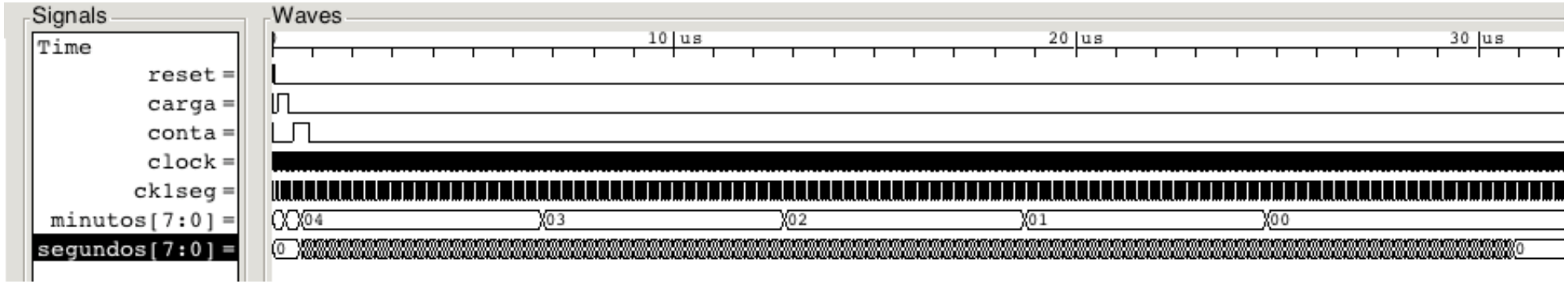
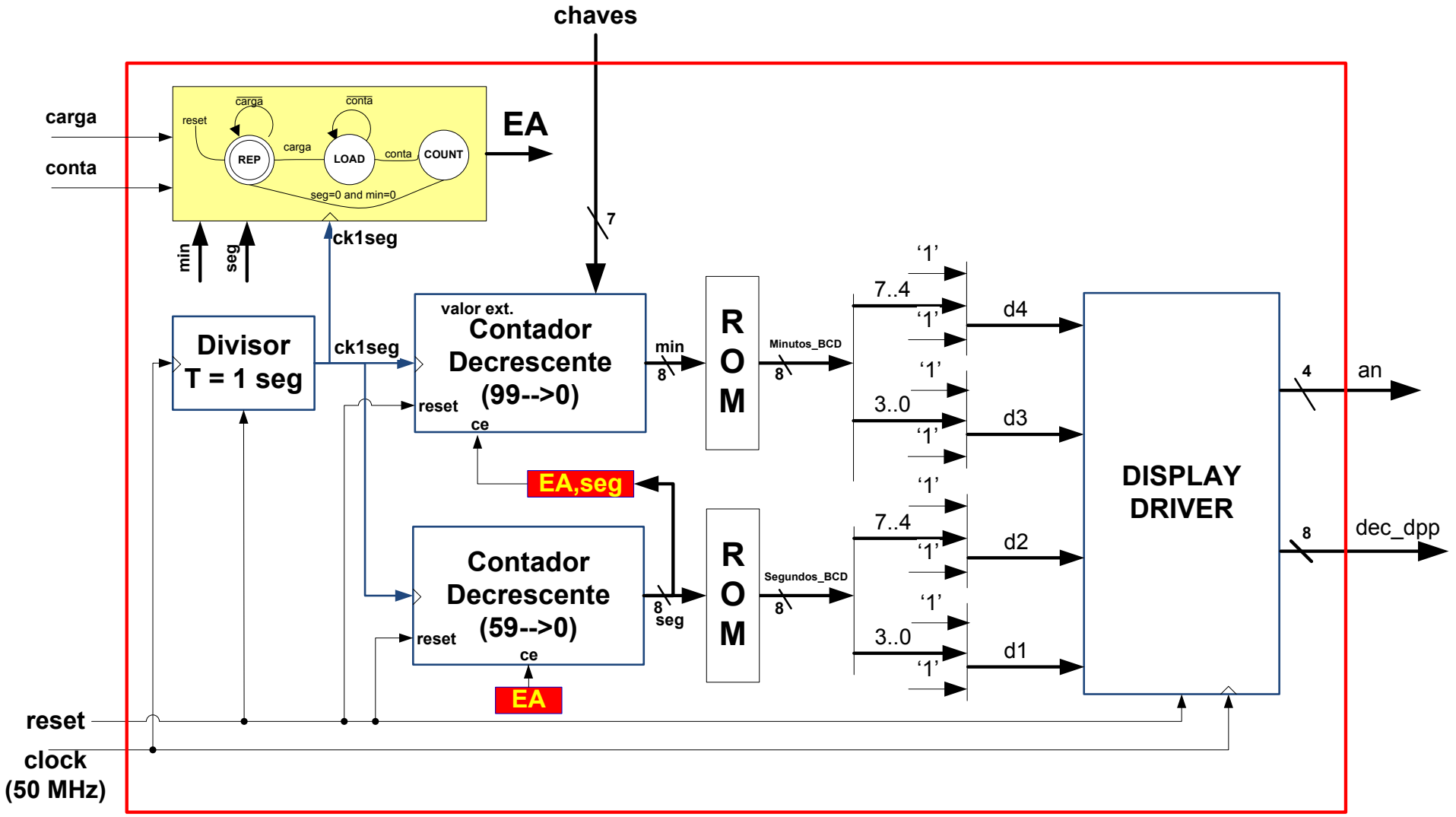


DIAGRAMA DE BLOCOS DO CIRCUITO



CONVERSÃO

- Supondo segundos = 0x0F
  - ✓ `segundos_BCD <= conv_to_BCD(CONV_INTEGER(segundos));`
- O valor retornado por conv_to_BCD e atribuído ao sinal segundos_BCD será 00010101 (15).

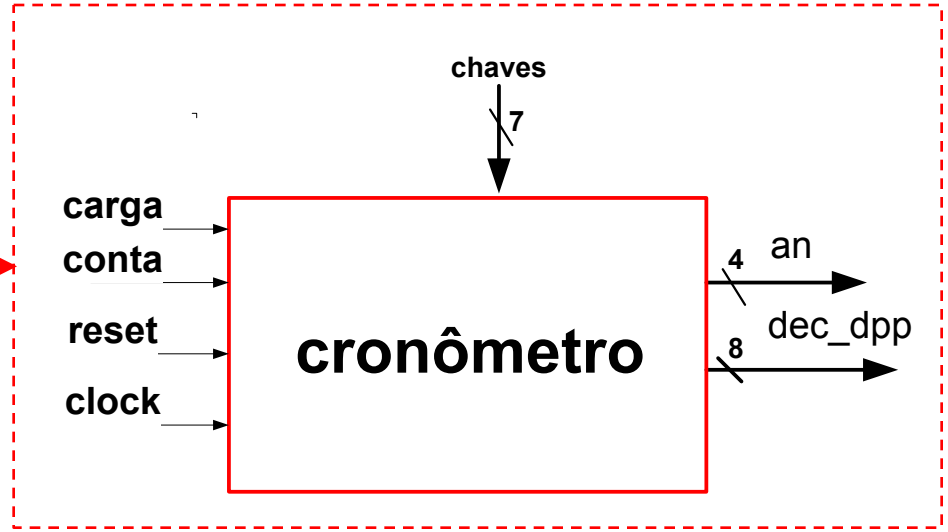
```
type ROM is array (0 to 99) of std_logic_vector (7 downto 0);
constant conv_to_BCD : ROM:=(
                "00000000", "00000001", "00000010", "00000011", "00000100",
                "00000101", "00000110", "00000111", "00001000", "00001001",
                "00010000", "00010001", "00010010", "00010011", "00010100",
                "00010101","00010110", "00010111", "00011000", "00011001",
                "00100000", "00100001", "00100010", "00100011", "00100100",
                "00100101", "00100110", "00100111", "00101000", "00101001",
                "00110000", "00110001", "00110010", "00110011", "00110100",
                "00110101", "00110110", "00110111", "00111000", "00111001",
                "01000000", "01000001", "01000010", "01000011", "01000100",
                "01000101", "01000110", "01000111", "01001000", "01001001",
                "01010000", "01010001", "01010010", "01010011", "01010100",
                "01010101", "01010110", "01010111", "01011000", "01011001",
                "01100000", "01100001", "01100010", "01100011", "01100100",
                "01100101", "01100110", "01100111", "01101000", "01101001",
                "01110000", "01110001", "01110010", "01110011", "01110100",
                "01110101", "01110110", "01110111", "01111000", "01111001",
                "10000000", "10000001", "10000010", "10000011", "10000100",
                "10000101", "10000110", "10000111", "10001000", "10001001",
                "10010000", "10010001", "10010010", "10010011", "10010100",
                "10010101", "10010110", "10010111", "10011000", "10011001");
```

