Universidade Federal de Santa Catarina
Centro Tecnológico - CTC
Departamento de Engenharia Elétrica - EEL

Juliana Silva Pinheiro (16100735)
Luciana Marasca Wiener (16100737)

01208A

<julianas.pinheiro@outlook.com>
<luciana.wiener@gmail.com >

Relatório de Projeto Final EEL5105
2016.1

Florianópolis, 09 de julho de 2016.

Conteúdo

1. Introdução

O projeto apresentado tem como objetivo implementar um circuito que possui funções parecidas com a de um celular, que permite utilizar uma senha e ligar para os contatos de uma agenda de acordo com o saldo inicial do circuito. Para isso, foram utilizados 22 arquivos VHD divididos em 6 blocos: Registradores, Agenda, Comparador, Contadores, Seletores e Controlador.

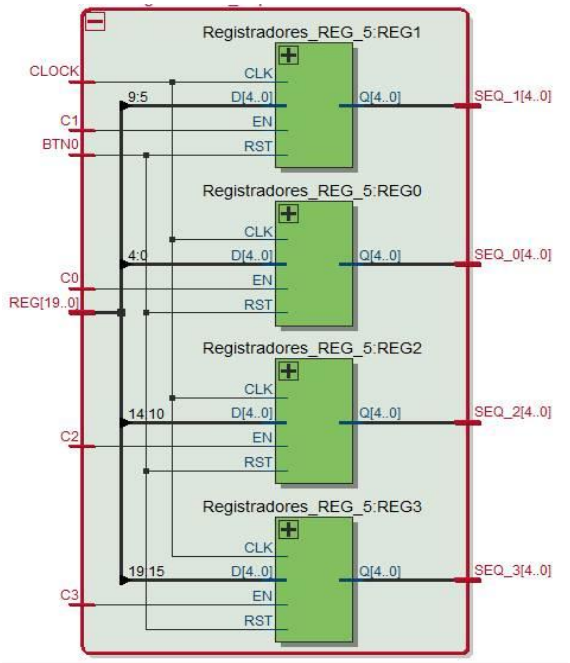
O bloco Registradores é composto por quatro registradores com a função de armazenar os valores inseridos na placa. O bloco Contador gera dois contadores com frequências diferentes; o primeiro, descendente, para descontar o saldo, e o outro ascendente, para cronometrar o tempo de uma ligação. O Comparador tem duas funções: indica se o saldo gerado pelo Contador chega a zero e se a senha inserida nos Registradores está correta.

A Agenda é utilizada para acessar os nomes dos contatos já salvos na memória. O grupo Seletores é composto por um decodificador e dois multiplexadores, que têm o papel de controlar o o celular e os LEDs de cada estado. O Controlador é responsável por fornecer os sinais para os outros blocos e controlar quais ações serão realizadas. Uma descrição mais detalhada de cada bloco será feita nas próximas seções.

1.1 Registradores

- **DESCRIÇÃO**

O bloco registrador possui dois arquivos: Registradores_Topo e Registradores_REG_5. Nesse bloco, são descritos quatro registradores, cada um com 5 bits, para armazenamento de sequências de 20 bits provenientes do multiplexador 4:1 do bloco Seletores.



O Topo, representado ao lado, indica as portas a que são atribuídas as entradas e saídas. Os registradores possuem um **CLOCK** mútuo. **BTN0** é o RESET de cada registrador, ou seja, torna as saídas iguais a 0, independente de outros valores. Ele é ativo quando seu valor lógico é baixo. As entradas **C0**, **C1**, **C2** e **C3** são os *enables* de cada registrador e estão

ativadas quando seus valores são 1. A entrada REG de 20 bits, vinda do multiplexador 4:1 dos Seletores, pode tomar valores de senha, de nomes da Agenda, da conta ascendente do Contador ou de “0”. As saídas **SEQ_0**, **SEQ_1**, **SEQ_2** e **SEQ_3** são as divisões dos 4 registradores.

- **SIMULAÇÃO**

Durante a PARTE 1, o CLOCK é definido com 100ps de período na simulação. REG toma um valor diferente de 0, porém, o resultado da simulação é 0. Isso acontece porque BTN0 está ativado, ou seja, o reset de cada registrador tem nível logico 0.

Já na PARTE 2, ao mudar o sinal de BTN0 para 1, os valores de saída continuam em 0, pois todos os C[0...3] estão inativos. Ou seja, para que o valor seja transmitido, em qualquer um dos registradores, BTN0 precisa estar desabilitado e pelo menos um dos C[0...3] precisa estar ativo.

