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

Diogo Junior de Souza (16100721)
Fabíola Maria Kretzer (16100725)

TURMA 1208A

<diogojrdesouza@gmail.com >
<fabiolakretzer@hotmail.com >

Relatório de Projeto Final EEL5105
2016.1

Florianópolis, 9 de Julho de 2016.

Conteúdo

1. Introdução

A ideia do projeto é implementar algo similar com o funcionamento de um telefone celular. O sistema inicia desligado, ao ser ligado, é preciso inserir um password (senha), que possui 4 dígitos decimais. Os dígitos devem ser inseridos com a ordem correta. Se eles forem incorretos, o password poderá ser inserido novamente. Isso ocorrerá até o password estar correto. Se o usuário errar 4 vezes o sistema será automaticamente desligado. Quando o password estiver correto, o usuário poderá inserir o número do nome da pessoa e o nome aparecerá nos displays, depois é só fazer a ligação. Assim haverá dois contadores, um ascendente que está interligado com o decodificador 7 segmentos, e indicará o tempo da ligação, em segundos e em minutos. E o contador descendente irá descontar do valor dos créditos de acordo com a frequência. Assim quando o saldo acaba o celular é automaticamente desligado. Se ainda estiver saldo poderá terminar a ligação e inserir outro número para ligar, isso sucessivamente até o saldo acabar. Quando o saldo terminar o sistema é desligado.

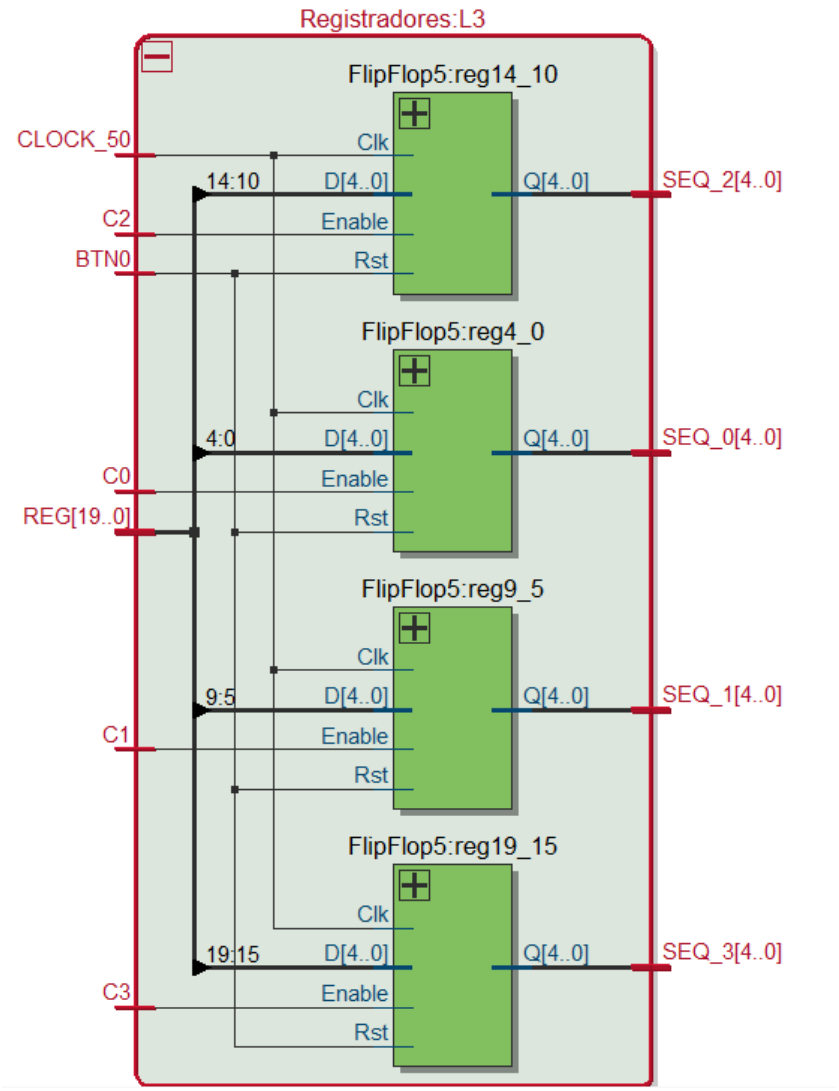
Para a descrição de toda essa estrutura foram necessários arquivos VHDL, separados em 6 blocos (registradores, contadores, selectores, comparadores, controlador, agenda), que foram unidos e interligados através de um bloco superior, chamado Topo. Cada bloco possui uma função. O bloco dos registradores é o responsável por gerar os dígitos do password. Assim o bloco dos comparadores verifica se o password inserido está correto e ainda se o saldo é igual a zero. Esse saldo é gerado por um contador decrescente que está no bloco dos contadores. Esse bloco também é o responsável por um contador ascendente, que contará os minutos e os segundos da ligação, cada um dos contadores irá ter uma velocidade (obtida através do clock) diferente. O bloco dos selectores selecionam o número de bits necessário para o password ser inserido e devem selecionar o que é esperado para cada estado. A agenda é onde estão definidos todos os 16 contatos, das quais se pode ligar. Já o bloco controlador está interligado com todos os outros blocos.