SUGESTÃO PARA ESTRUTURA DE CÓDIGO

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity dec_cron is
        generic ( CLOCK_FREQ : integer := 25000000 );
        port (
            PINOS CONFORME O DIAGRAMA DE BLOCOS
        );
end dec_cron;

architecture dec_cron of dec_cron is
   DECLARAR OS SINAIS NESSÁRIOS
   type ROM is array (0 to 99) of std_logic_vector (7 downto 0);
   constant conv_to_BCD : ROM:=(
                "00000000", "00000001", "00000010", "00000011", "00000100", ...
               );
    type states is (REP, LOAD, COUNT);
    signal EA, PE : states;
```



SUGESTÃO PARA ESTRUTURA DE CÓDIGO

```
begin
        P1:  divisor de clock para gerar o ck1seg
        P2/P3: máquina de estados para determinar o estado atual (EA)
        P4: contador de segundos
        P5: contador de minutos
        -- instanciação das ROMs
        segundos_bcd <= conv_to_BCD(conv_integer(segundos));
        minutos_bcd  <= ...
        -- display driver
        d1 <= '1' & segundos_bcd(3 downto 0) & '1';
        d2 <= ...
        d3 <= ...
        d4 <= ...
        display_driver : entity work.dspl_drv
        port map (
          ...
        );
end dec_cron;
```

TESTBENCH PARA VALIDAR O CRONÔMETRO

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity tb_dec_counter  is
end tb_dec_counter;

architecture  tb_dec_counter  of tb_dec_counter  is

  signal clock, reset, carga, conta : std_logic := '1';
  signal chaves: std_logic_vector(6  downto  0);

begin

  clock <= not clock after 5 ns;
  reset <= '1', '0' after 73 ns;
  carga <= '0', '1' after 133 ns,  '0' after 425 ns;
  conta <= '0', '1' after 543 ns,  '0' after 925 ns;
  chaves <= "0000101";

  uut : entity work.dec_cron
        generic map ( CLOCK_FREQ => 4 )    --  para simulação utilizar um divisor menor
        port map (
          clock     => clock,
          reset     => reset,
          carga     => carga,
          conta     => conta,
          chaves    => chaves,
          an        => open,
          dec_ddp   => open);
end tb_dec_counter;
```

UCF: PROTOTIPAR O PROJETO

- Atenção para os nomes declarados dos sinais!!!

```
# pinos para as entradas e saídas
NET "clock"      LOC = "b8";
NET "carga"      LOC = "h13";
NET "conta"      LOC = "e18";
NET "reset"      LOC = "b18";
NET "chaves<0>"  LOC = "g18";
NET "chaves<1>"  LOC = "h18";
NET "chaves<2>"  LOC = "k18";
NET "chaves<3>"  LOC = "k17";
NET "chaves<4>"  LOC = "l14";
NET "chaves<5>"  LOC = "l13";
NET "chaves<6>"  LOC = "n17";
NET "an<0>"      LOC = "f17";
NET "an<1>"      LOC = "h17";
NET "an<2>"      LOC = "c18";
NET "an<3>"      LOC = "f15";
. . .
```

```
. . .
NET "dec_ddp<0>"      LOC = "c17";
NET "dec_ddp<1>"      LOC = "h14";
NET "dec_ddp<2>"      LOC = "j17";
NET "dec_ddp<3>"      LOC = "g14";
NET "dec_ddp<4>"      LOC = "d16";
NET "dec_ddp<5>"      LOC = "d17";
NET "dec_ddp<6>"      LOC = "f18";
NET "dec_ddp<7>"      LOC = "l18";
```

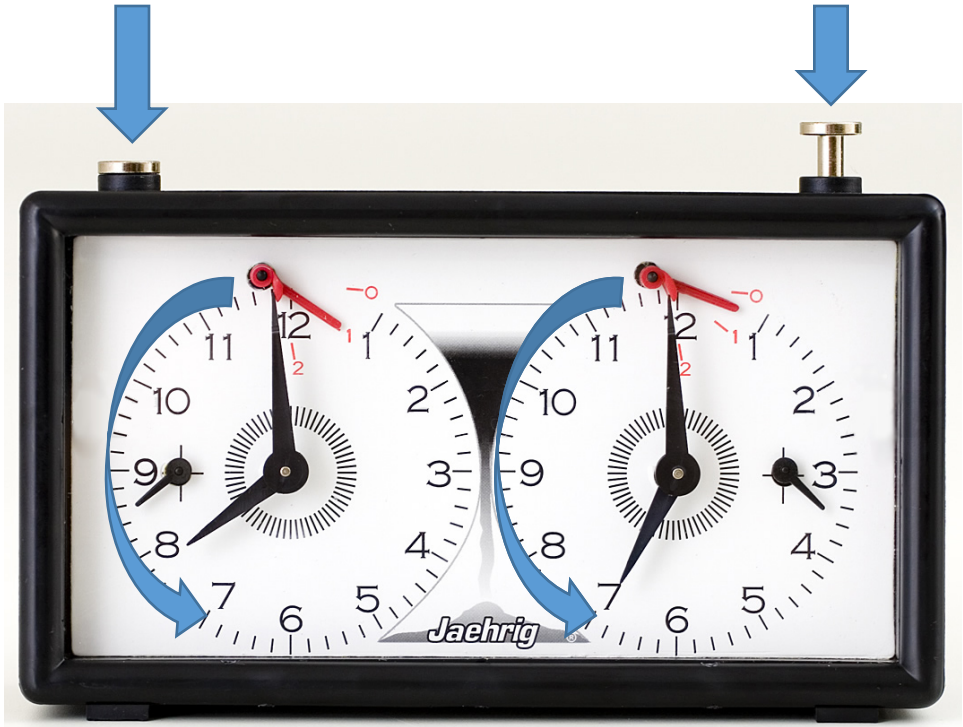
RELÓGIO DE XADREZ

RELÓGIO DE XADREZ

1. Em campeonatos, o xadrez é jogado usando um relógio. A razão para isso é limitar o tempo máximo de jogo e evitar que ele dure para sempre.
2. Um relógio de xadrez é composto por dois cronômetros e dois botões, que controlam a contagem dos cronômetros.