Agora, na PARTE 3, BTN0 continua 1. Quando mudamos os valores de C0 e C2 (enable) para 1, apenas as SEQ0 e SEQ2 mudam o seu valor, pois estes correspondem aos registradores REG0 e REG2. Isso só acontece após a próxima subida do CLOCK depois da troca de valores de C, e isso garante que os valores sejam estáveis antes de realizar a troca.

Na PARTE 4, todos os enables C[0...3] foram definidos em 1, e, portanto, após a próxima subida do CLOCK, todos os valores do REG são propagados às sequências.

Para demonstrar o funcionamento do BTN0 (reset) de todos registradores, durante a PARTE 5, BTN0 retorna a zero. Por consequência, todas as saídas se tornam zero, independentemente do valor de seu respectivo enable C.

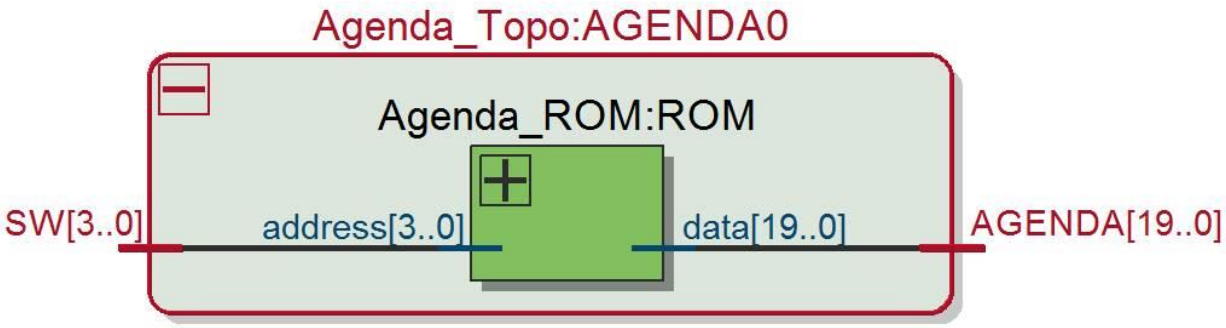
1.2 Agenda

O bloco Agenda foi projetado com dois arquivos VHDLs: Agenda_Topo e Agenda_ROM.

O arquivo Agenda_ROM descreve uma memória ROM com 4 bits, para que sejam armazenados os 16 nomes da agenda, compostos por 20 bits. A ROM aloca os nomes (valores em binário) para cada

espaço da memória, que nesse caso tem tamanho 15. Assim, quando o usuário insere um valor x de 0 a 15, a saída retorna o valor correspondente ao nome da pessoa armazenada no espaço x da memória.

O arquivo Agenda_Topo, representado na imagem abaixo, liga a entrada *address* da ROM aos quatro primeiros seletores SW da placa, para a escolha de um nome, e a saída *data* à AGENDA, com 20 bits, com o valor do nome, que será utilizado em outros blocos.

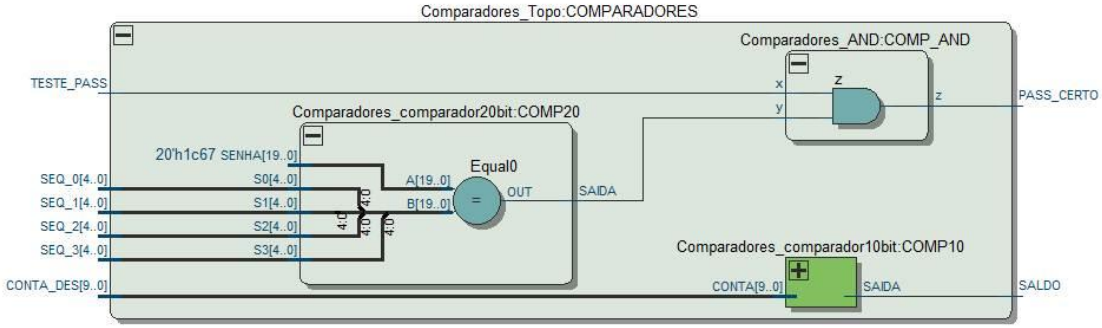


O arquivo **decod7seg** especifica os valores binários para as letras e números da agenda. Para preencher a tabela com os nomes, foram utilizados 5 bits para cada letra. Considerando que o display também precisaria de números para seu funcionamento, sobrariam apenas 21 letras para serem feitas. Visando maior aproveitamento, letras que possuem o mesmo desenho que o número, como, por exemplo, I e 1, 2 e Z e as duas letras H e X, fazem uso do mesmo valor binário.