1.1 Registradores

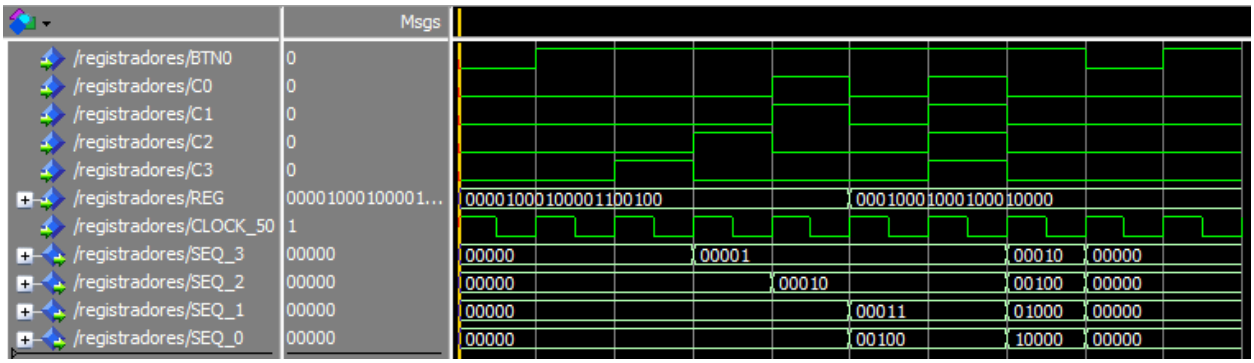
Para fazer esse bloco foram necessários dois os arquivos VHDLs: **Registradores** e **FlipFlop5**.

- **FlipFlop5:** esse componente possui uma abordagem comportamental. Neste arquivo está presente o funcionamento de um flip-flop.
- **Registradores:** esse componente também possui uma abordagem comportamental e é o responsável pelos quatro flip-flops de 5 bits cada um. As entradas C3, C2, C1 e C0 são saídas do bloco controlador e representam o enable de cada flip-flop. Se o reset for ativo as saídas dos flip-flops ficarão em zero. E voltará ao estado E0 (desligado).

O diagrama de blocos dos Registradores ficou conforme a imagem a seguir:



Simulando o comportamento dos Registradores no ModelSim obtemos o seguinte diagrama de tempo:



Podemos notar que cada saída *SEQ_3* à *SEQ_0* só recebe o valor que está na entrada *REG* quando as suas respectivas entradas *C3* à *C0* (que funcionam como “carga” ou “enable”) são ativadas.

O *BTN0* funciona como “Reset” e zera todas as saídas quando é ativado.

1.2 Agenda

A memória é um dispositivo capaz de armazenar dados e programas. Nesse projeto foi utilizada a memória ROM, que é um tipo de memória que permite apenas a leitura. As suas informações são gravadas pelo desenvolvedor do software ou do hardware uma única vez e após isso não podem ser alteradas ou apagadas, somente acessadas.