RELÓGIO DE XADREZ

3. O funcionamento básico desse tipo de relógio é definido como:
   - ✓ Um tempo máximo de jogo para cada jogador é definido, e os dois cronômetros são ajustados para esse valor;
   - ✓ Um jogador inicia sua jogada e seu cronômetro começa a regredir;
   - ✓ Assim que o jogador terminar sua jogada, ele aperta o botão, seu cronômetro para de regredir e é a vez do próximo jogador realizar sua jogada;
   - ✓ Esse processo segue até que um dos cronômetros chegue ao ponto 00:00.



RELÓGIO DE XADREZ

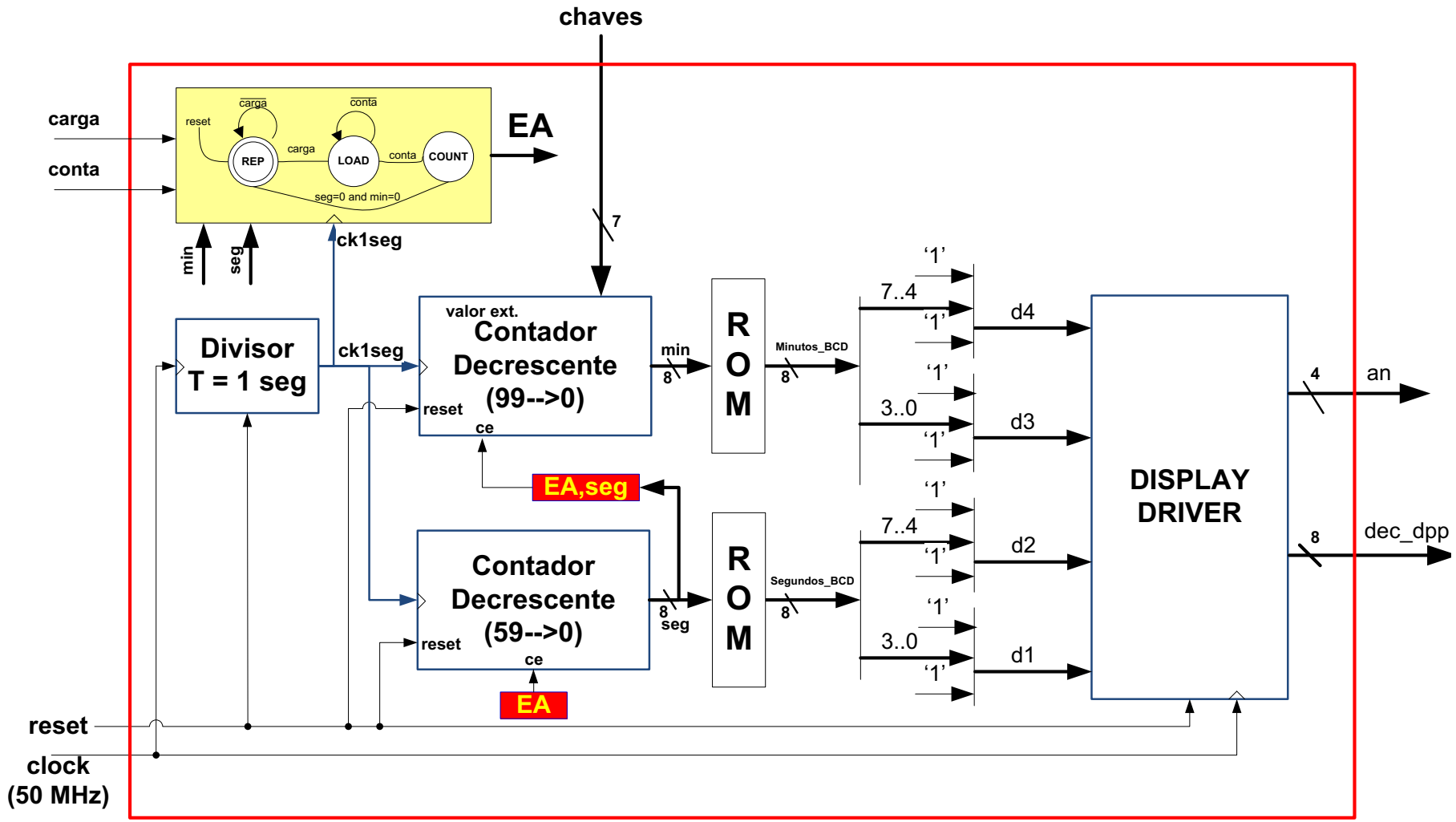
4. O objetivo desse trabalho é o projeto de um relógio de xadrez, utilizando conceitos de projeto de circuitos digitais e a placa de prototipação.
5. Para tanto, utilize:
   - ✓ Os leds da placa para identificar o turno de cada jogador;
   - ✓ O display de 7 segmentos para mostrar o valor dos cronômetros e identificar o jogador vencedor;
   - ✓ Os botões para controlar o relógio;
   - ✓ As chaves para programar os cronômetros.

RELÓGIO DE XADREZ

6. Para o controle do relógio, descreva uma máquina de estados utilizando dois processos para o bloco de controle e lógica combinacional para ler e escrever nos pinos de E/S.
7. Reuse o código da parte 1 desse laboratório para os cronômetros.

PONTO DE PARTIDA

- Cronômetro validado por simulação e prototipação.