1.3 Comparador

- **DESCRIÇÃO**

O bloco Comparadores é composto por dois comparadores de 10 e 20 bits e uma lógica combinacional onde é implementada a operação *and*. Para criar os comparadores, foi realizada a abordagem comportamental porque foi a opção que se era mais familiarizada e utilizada com mais frequência nas aulas do laboratório.



A imagem acima descreve o bloco completo Comparador. A visualização está expandida na lógica combinacional, podendo ser conferida a operação lógica AND. Também pode-se visualizar o comparador de 20 bits. Ele recebe os valores **SEQ_0**, **SEQ_1**, **SEQ_2** e **SEQ_3** vindos do registrador, os concatena em uma sequência **SEQ**. Utilizando o código "*when SEQ = SENHA*", o valor de saída se torna 1 quando a sequência de entrada for igual a senha do celular e "*else 0*" para indicar que a senha não é igual, ou seja, está incorreta.

A lógica combinacional, neste caso, serve para verificar se a saída do comparador de 20 bits e a entrada **TESTE_PASS** ambas são iguais a um. Se forem, **PASS_CERTO** se torna 1 e a senha está correta. O comparador de 10 bits recebe o valor da contagem descendente **CONTA_DES** do Contador. Retorna 0 em **SALDO** se a contagem chega a 0, senão continua em 1.

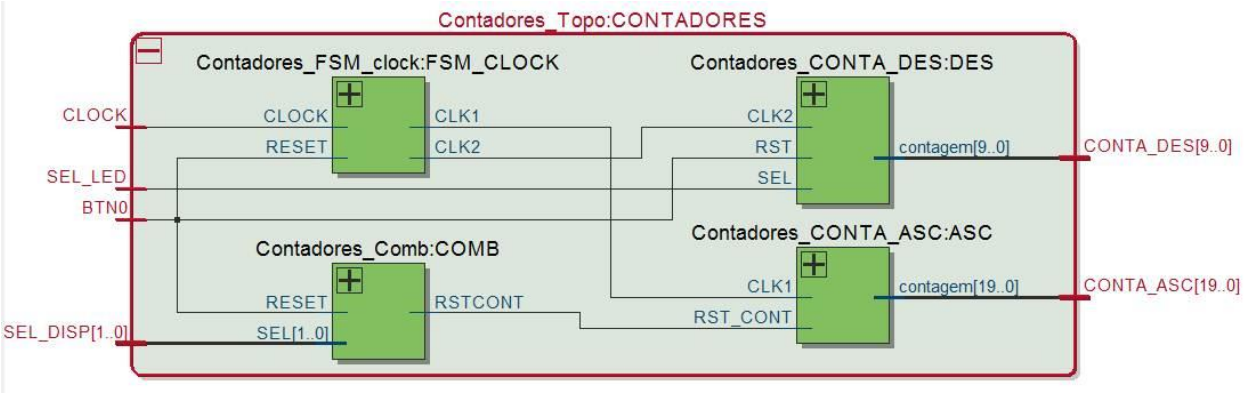
- **SIMULAÇÃO**

Para a simulação do comparador, são necessárias as entradas **SEQ_0**, **SEQ_1**, **SEQ_2**, **SEQ_3**, **TESTE_PASS** e **CONTA_DES**. A saída para **PASS_CERTO** somente será 1 quando a variável TESTE_PASS for 1 e a sequência (**SEQ_0**, **SEQ_1**, **SEQ_2**, **SEQ_3**) for igual a "00000001110001100111", que corresponde à senha correta do celular. A saída para **SALDO** será 1 enquanto a entrada CONTA_DES for maior que 0. Quando o contador chegar a 0, SALDO terá valor 0.

Na PARTE 1, a primeira tentativa de inserir uma senha é errada, o valor de TESTE_PASS é 0 e a saída, como esperado, é 0. Na PARTE 2, o valor de TESTE_PASS é alterado para 1, mas o valor da saída não muda, pois a sequência ainda está incorreta. Já na PARTE 3, a sequência correta é armazenada e a saída PASS_CERTO se torna 1. Durante as simulações anteriores, constantemente o valor de CONTA_DES é diferente de 0 e, portanto, a saída SALDO era 1. Quando chegamos à PARTE 4, o valor de CONTA_DES chega a 0, e leva também a saída SALDO para 0.