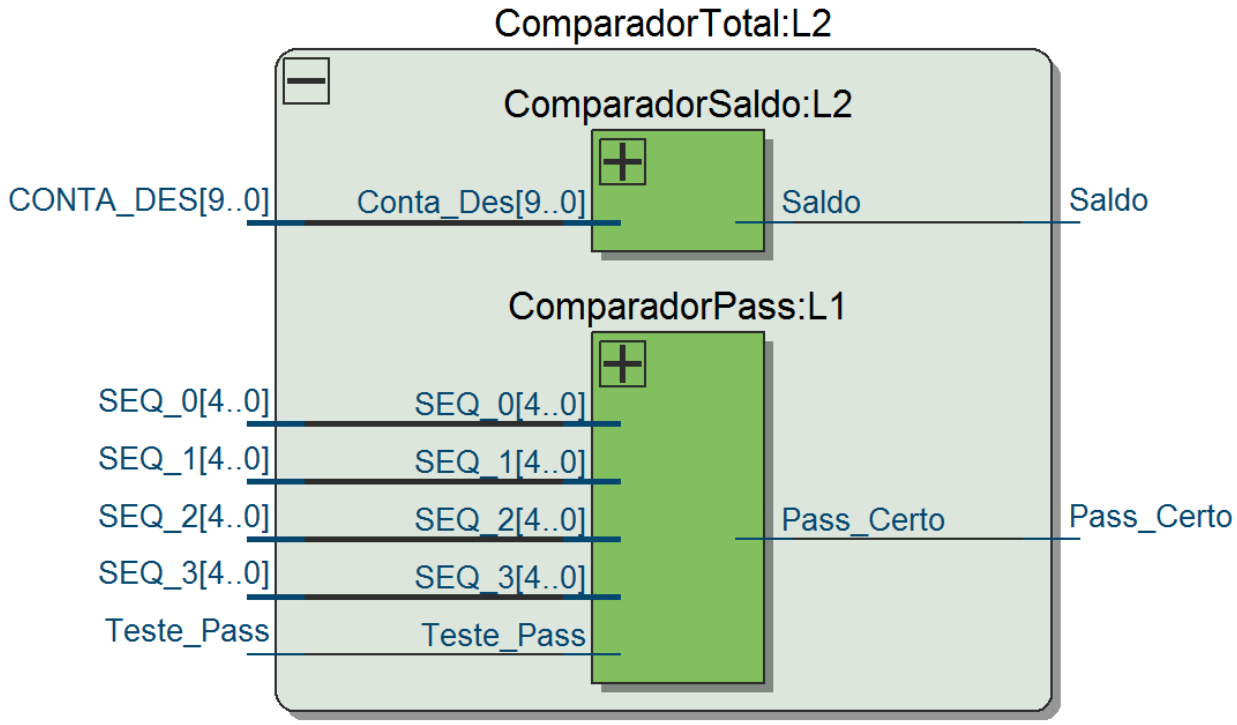
Para desenvolver a agenda foi necessário um arquivo VHDL, chamado ROM1. Nesse arquivo estão os 16 contatos. Que serão inseridos pelo usuário nos Swithes(3,2,1,0). Para obter os contatos foi necessário obter 32 letras ou números, que foram gerados na saída da FSM_ctrl C4, C3, C2, C1 e C0, que serão as entradas do enable dos registradores. Assim os registradores saberão quais são os 20 bits a serem liberados no SEQ_3, SEQ_2, SEQ_1 e SEQ_0. Cada um correspondendo a cinco bits, formando as quatro letras de cada contato. Assim quando a memória receber os quatro números correspondentes ao contado será liberado 20 bits para os displays HEX3, HEX2, HEX1 e HEX0.

1.3 Comparador

O bloco comparador está organizado em um bloco contendo três VHDLs, **ComparadorPass**, **ComparadorSaldo** e **ComparadorTotal**.

- **ComparadorPass:** arquivo foi utilizado uma abordagem comportamental, que receberá os 20 bits vindos dos registradores SEQ_3, SEQ_2, SEQ_1 e SEQ_0. É o responsável por comparar se os 20 bits do password estão corretos, e deverá liberar 0 se estiver incorreto e 1 se estiver correto. Depois essa saída passará por uma porta lógica AND que irá verificar se o valor da saída do verificador e da entrada TESTE_PASS, que vem da FSM_ctrl são iguais a 1 e liberam 1 lógico para a FSM_ctrl.
- **ComparadorSaldo:** nesse arquivo foi utilizado também uma abordagem comportamental, esse comparador recebe os 10 bits que irão vir do conta_descendente e deverá verificar se o saldo é igual à zero. Se for diferente libera 0 para a entrada da FSM_ctrl, senão libera 1, para que o sistema seja desligado.
- **ComparadorTotal:** esse comparador também possui uma abordagem comportamental, e é o aquivo responsável pela junção do ComparadorPass e do ComparadorSaldo.