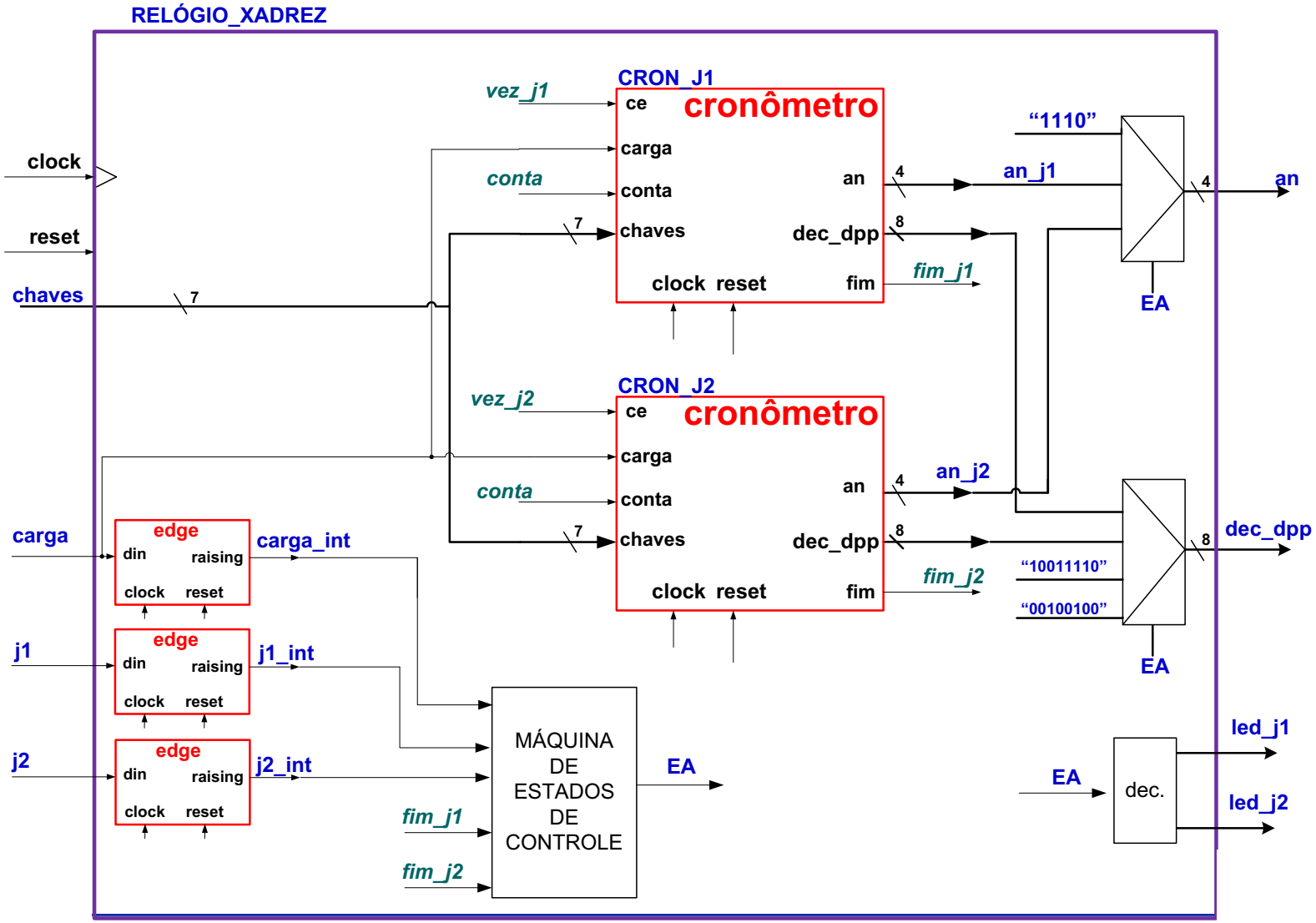
PRIMEIRA ATIVIDADE

- Acrescentar duas portas ao cronômetro: **ce** e **fim**.
  - ✓ **ce**: habilita ou não o divisor de clock para obtenção de 1 seg.
    - O teste do ce é inserido no divisor de clock, depois do teste da borda de clock.
  - ✓ **fim**: indica que o cronômetro chegou a zero.
    - Na prática ocorre quando voltamos para o estado de REP.

```
entity dec_cron is
  generic (CLOCK_FREQ : integer := 25000000 );
  port ( ce       : in  std_logic;
         fim      : out std_logic;
         clock    : in  std_logic;
         reset    : in  std_logic;
         carga    : in  std_logic;
         chaves   : in  std_logic_vector(6 downto 0);
         an       : out std_logic_vector(3 downto 0);
         dec_ddp  : out std_logic_vector(7 downto 0));
end dec_cron;
```

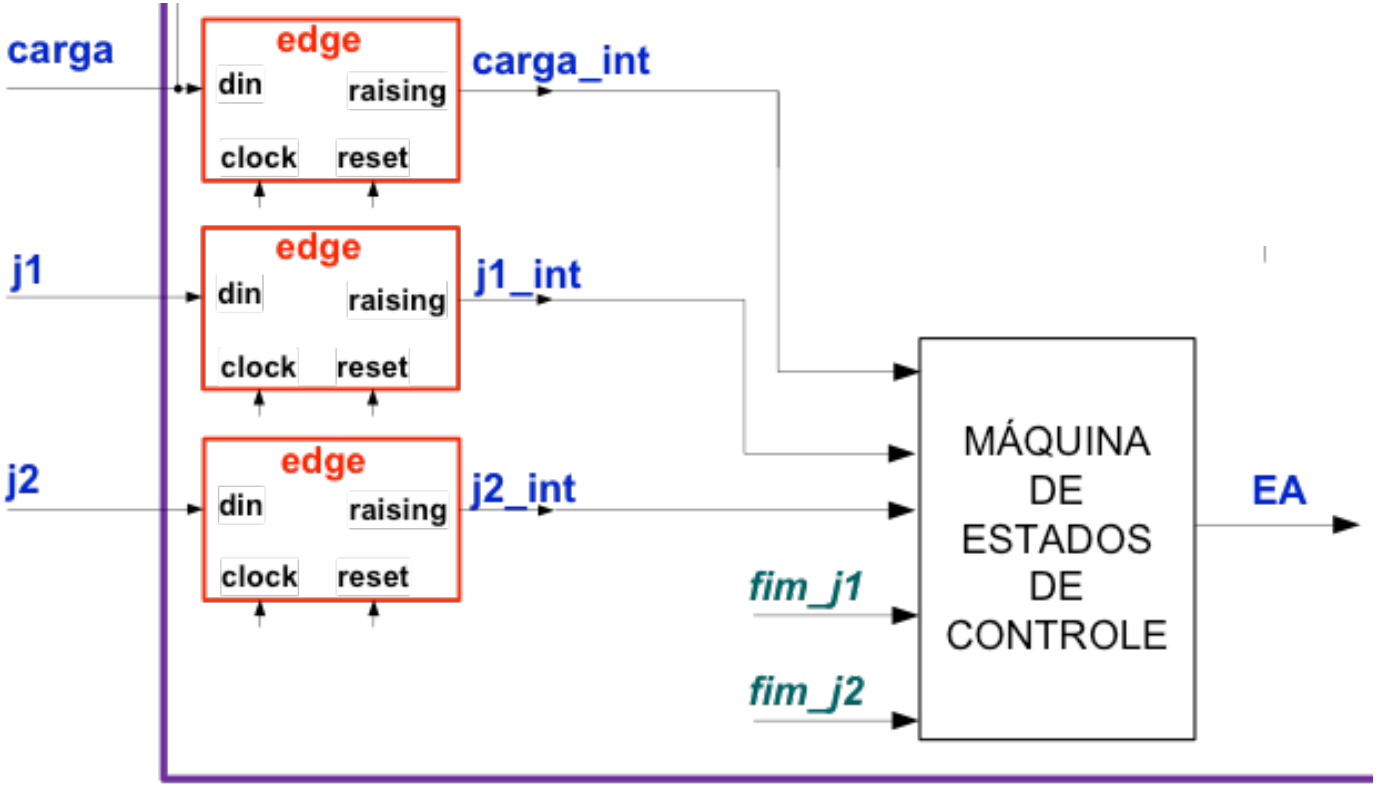
RECOMENDA-SE SIMULAR ESTA MODIFICAÇÃO ALTERANDO O TESTBENCH FORNECIDO NA PRIMEIRA PARTE DO TRABALHO ANTES DE UTILIZÁ-LA NO RELÓGIO DE XADREZ!

