Projeto 2 – Sistemas Digitais III

Cofre de hotel

Lucas Toledo Pastori - 11.117.775-4

Carlos Henrique - 11.117.881-0

Turma 720

Professor Pedro Benko

Introdução

O objetivo desse projeto é fazer a implementação de um cofre utilizando redes de Petri. Redes de Petri é representação matemática para sistemas distribuídos discretos. Uma rede de Petri é formada por arcos, transições e lugares.

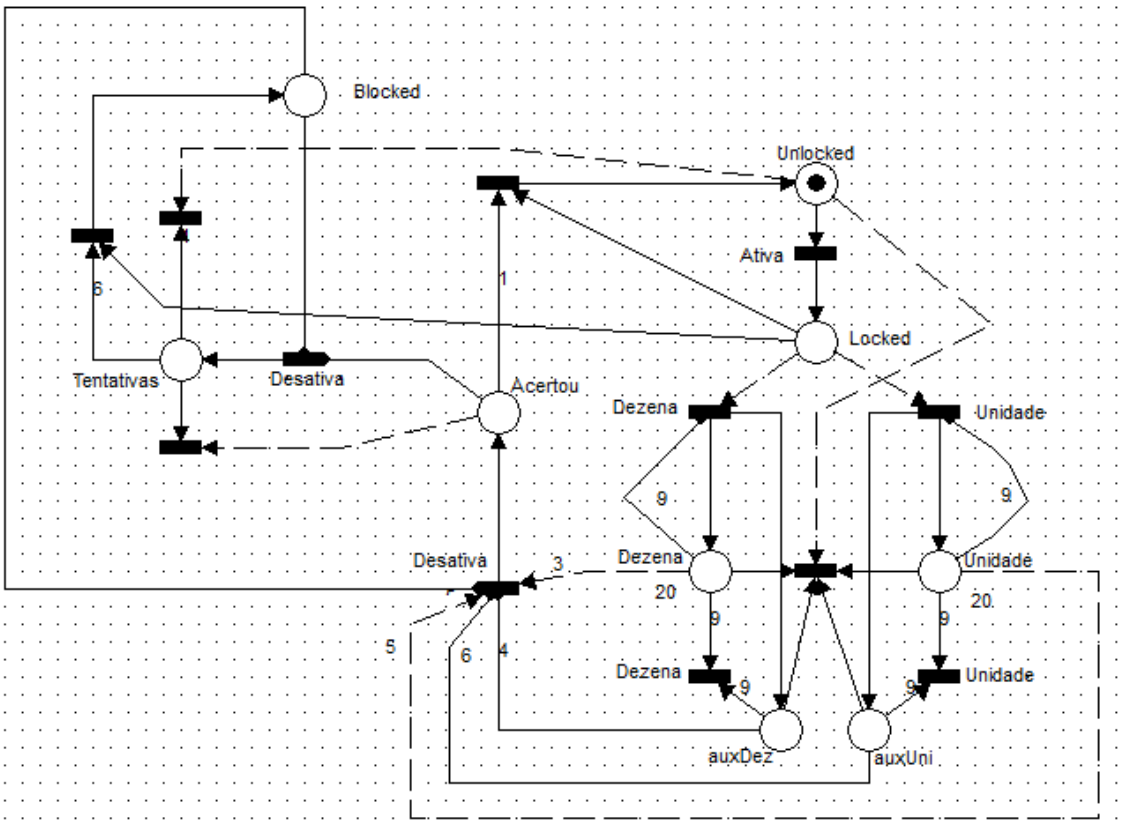
Para os dois alunos que implementaram esse projeto, a dupla foi chamada de dupla E, o número de tentativas permitidas para liberar o cofre são 6 e a senha para desbloquear o cofre é 35.

Deseja-se implementar o controle da abertura e fechamento de um cofre para hotéis. De início, o cofre estará aberto, com o led UNLOCKED acesso, permitindo assim que os hospedes coloquem seus objetos pessoais dentro dele, logo após isso, quando fechar a porta do cofre deve ser pressionado o botão ATIVA, fazendo assim o cofre ficar trancado (apenas o led LOCKED acesso).

Para digitar a senha do cofre, será usados os botões DEZ e UNID, a senha que estiver sendo colocada estará no display da senha. Colocando a senha desejada e após isso pressionar o botão DESATIVA, caso a senha esteja correta, o cofre será destravado e o led UNLOCKED ficará acesso. Porém, caso a senha esteja incorreta, aparecerá “1” no display de tentativas, esse valor indicado no display de tentativas será incrementado a cada tentativa erradas, a partir do momento que esse número ultrapassar 6, o cofre será bloqueado, com o led de BLOCKED aceso e o led LOCKED apagado. A partir desse momento, o cofre só pode ser aberto pelo gerente, quando ele pressionar o botão RESET.