O diagrama de blocos obtido a partir da descrição RTL para o Comparador Total segue na imagem:

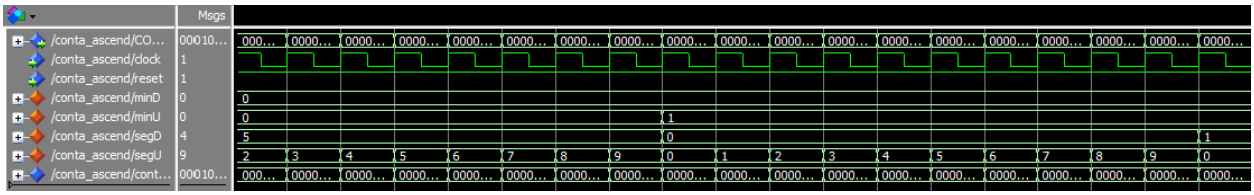


1.4 Contadores

Este bloco foi dividido em quatro VHDLs, chamados de **conta_ascendente**, **conta_descendente, FSM_clock** e **Contadores**.

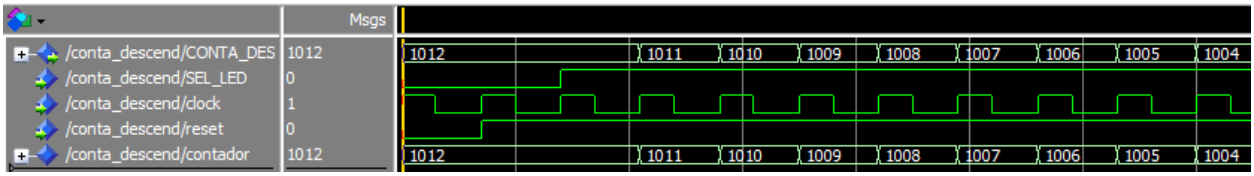
- **FSM_clock:** possui uma abordagem comportamental e é o responsável por gerar os dois clock: CLOCK1 e CLOCK2. O CLOCK1 possui uma frequência de 1Hz, estando ligado a entrada do arquivo conta_ascendente.Para gerar o CLOCK1 é necessário um contador para fazer uma contagem crescente até 50.000.000, que é o valor correspondente a 1 HZ. Ao chegar a esse valor é gerado um sinal de clock, que é ligado à entrada do arquivo conta_ascendente. Já no CLOCK2 acontece a mesma coisa só que o valor máximo do contador de 50.000.000 é dividido por 7, pois o valor da frequência desse clock é de 7 Hz, estando ligado a entrada do arquivo conta_descendente.
- **conta_ascendente**: esse VHDL também possui uma abordagem comportamental e é os responsável por gerar os minutos e os segundos de cada ligação nos displays de 7 segmentos.
- **conta_descendente**: esse arquivo de abordagem comportamental é o responsável por fazer uma contagem decrescente do valor do saldo. E a saída desse VHDL está ligada ao ComparadorSaldo. Assim a cada instante de tempo irá ser feita uma comparação para verificar se o saldo é igual à zero.
- **Contadores**: esse VHDL possui uma abordagem comportamental e possui a função de juntar os dois outros arquivos VHDL.

A simulação do **contador ascendente** segue abaixo:



A cada ciclo de clock (CLK1 de 1Hz), o contador aumenta em 1 segundo. O sinal do contador foi dividido em minD (dezena de minuto), minU (unidade de minuto), segD (dezena de segundo) e segU (unidade de segundo), representando o tempo decorrido de ligação no formato MM:SS. Na imagem é possível notar que quando o contador chega em “00:59”, o próximo valor é “01:00”, conforme esperado.