DIAGRAMA DE BLOCOS

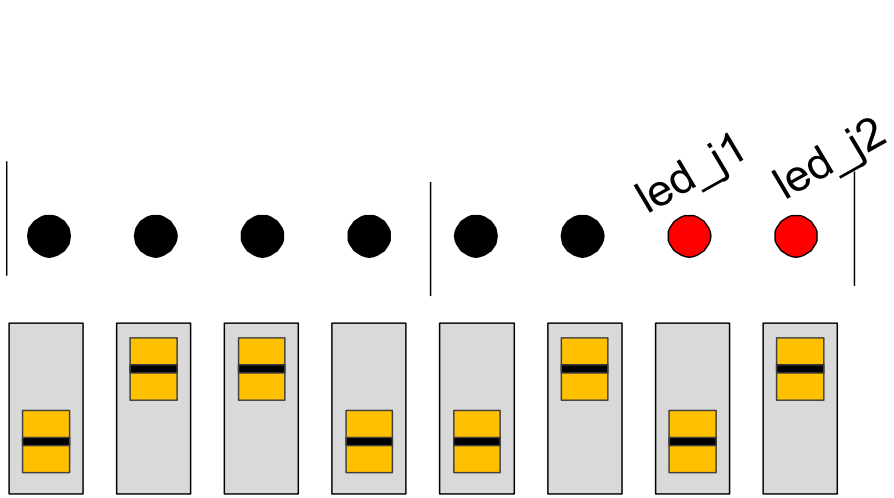


FUNÇÃO DO CIRCUITO *EDGE*

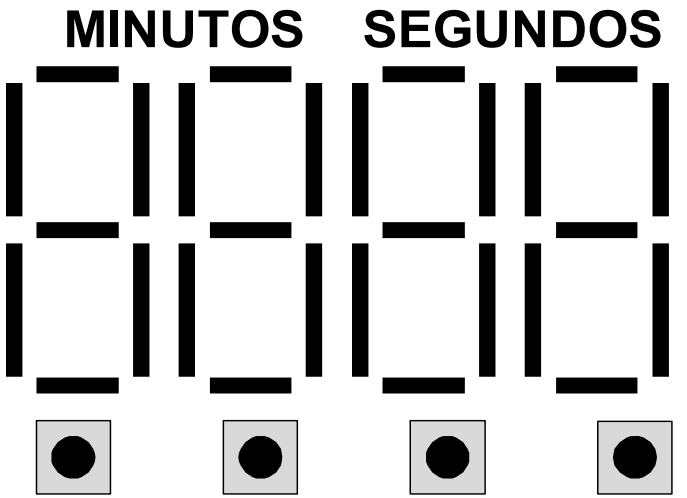
- Dado que a máquina de estados opera na frequência de 50 MHz, o valor oriundo da tecla deve ser filtrado para não ficar ativo durante muito tempo (isto é, por mais de um ciclo de clock).



ENTRADAS E SAÍDAS

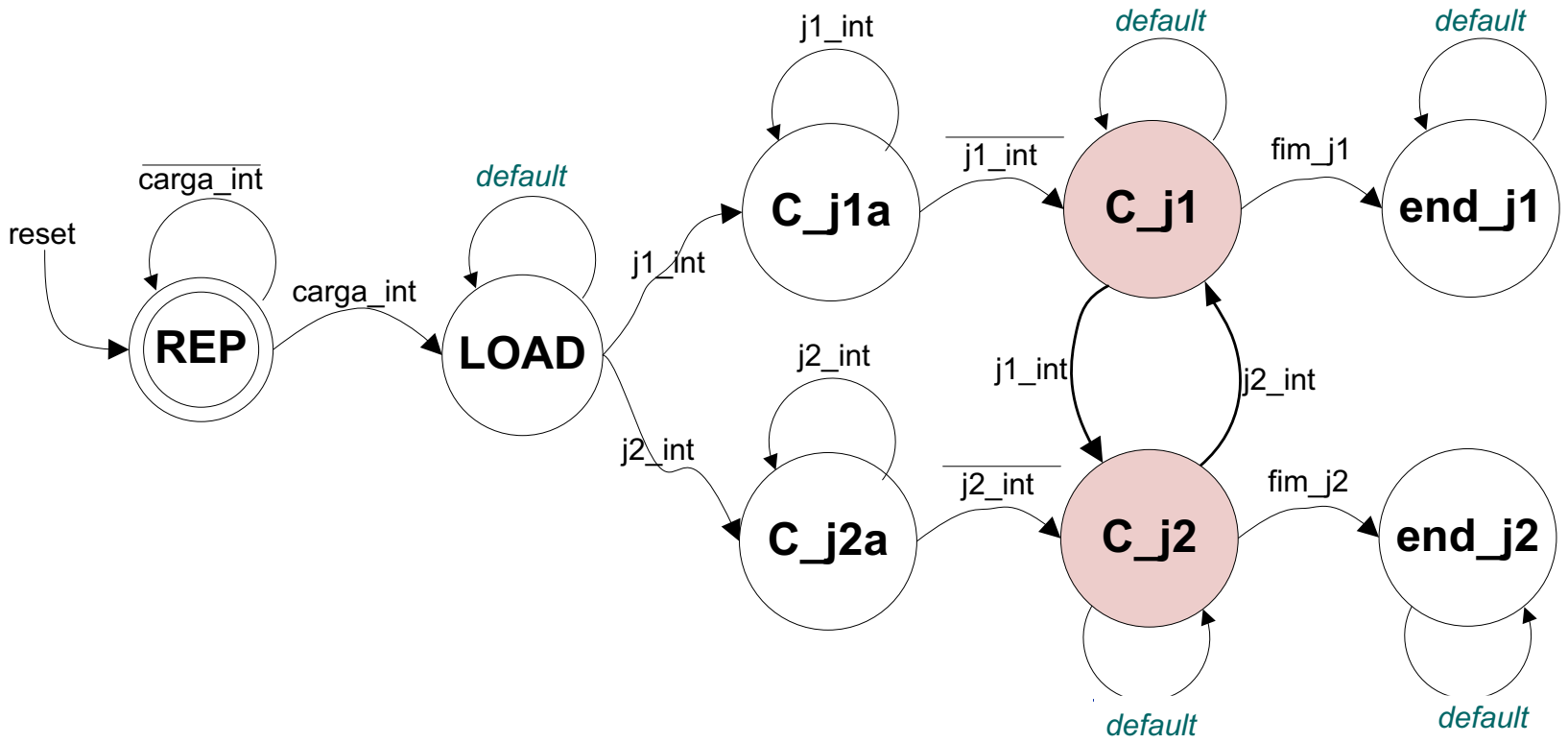


Chaves para programação
(utilizar apenas 7 das 8)