Implementação

Para implementar e desenvolver a lógica do projeto foi utilizado inicialmente o programa HPsim, onde foi construída a rede de Petri do problema. Abaixo, a imagem da rede de Petri finalizada e que servirá de apoio para a explicação da lógica.



Primeiramente foram definidos quais seriam os lugares e transições principais, onde seria feita a representação de luzes, botões e chaves. Na situação inicial de reset do sistema, temos o cofre aberto que é representado pela marca no lugar Unlocked, que permanece nesse estado até que o usuário coloque seus pertences no cofre e acione por meio de um botão a transição Ativa.

A marca então passa para o lugar Locked representando o fechamento do cofre. Em seguida é necessário a inserção de uma senha de dois dígitos para abrir o cofre, sendo a senha correta 35 neste caso. Dois lugares foram posicionados para acumular os números apresentados no visor, assim o usuário pode incrementar a dezena ou a unidade por meio dos botões do painel, e esses acionam suas respectivas transições.

Com os lugares inicialmente em 0 marcas, o usuário aciona os botões para incrementar os números e selecionar sua senha. O incremento é feito pelo arco de peso 1, e ao chegar no número limite de nove marcas, o lugar inibe a transição de incremento e para fazer o ciclo e retornar para zero marcas os botões de incremento acionam uma transição consumidora que elimina todas as marcas desses dois lugares independentemente.

Para realizar a confirmação da senha, foi necessário adicionar mais dois lugares que são exatamente iguais aos acumuladores da unidade e dezena. Os lugares acumuladores principais fornecem o limite inferior da senha, que no caso são os próprios números da senha, e isso é feito por dois arcos de teste com seus valores correspondentes a função dos lugares. Os lugares auxiliares fornecem por meio de arcos inibidores o limite superior da senha, onde os valores dos arcos correspondem ao valor da senha mais um.

Esses quatro arcos, dois de teste com peso 3 e 5 e dois inibidores com peso 6 e 4, são ligados na transição Desativa, que corresponde ao botão que da o comando de abrir o cofre. Portanto essa transição Desativa só é habilitada com a senha 35, e caso o usuário tenha de fato selecionado a senha certa essa transição coloca uma marca no lugar Acertou, e assim o sistema sabe que é possível realizar a abertura do cofre.

Tendo uma marca no lugar Acertou e no lugar Locked, uma transição permanentemente ativada retira as marcas desses lugares e coloca uma marca no lugar Unlocked, assim o cofre está novamente na situação de aberto e o sistema volta para sua condição inicial.

Caso o usuário selecione a senha incorreta, ao ativar a transição Desativa, o lugar Acertou não recebe marcas e assim o sistemas sabe que o usuário errou a senha. O lugar Tentativas acumula a quantidade de vezes que o usuário erra a senha e tenta abrir o cofre, sendo assim a transição desativa neste caso coloca uma marca nesse lugar e o valor desse é mostrado no painel do cofre.

Nesta parte há um pequeno problema, onde mesmo se o usuário acertar a senha o sistema mostra que ele tem uma tentativa, mas isso foi solucionado adicionando uma transição consumidora que é ativada quando o usuário acerta a senha, o que torna imperceptível o incremento por conta da alta frequência de clock.

Quando o lugar Tentativas atinge seis marcas, o sistema entende que o usuário não poderá mais ter acesso ao cofre e deverá chamar o gerente do hotel para abri-lo novamente. Nesse caso o lugar Blocked recebe uma marca e com isso desabilita todas

as transições nomeadas de Desativa, ou seja, o usuário não tem mais o acesso à abertura do cofre.

O cofre só pode ser aberto então se o gerente der um reset no sistema do cofre e assim ele retorna à situação inicial. Uma outra observação a ser feita é que ao ter uma marca no lugar Unlocked o sistema consome todas as marcas acumuladas durante o processo para sempre retornar à situação inicial.

Código em VHDL

Abaixo, o código que traduz a lógica da rede de Petri para o Xilinx-ISE.