1.4 Contadores

O bloco Contadores possui cinco arquivos: Topo, FSM_clock, CONTA_ASC, CONTA_DES, e Comb. Esse bloco é responsável por criar os contadores crescente e decrescente, utilizando um FSM_clock e uma lógica combinacional, como pode ser observado na imagem abaixo.



- **FSM_clock**

O FSM_clock recebe um clock próprio para ele e a entrada BTN0, que é o seu reset. Para obter os relógios CLK1 e CLK2, de 1Hz e de 9Hz, não foram atribuído estados. Foram criados dois contadores, que são incrementados com a subida de clock, separando o código em dois *processes* diferentes. O *process* para o CLK1 descreve um contador1 que incrementa até “49.999.999”, enquanto CLK1 permanece em nível lógico baixo. Ao chegar nesse valor, ele sinaliza uma subida de CLK1. No *process* para o CLK2, o relógio permanece em “0” e sobe quando o contador2 chega a "5.555.555". Os valores utilizados foram estes devido a limitações na placa, como foi visto em aula.

- **CONTA_ASC**

Para realizar a contagem crescente, foi utilizado como entrada o **CLK1** já estabelecido no FSM_clock e um **RST_CONT** vindo de BTN0, capaz de reiniciar a contagem, ou seja, fazê-la voltar a zero. Para representar o formato MM:SS, foram criados 4 sinais que representam as unidades e dezenas dos minutos e segundos (sinais *minutosD, minutosU, segundosD* e *segundosU*). Assim, quando segundosU chega a 9, ele retorna a 0 e o sinal segundosD se torna 1. Da mesma forma funciona o sinal minutosU. Quando segundosD chega a 59, ele volta a ser 0 e minutosU passa a ser 1. Desse modo, a saída **CONTA_ASC[19...0]** consegue se adaptar ao formato.

- **CONTA_DES**

Para realizar a contagem decrescente, foi utilizado como entrada o **CLK2** já estabelecido no FSM_clock, um **RST** que é gerado pela logica combinacional Comb do bloco e um **SEL** proveniente do bloco Controlador, que funciona como um *enable*. A contagem é feita partindo do saldo inicial de 1011_{10}

(1111110011_2) até chegar a 0 e só acontece se SEL for igual a 1. Se o RST for acionado, o contador volta ao saldo inicial e recomeça.