Segue abaixo a simulação do **contador descendente**:



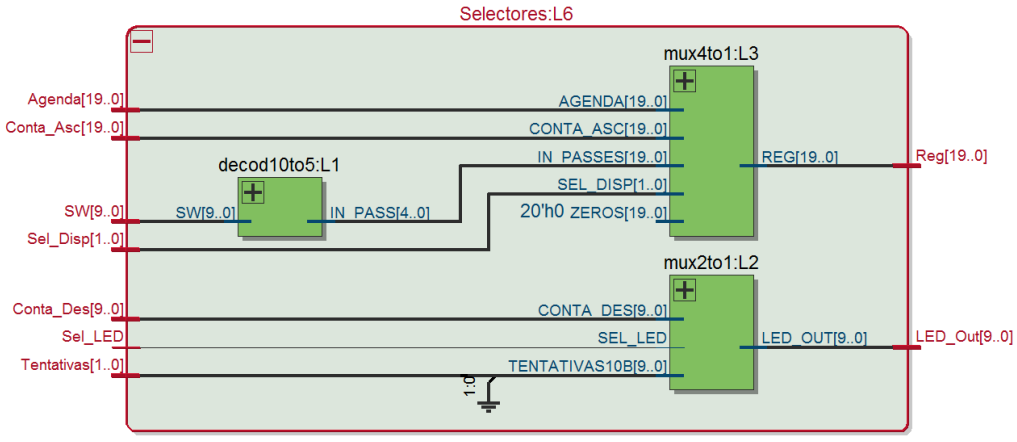
Pode-se notar que que o *contador* começa com o valor máximo de “saldo do celular” (1012), e quando o SEL_LED está ativo, ou seja, quando o equipamento está no estado E8 de Ligação, o contador vai diminuindo seu valor no clock de 7Hz (CLK2), até chegar a zero.

1.5 Selectores

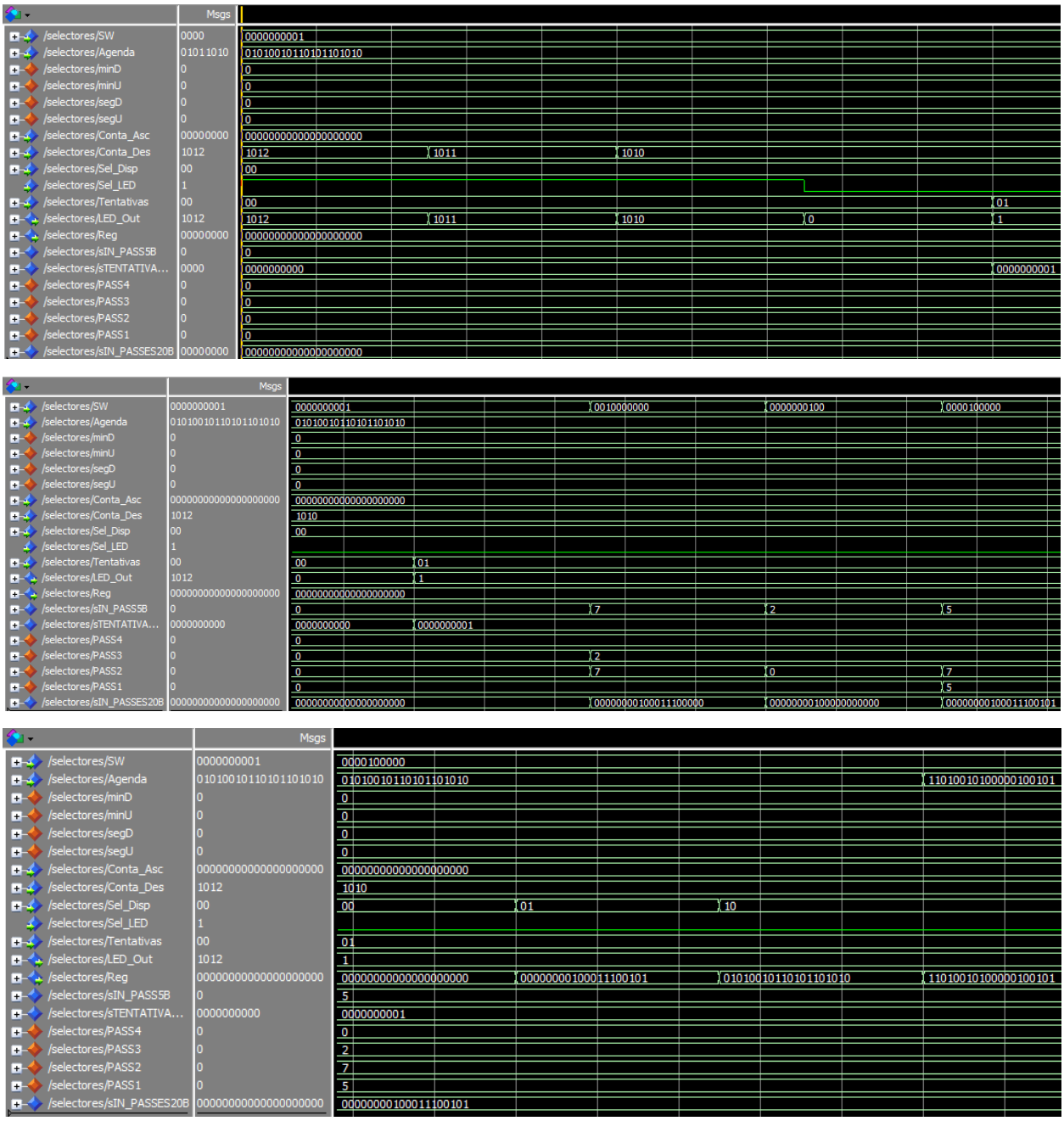
O bloco dos selectores conta com cinco VHDLs, que farão a seleção do password e deverão atuar para selecionar o que é esperado nos displays e nos LEDs. Os arquivos são: **mux2to1**, **mux4to1,** **decod7seg10to5** e **Selectores**.