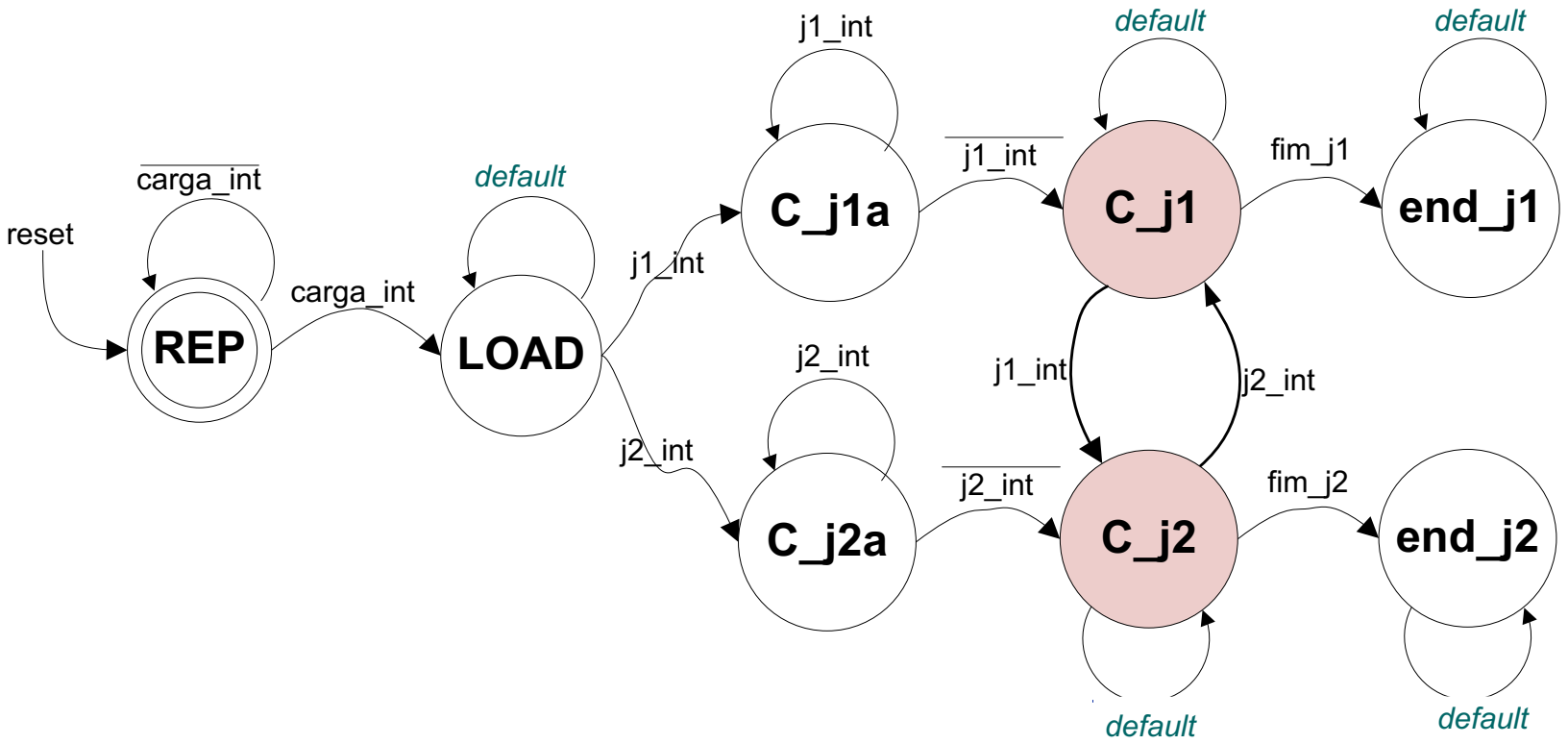
MÁQUINA DE ESTADOS E OPERAÇÃO

1. Reset: vai para o estado REP.
2. Carga: passa para o estado LOAD e carrega os dois contadores com o mesmo valor definido pelas chaves.



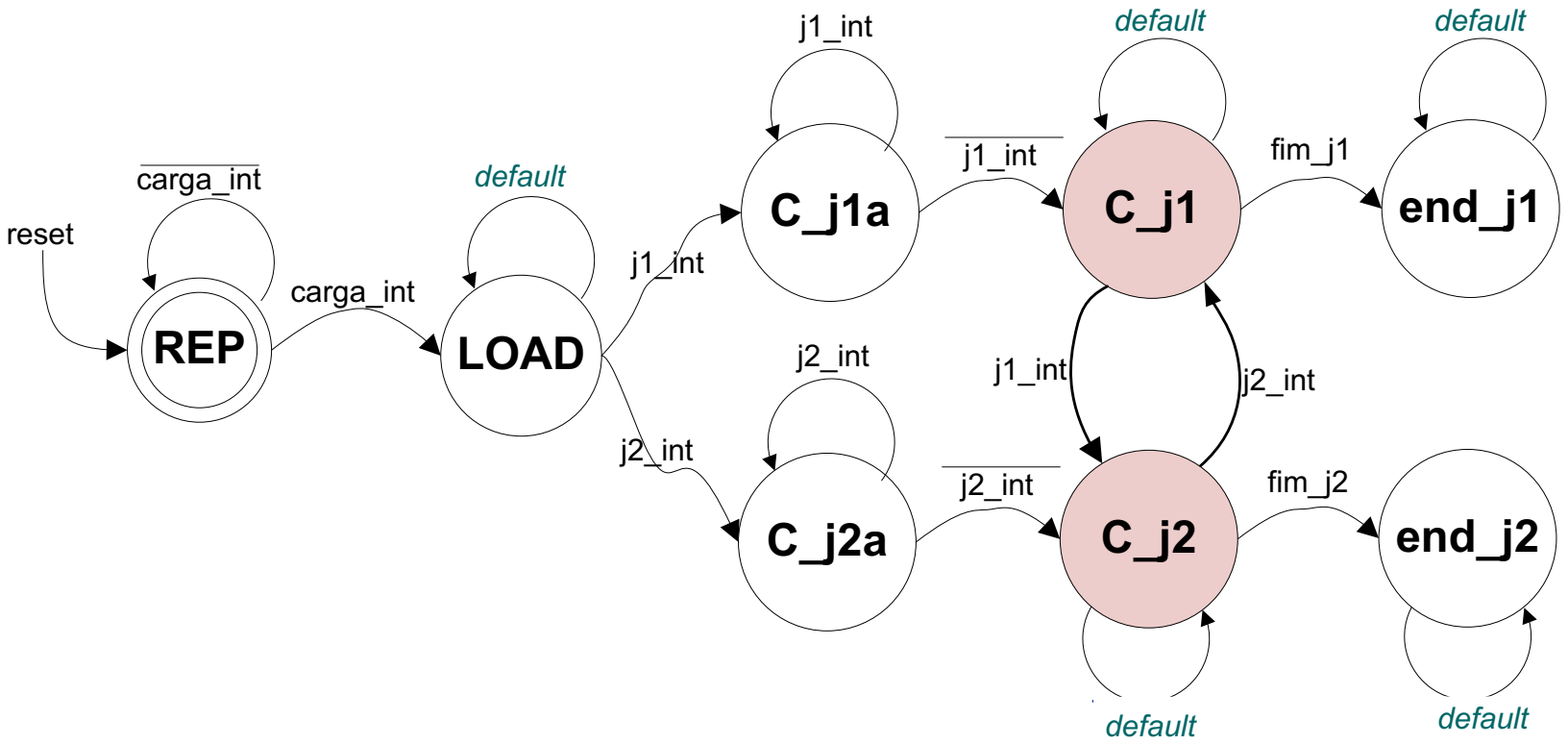
MÁQUINA DE ESTADOS E OPERAÇÃO

3. Quando um jogador pressionar sua tecla (ex: **j1**) passa-se para o estado **C_j1a**. Assim que o valor de **j1_int** voltar a 0, começa-se a contagem do cronômetro do jogador → estado **C_j1** (estado intermediário para evitar *glitch* com as teclas).