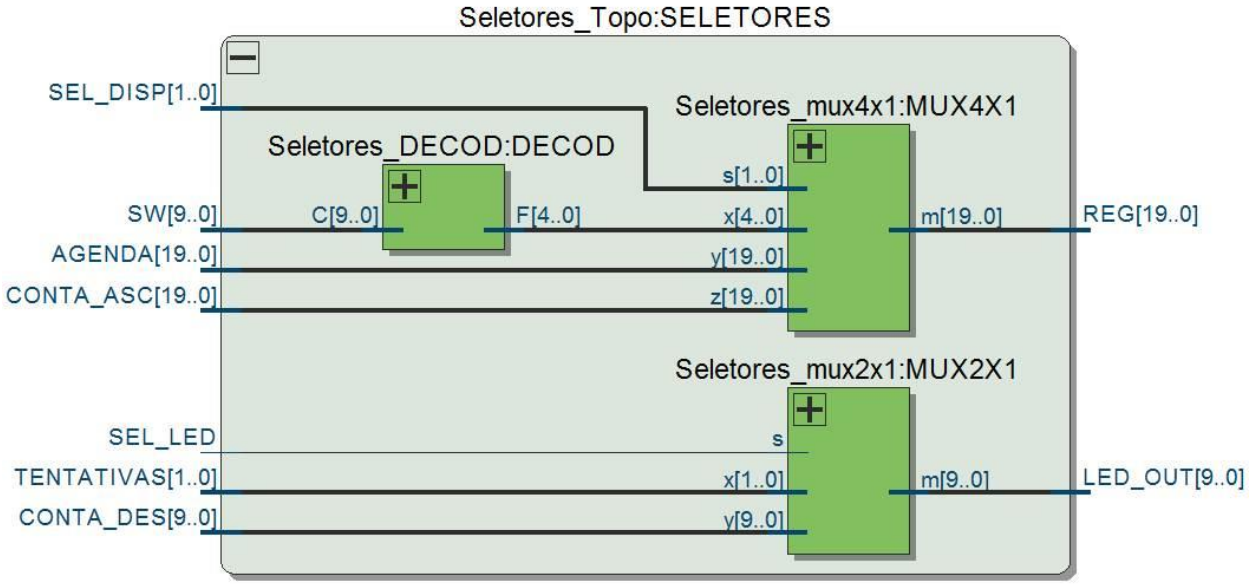
- **Comb**

O arquivo Comb é responsável por receber o valor de **SEL_DISP[1...0]** vindo do bloco Controlador e o valor **BTN0** com a função de reset. Ele faz uso da abordagem comportamental. A saída **RST_CONT** é '0' (ou seja, está ativo) quando BTN0 for '0' ou quando SEL for diferente de "11". Senão, é '1' (desativado).

No início, BTN0 é 0, ou seja, os valores iniciais de CONTA_ASC e CONTA_DES são 0 e 111110011, respectivamente. Para ser feita a simulação, os relógios de cada contagem foram reduzidos. Ao trocar o valor de BTN0, e alguns ciclos de clock, as duas saídas começam a fazer sua contagem.

1.5 Seletores

Esta unidade consiste em um decodificador e dois multiplexadores. Possui as entradas **SW[9..0]**, que entra no DECOD, **SEL_DISP[1..0]**, **AGENDA[19..0]**, **CONTA_ASC[19..0]**, que entram no multiplexador 4:1, **SEL_LED, TENTATIVAS[1..0]** e **CONTA_DES[9..0]**, que entram no multiplexador 2:1. As saídas são **LED_OUT[9..0]**, de 10 bits, do mux 2:1 e **REG[19..0]**, de 20 bits, do mux 4:1.

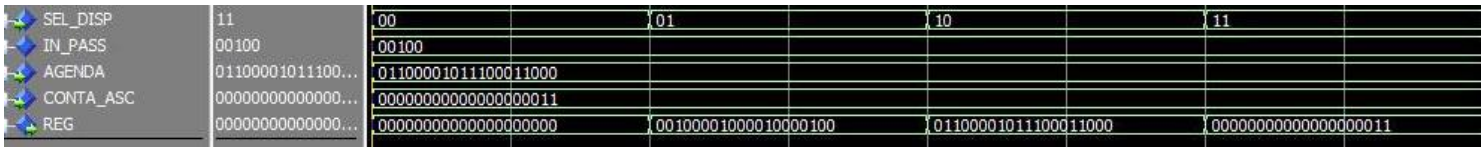


- **DECOD:**

O decodificador é responsável por transformar os valores de 10 bits inseridos nos switches 0 a 9 para valores de 5 bits. Nesse componente, foi utilizado a abordagem comportamental. Ele é útil para a inserção da senha, já que esta possui apenas 20 bits. O decodificador foi descrito de forma que, se fossem acionados 0 ou mais que 1 switch ao mesmo tempo, o display ficaria vazio no espaço alocado para o dígito.

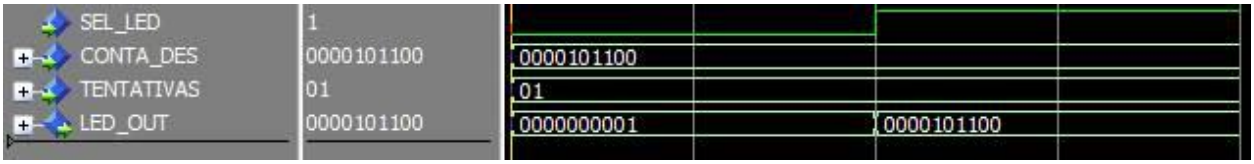
- **MUX 4:1**

Faz a escolha do que será mostrado no display, através de *SEL_DISP[1..0]*. Cada estado do controlador terá um display diferente. Neste caso, se o seletor for *"00",* a saída *REG* será 20 0's; se for *"01"*, os dígitos da senha; se tiver valor *"10",* os nomes da agenda; e por fim, se *"11"*, o contador ascendente do bloco Contadores.