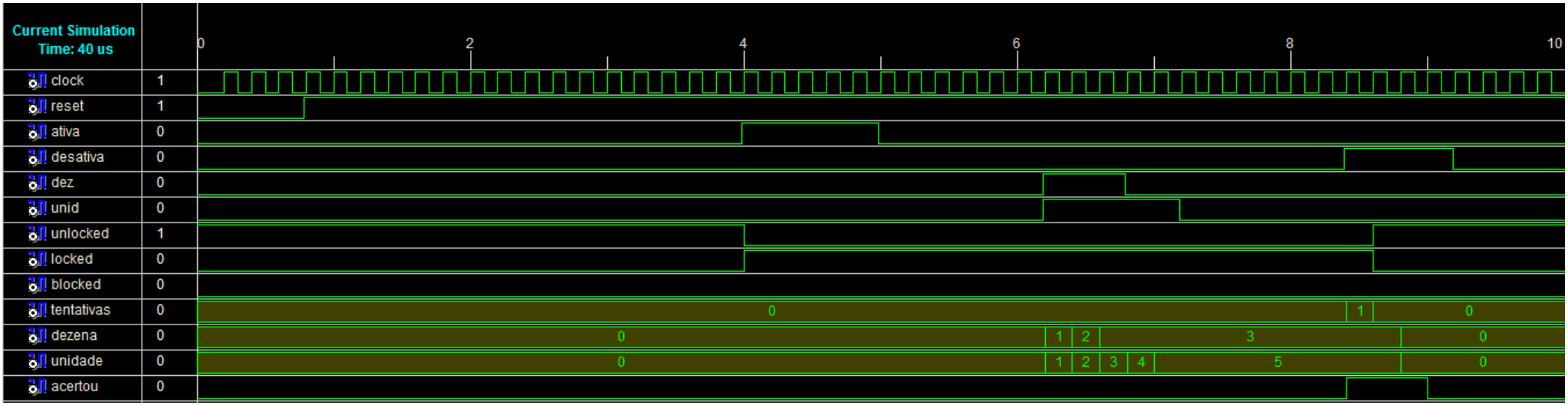
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity cofre is
port
( clock, reset : in std_logic;
  ativa, desativa, dez, unid : in std_logic;
  unlocked, locked, blocked : out std_logic;
  tentativas, dezena, unidade : out integer
);
end cofre;

architecture Behavioral of cofre is

signal unlk, lkd, blk, acertou : std_logic;
signal cdez, cuni, tent, auxcdez, auxcuni : integer;

begin
process (clock, reset)
begin

if reset = '0' then

  unlk<='1'; lkd<='0'; blk<='0'; acertou<='0'; cdez<=0; cuni<=0; tent<=0; auxcdez<=0; auxcuni<=0;

elsif clock'event and clock='1' then

--unlocked
if ativa = '1' then
unlk<='0';
elsif acertou = '1' and lkd= '1' then
unlk<='1';
end if;
```

```
--locked
if ativa= '1' and unlk = '1' then
lkd<='1';
elsif acertou = '1' and lkd = '1' then
lkd<='0';
elsif tent = 6 and lkd = '1' then
lkd<='0';
end if;
--dezena
if cdez = 9 then
cdez<= 0;
elsif unlk = '1' then
cdez<= 0;
elsif lkd= '1' and dez = '1' and cdez < 9 then
cdez<= cdez+1;
end if;
--unidade
if cuni = 9 then
cuni<= 0;
elsif unlk = '1' then
cuni<= 0;
elsif lkd= '1' and unid = '1' and cuni < 9 then
cuni<= cuni+1;
end if;
--auxcuni
if auxcuni = 9 then
auxcuni<= 0;
elsif unlk = '1' then
auxcuni<= 0;
elsif lkd= '1' and unid = '1' and cuni < 9 then
auxcuni<= auxcuni+1;
end if;
--auxcdez
if auxcdez = 9 then
auxcdez<= 0;
elsif unlk = '1' then
auxcdez<= 0;
elsif lkd= '1' and dez = '1' and auxcdez < 9 then
auxcdez<= auxcdez+1;
end if;
```

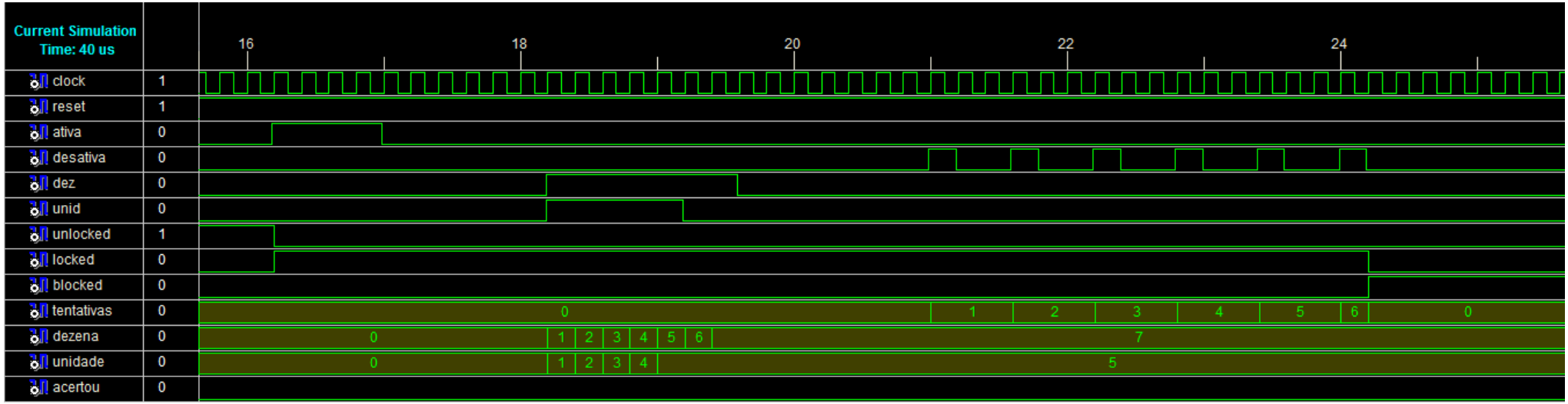
```
--acertou
if desativa = '1' and cdez>2 and cuni>4 and auxcdez<4 and auxcuni<6 then
acertou<='1';
elsif acertou = '1' and lkd = '1' then
acertou<='0';
elsif unlk = '1' then
acertou<='0';
end if;
--tentativas
if unlk = '1' then
tent<=0;
elsif tent= 6 then
tent<=0;
elsif acertou = '1' then
tent<=0;
elsif desativa = '1' and acertou = '0' and blk = '0' then
tent<=tent+1;
end if;
--blocked
if tent = 6 and lkd = '1' then
blk<='1';
end if;