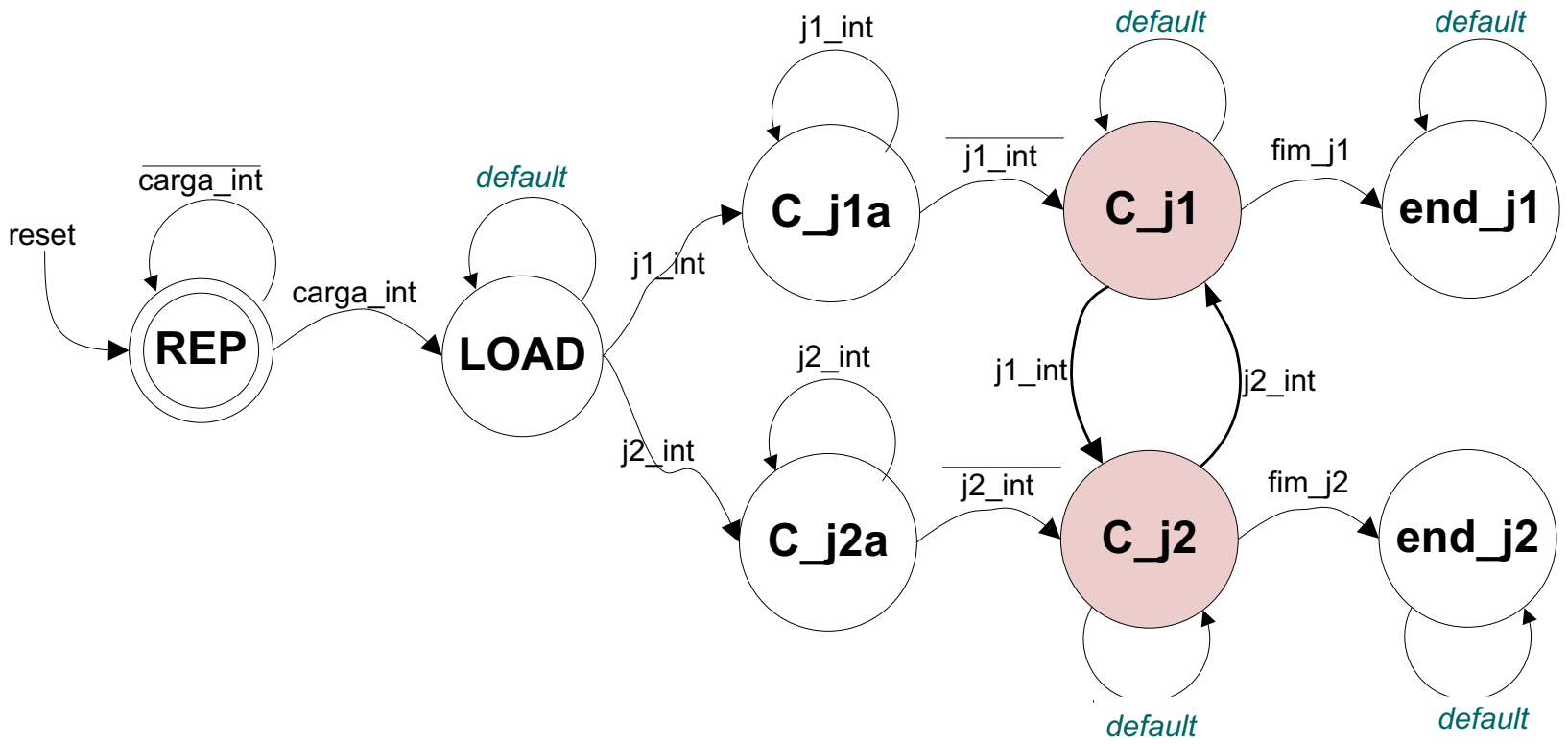
MÁQUINA DE ESTADOS E OPERAÇÃO

4. O jogador que iniciou (ex: **j1**) pressiona **j1** novamente e a contagem do jogador dois (**j2**) inicia. Os jogadores irão alternar suas jogadas até que a contagem de um deles expirar (00:00).