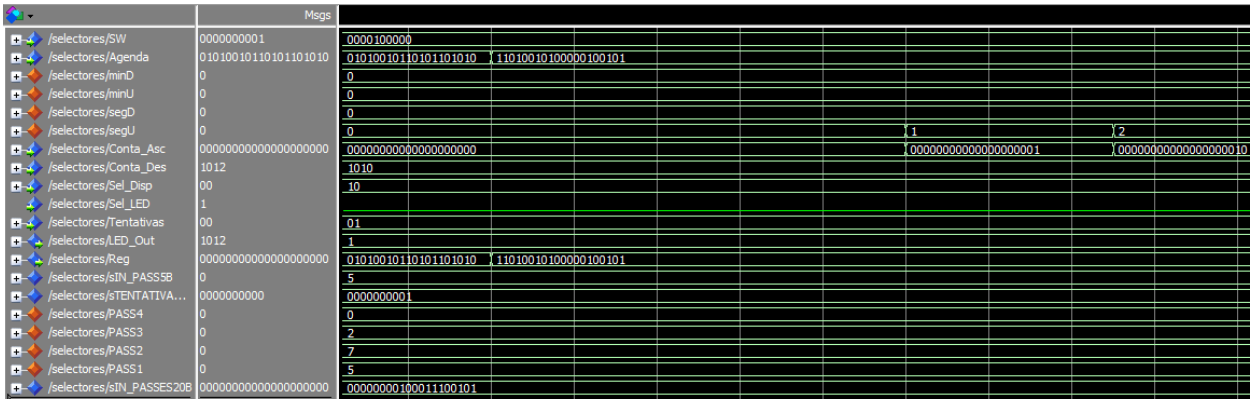
- **mux2to1:** possui uma abordagem comportamental e é o responsável pelo valor do saldo aparecer nos LEDS.
- **mux4to1:** também possui uma abordagem comportamental e é o responsável por selecionar os 20 bits para a entrada do registrador.
- **decod7seg10to5**: possui uma abordagem comportamental e é o responsável por transformar o sinal de 10 bits de entrada do password para os 5 bits necessários na entrada do decodificador de 7 segmentos.
- **Selectores:** esse arquivo possui uma abordagem comportamental e é o responsável pela junção entre os demais arquivos presentes neste bloco.

Segue na imagem abaixo o diagrama de blocos obtido para os **Selectores**.