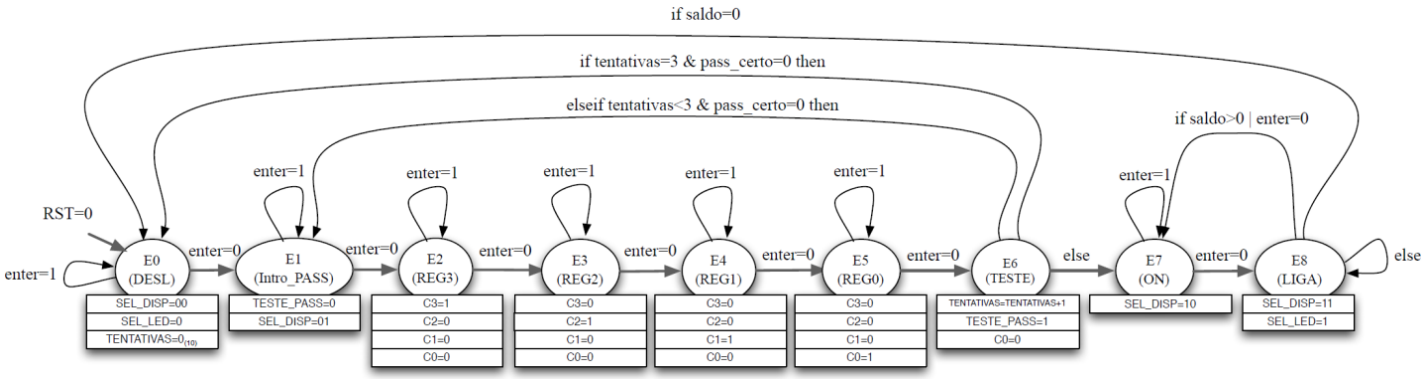
- **MUX 2:1**

Decide quais LEDs ligarão através de *SEL_LED*. No estado entre INTRO_PASS e ON, com **SEL_LED = 0**, mostrará o número de tentativas através de *LED_OUT*. Se **SEL_LED = 1**, no estado LIGA, mostrará a contagem decrescente do bloco Contadores.



2 Controlador

Este bloco é composto por dois arquivos: Topo e FSM_Control. Possui como entradas um **CLOCK**, um **BTN3** (enter), **BTN0** (reset), **PASS_CERTO** e **SALDO**, ambos provenientes do bloco Comparador. As saídas são **C0**, **C1**, **C2** e **C3**, que fazem o controle dos registradores, **ESTADOS[4..0]** para representar os estados, **SEL_DISP[1..0]**, **SEL_LED**, **TENTATIVAS[1..0**], saída indica o número de tentativas incorretas de senha, e **TESTE_PASS**.



O diagrama de blocos acima mostra o comportamento da FSM_Control. Para evitar problemas de latch ou unsafe behavior, as variáveis a serem usadas são definidas após o início do process e antes de codificar os estados da FSM, tomando valores como 0 ou algum valor inicial.

No estado inicial E0 (DESL) os contadores estão zerados, pois o celular está desligado. As saídas *SEL_DISP* e *SEL_LED* são "00" e '0' representando que o display será zerado. O número de tentativas TENTATIVAS_PROXIMO para introduzir a senha é iniciado em zero.

Ao apertar o botão KEY3 (Enter), o usuário entra em E1, em que é feito a inserção da senha na máquina. A saída *TESTE_PASS* é 0, porque a comparação com a senha correta não será feita neste estado. A saída *SEL_DISP* vira "01", o que significa que a senha será mostrada no display. Para melhor entendimento, se houver uma tentativa anterior armazenada nos registradores, essa tentativa é mostrada no display até ser inserido novos valores nos estados seguintes.

Em E2, E3, E4 e E5 (REG1, REG2, REG3 e REG4, respectivamente) a FSM aguarda o usuário selecionar os números da senha. Considerando que os registradores só funcionam com um enable C, o estado aguarda o seu respectivo enable ser acionado, de acordo com os registradores.