end if;
end process;

unlocked<=unlk;
locked<=lkd;
blocked<=blk;
tentativas<=tent;
unidade<=cuni;
dezena<=cdez;

end Behavioral;
```

Simulação

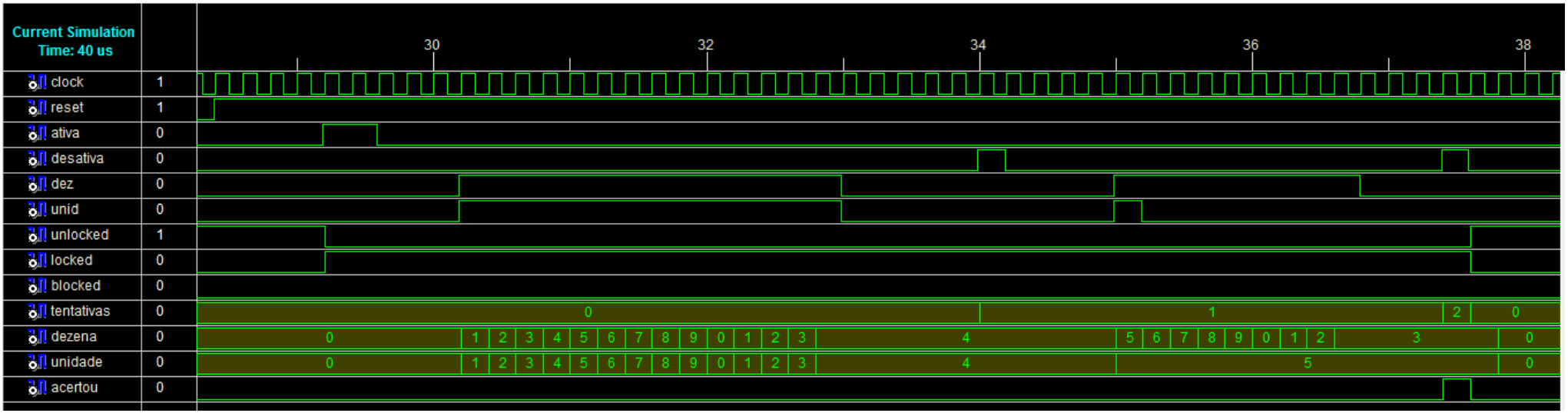
Funcionamento em situação normal:



Funcionamento em situação de bloqueio:



Funcionamento em situação de abertura com tentativas:



Conclusão e considerações

Ao longo desse projeto pudemos perceber o quão abrangente e importante é a modelagem de problemas pela rede de Petri, assim como o uso da descrição pela linguagem VHDL.

A modelagem do problema em rede de Petri foi relativamente simples, pois após entendermos bem o que deveria ser feito, criamos uma rede muito simples e otimizada que solucionava o problema. Assim, a parte da implementação em VHDL também ficou bem simples por conta da rede estar bem reduzida.

A lógica do projeto se mostrou eficiente, pois com a simulação pode-se perceber que todos os requisitos do roteiro foram atendidos, e isso mostra que o resultado foi um sucesso.