A simulação dos **Selectores** no ModelSim apresentou os seguintes resultados:

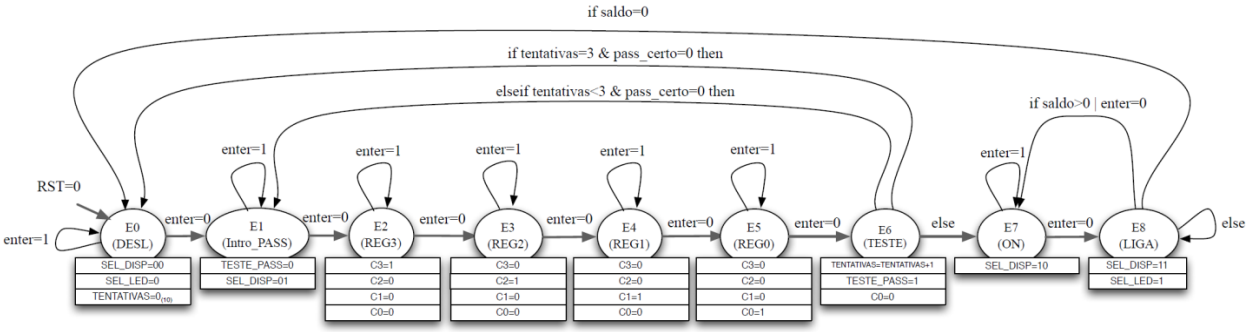




Nas imagem acima podemos ver que quando o signal *SEL_LED* é igual a 1, o valor apresentado nos LEDs (*LED_Out*) é o valor do *Conta_Des* (saldo do celular). Quando o SEL_LED é igual a 0, o valor apresentado nos LEDs será o número de *Tentativas* que foram feitas para acertar o *password*.

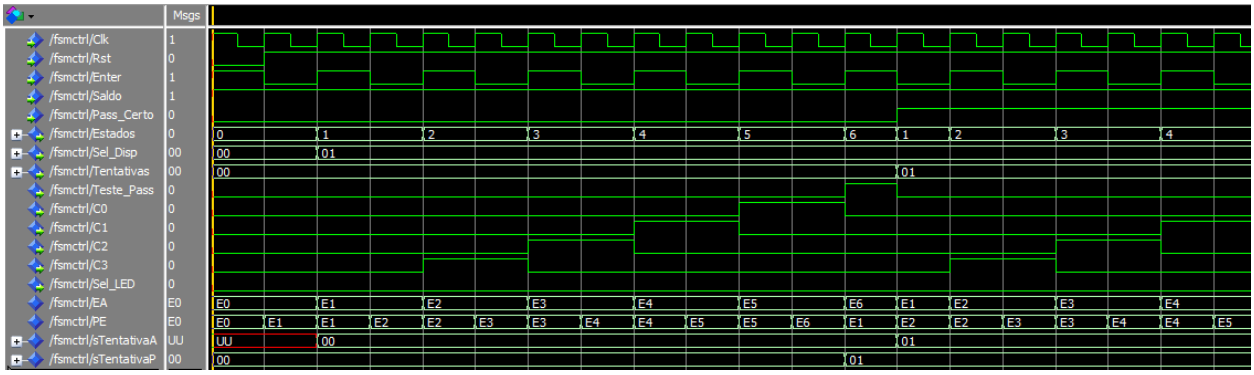
Da mesma maneira, quando o *SEL_DISP* é 00, o valor que irá para os Registradores é todo zerado. Quando *SEL_DISP* é 01, o valor que irá para os Registradores é o *password* que foi "digitado" através dos switches (SW) e gravados no sinal IN_PASSES20B, sendo 5 bits para cada digito da senha. Quando *SEL_DISP* é 10 o valor que vai para o Registrador é a saída da **Agenda**, ou seja, o nome do contato selecionado nos switches. E por fim, quando *SEL_DISP* é 11, o display irá mostrar o valor de **Conta_Asc**, ou seja, a duração da ligação.