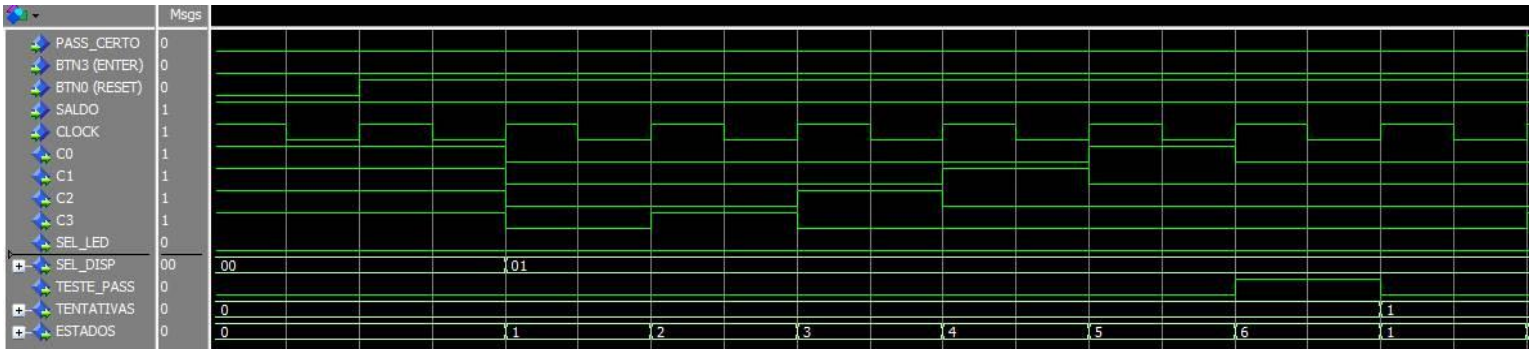
No estado E6, acontecerá a verificação da senha. Para isso, TESTE_PASS se torna '1' e o número de tentativas incrementa em 1. É neste estado que acontece o teste da senha e a verificação do limite. Caso três tentativas tenham sido realizadas, o celular volta ao estado E0, ou seja, desliga. Caso a senha seja incorreta, mas não se passou do limite de 4 tentativas, o sistema volta ao estado E1 e espera uma nova tentativa. Se a senha estiver correta, a FSM segue para E7 (ON). O uso de dois sinais TENTATIVAS_ATUAL e TENTATIVAS_PROXIMO é uma alternativa ao uso de um registrador, porque é preciso incrementar o próprio valor do sinal. Então, para salvar um valor anterior, se utiliza TENTATIVAS_ATUAL e TENTATIVAS_PROXIMO.

No estado E7 (*ON*) é permitido escolher um nome da agenda usando SW(4...0). No display, os nomes da agenda são mostrados. Ao apertar Enter, o nome apresentado no display é escolhido para realizar a ligação e a FSM passa ao estado LIGA. No estado E8 (*LIGA*), a contagem crescente realizada no bloco Contador é mostrada no display no formato MM:SS e a decrescente é visualizada através dos LEDs, em binário. Se o saldo chegar a 0, o celular é desligado. Caso o saldo for maior que 0 e a tecla Enter for pressionada, indicando que o usuário quer fazer uma nova ligação, a FSM retorna a E7 (ON).

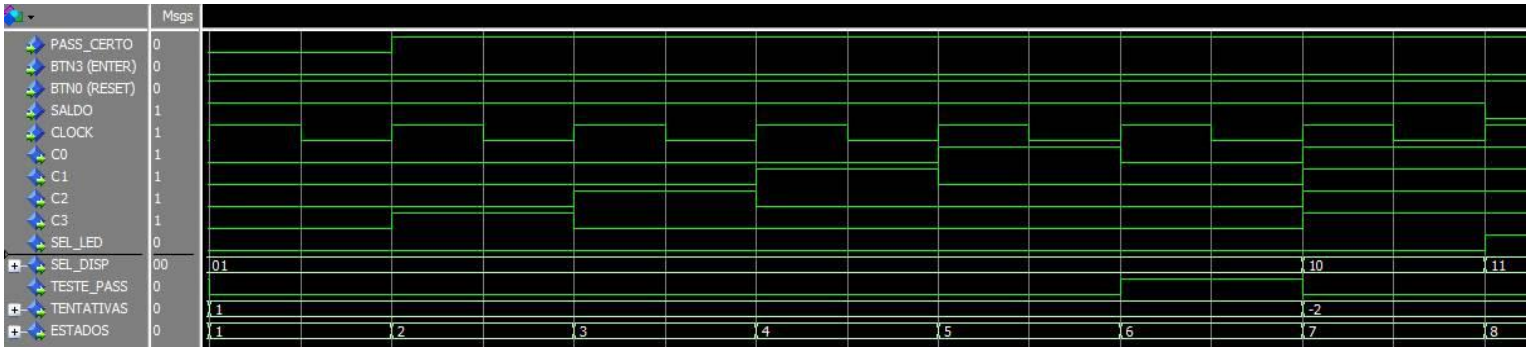
- **SIMULAÇÃO**

Para realizar uma simulação resumida, foi considerado que haveria um momento em que o reset estaria ligado (início), uma tentativa incorreta de senha e que o saldo chegaria a 0 após uma ligação. Isso demonstra o funcionamento das entradas e saídas, apesar de não abranger todos os casos.