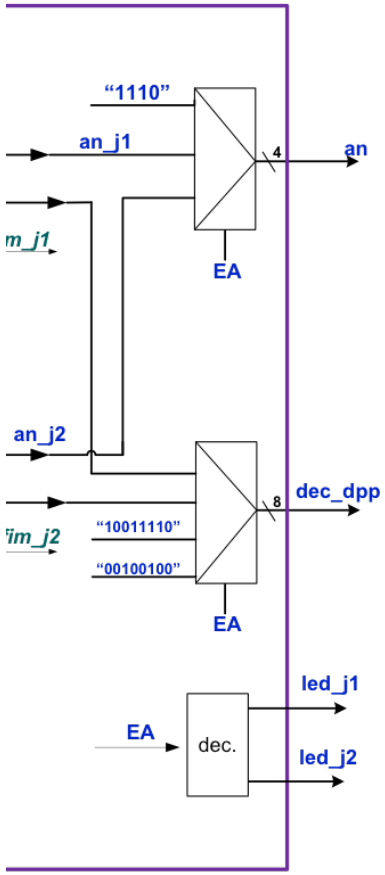
MÁQUINA DE ESTADOS E OPERAÇÃO

5. Quando a contagem expirar o *display* irá mostrar o jogador que venceu.
   - ✓ Valor 1 ou 2.



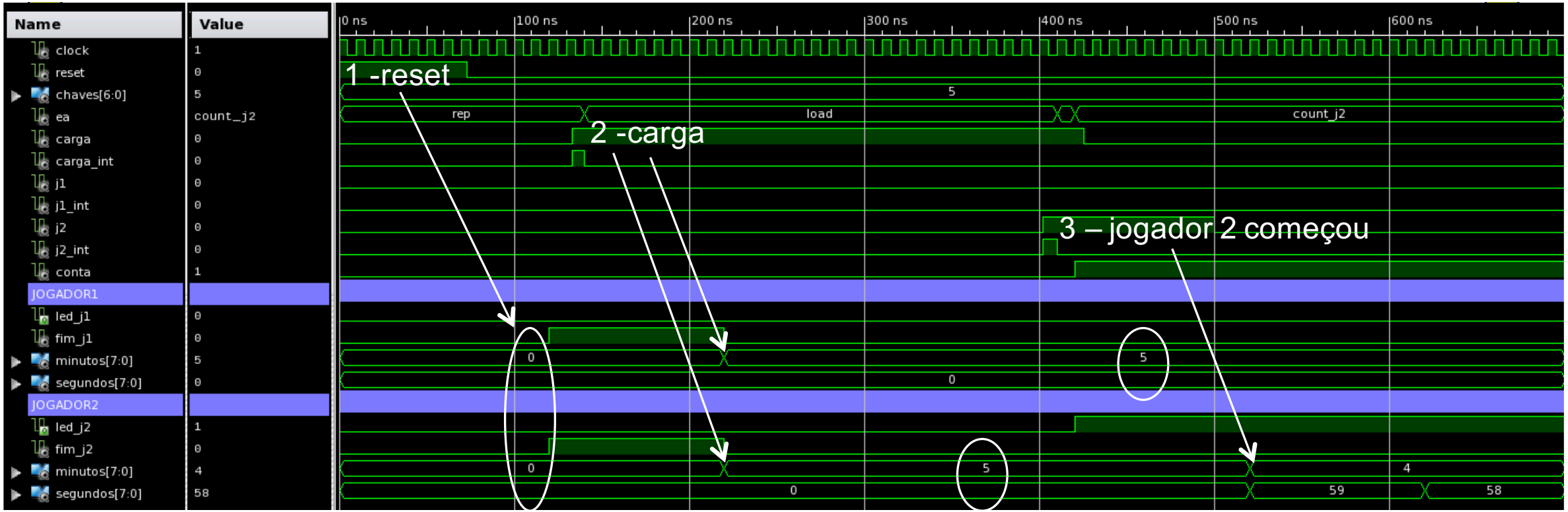
MÁQUINA DE ESTADOS E OPERAÇÃO

5. Quando a contagem expirar o *display* irá mostrar o jogador que venceu.
   - ✓ Valor 1 ou 2.



- **an** alterna entre o cronômetro 1 ou 2 em função do estado atual.
  - ✓ Quando um jogador ganhar fica ativo apenas o display da direita (com valor 0).
- **dec_dpp** alterna entre o cronômetro 1 ou 2 em função do estado atual.
  - ✓ Quando um jogador ganhar exibe-se 1 ("10011110") ou 2 ("00100100") no display da direita.
- Em função de quem está jogando um determinado *led* acende.

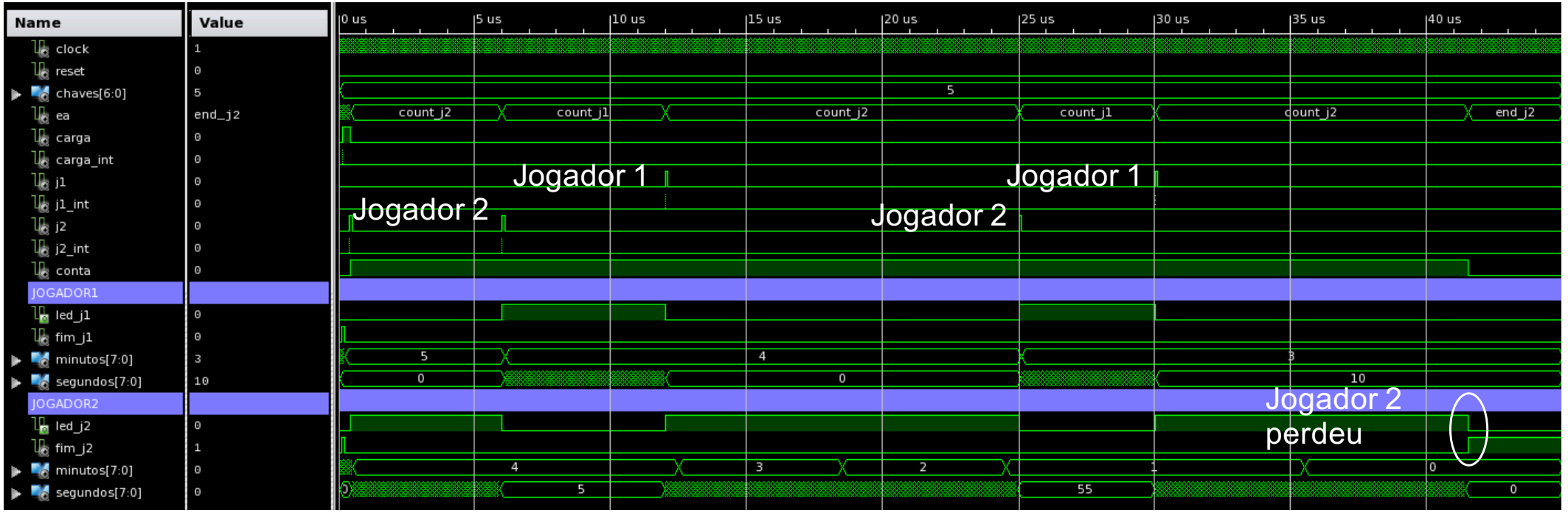
SIMULAÇÃO


Name
Value
clock
reset
chaves[6:0]
ea
carga
carga_int
j1
j1_int
j2
j2_int
conta
JOGADOR1
led_j1
fim_j1
minutos[7:0]
segundos[7:0]
JOGADOR2
led_j2
fim_j2
minutos[7:0]
segundos[7:0]
1
0
5
count_j2
0
0
0
0
0
1
0
0
5
0
1
0
4
58
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
1 -reset
2 -carga
3 – jogador 2 começou
rep
load
count_j2
5
0
5
0
0
5
4
0
59
58

SIMULAÇÃO

Sequência de jogadas:

1. Jogador 2 começa em 402 ns e termina sua primeira jogada em 6002 ns.
2. Jogador 1 termina sua primeira jogada em 12002 ns.
3. Jogador 2 termina sua segunda jogada em 25002 ns.
4. Jogador 1 termina sua segunda jogada em 30002 ns.
5. Jogador 2 expira o seu tempo e perdeu!!!!



SUGESTÃO PARA ESTRUTURA DE CÓDIGO

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity relogio_xadrez is
  generic ( CLOCK_FREQ : integer := 25000000 ); -- Usado para parametrizar o divisor de clock
  port ( ... ); -- Pinos conforme o diagrama de blocos ou o arquivo UCF
end relogio_xadrez;

architecture relogio_xadrez of relogio_xadrez is

  -- DECLARAR OS SINAIS NESSÁRIOS
  type states is (REP, LOAD, COUNT_J1a, COUNT_J1, COUNT_J2a, COUNT_J2, END_J1, END_J2);
  signal EA, PE : states;

begin
```

SUGESTÃO PARA ESTRUTURA DE CÓDIGO

```
begin
  a1: entity work.edge_detector port map (clock=>clock, reset=>reset, din=> carga, rising=>carga_int);
  a2: entity work.edge_detector port map ( ... j1_int);
  a3: entity work.edge_detector port map ( ... j2_int);
  i_cron_j1: entity work.dec_cron -- cronometro p/ jogador 1
             generic map (CLOCK_FREQ => CLOCK_FREQ )
             port map ( ... );
  i_cron_j2: entity work.dec_cron -- cronometro p/ jogador 2
             generic map (CLOCK_FREQ => CLOCK_FREQ )
             port map ( ... );
  --------- DOIS PROCESSOS PARA A MÁQUINA DE ESTADOS ------------
  -- Sinais controlados pelo Estado Atual da máquina de estados - combinacional
  led_j2  <= '1' when EA=COUNT_J2 else'0';
  led_j1  <= '1' when ...;
  conta   <= '1' when ...;
  vez_j1  <= '0' ... else '1';
  vez_j2  <= '0' ... else '1';
  an      <= an_j1 ...;
  dec_ddp <= dec_ddp_j1 when ...;
end relogio_xadrez;
```

TRABALHO

RESUMO DO TRABALHO 3A

- O **Trabalho 3A** (T3A) consiste em um arquivo compactado (.zip) contendo:
  - ✓ Um relatório em PDF descrevendo as implementações dos problemas.
  - ✓ Fonte do cronômetro inicial, não modificado, compatível com o testbench.
  - ✓ Fonte do relógio de xadrez, compatível com o testbench.
  - ✓ Dicas:
    - Você irá encontrar no material de apoio um testbench para cada problema proposto! Use ele para testar o código do seu projeto.
    - Você irá encontrar no material de apoio um detector de borda. Tenha cuidado para utiliza-lo, pois existe 2 versões, uma voltada a simulação e outra para síntese.