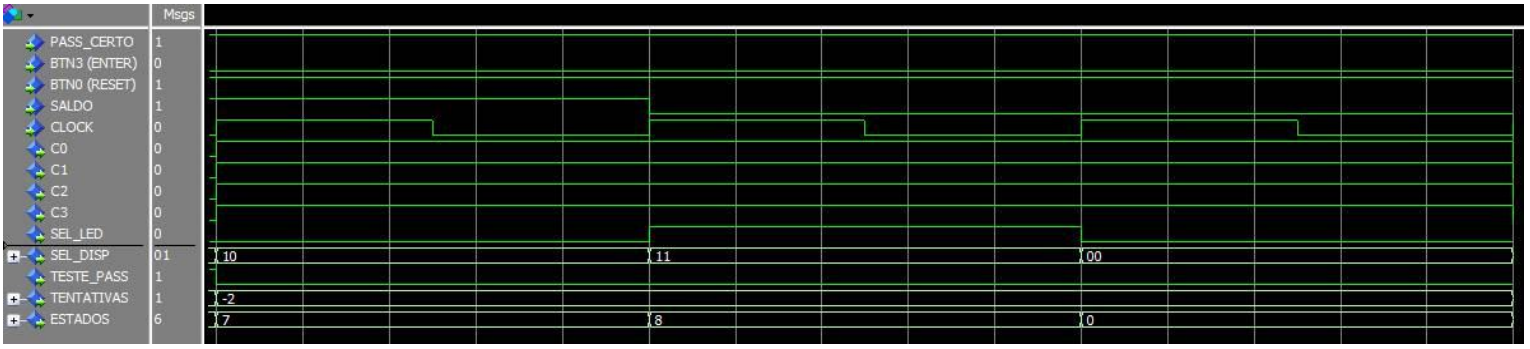
Na PARTE 1, iniciam-se os valores em 0. BTN0 está em nível lógico baixo e, portanto, as as saídas serão 0. Na PARTE 2, quando ele sobe para um, a FSM passa a E1. Logo após, passa a E2, E2, E4, E5 e E6, onde os dígitos são registrados através dos enables C0, C1, C2 e C3. Ao chegar em E6, na PARTE 3, a FSM volta a E1 pois PASS_CERTO = 0, ou seja, senha incorreta, e incrementa o valor das TENTATIVAS.



Durante a PARTE 4, ocorre uma nova tentativa de senha. Ao chegar na PARTE 5, PASS_CERTO torna-se 1, em que a senha está correta, para ser possível seguir na FSM. Deste modo, ao chegar em E7, as tentativas são incrementadas.



Na PARTE 6, a FSM passa ao estado LIGA e volta ao estado 0 pois SALDO foi modificado para 0.



3. Resultados e conclusões

Todos os blocos foram conectados pelo arquivo CELULAR.vhd, topo do projeto. Durante a criação e conexão de cada componente e bloco, foi possível entender a complexidade de construir um circuito com funções diferenciadas. Com as simulações, problemas de lógica com o código puderam ser resolvidos com facilidade. Ao simular na placa DE1, foi possível visualizar com clareza o funcionamento, os erros e o que podia ter sido feito diferente, e isso foi essencial para o trabalho.

Quanto aos testes com a placa, houve algumas falhas, mas causadas por erros no código ou na sintaxe, que puderam ser resolvidos logo após a detecção do problema. Na última simulação dos blocos e no último teste com a placa DE1, todos os resultados foram como esperados e não foram detectados erros, o que foi satisfatório.

Anexo A – Observações

O projeto, a princípio, parecia extremamente difícil devido a quantidade de componentes, mas se mostrou uma questão de estudo, esforço e tentativa.

Na elaboração de muitos dos componentes, possíveis erros foram encontrados diretamente na simulação de cada bloco, um caso sendo o incremento da saída TENTATIVAS na máquina de estados, que foi resolvido com a criação de dois sinais para possibilitar incrementar o valor. Porém, a maioria dos erros foram ajustados após passar o projeto à placa DE1. Um destes problemas foi que, apesar de ser possível passar pelos estados pressionando o botão KEY(3), os switches não funcionavam devidamente para inserir a senha. Reconheceu-se então o erro na definição na máquina de estado dos sinais de enable dos registradores, que foi de simples resolução.

Alguns outros pequenos problemas surgiram no uso da placa, mas se mostraram como erros em pequenos detalhes nos códigos por algum descuido. Consideramos que o uso da placa nos permitiu muito mais facilmente encontrar e resolver erros e problemas no projeto.