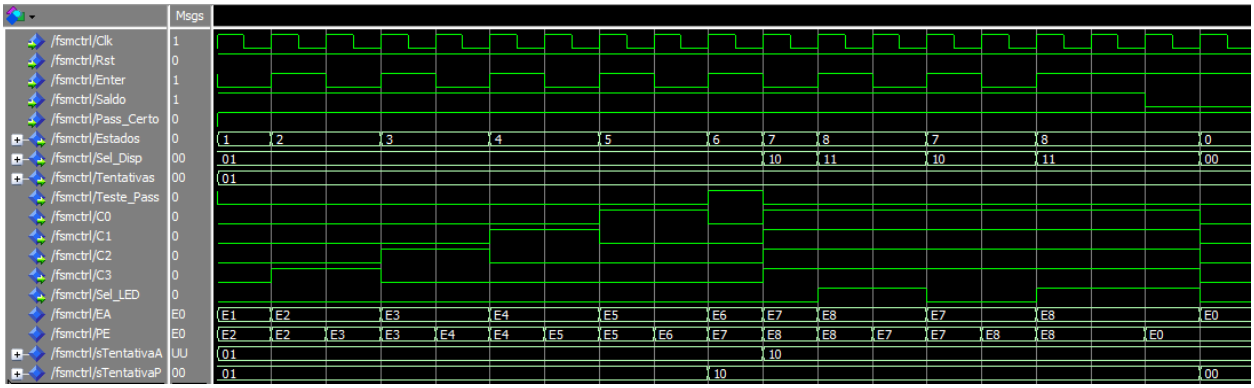
2 Controlador



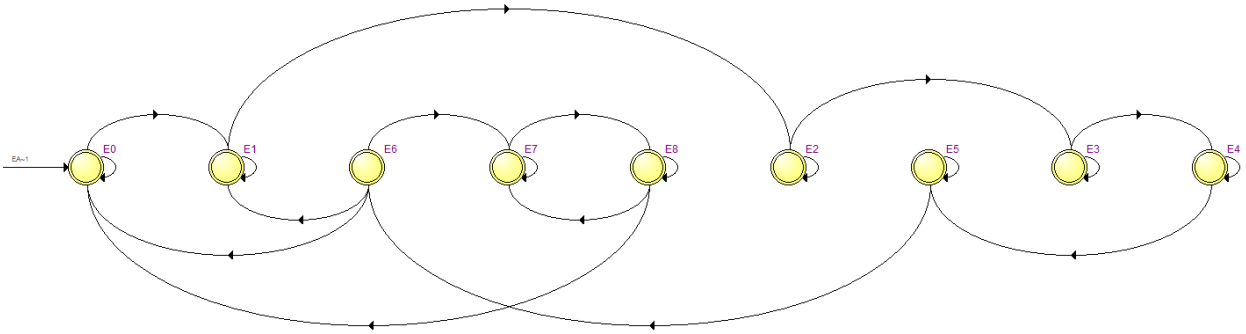
Esse diagrama de estados possui nove estados: E0, E1, E2, E3, E4, E5, E6, E7 e E8. O estado inicial é o E0, que é o estado inicial, quando ainda o sistema está desligado. Ao se apertar o botão “enter”, passa para o estado E1, onde o sistema está pronto para receber o password. Nos estados E2, E3, E4 e E5, o usuário deve inserir o password. No estado E6, o sistema irá verificar se o password inserido está correto. Se estiver incorreto o sistema voltará ao estado E1, e assim sucessivamente. Se o usuário já tentou inserir sua senha 4 vezes e errou em todas as tentativas, o sistema será desligado e voltará ao estado E0. Já se no estado E6 o usuário inseriu corretamente, o sistema passará ao estado E7. Nesse estado o usuário poderá fazer ligações a contatos que estão na memória ROM (citado anteriormente). Para acontecer uma ligação, basta o usuário inserir o número do contado e pressionar o botão “enter”, que irá começar uma ligação, passando para o estado E8, se desejar que a ligação termine é só pressionar novamente o botão “enter”, e voltará ao estado E7. O usuário poderá fazer isso o número de vezes que julgar necessário, mas assim que o saldo acabar o sistema é automaticamente desligado e volta para o estado E0.

Todo esse funcionamento foi obtido na simulação no ModelSim, conforme mostram as imagens abaixo.





Podemos ver também o mesmo diagrama de estados do nosso Controlador gerado pelo Quartus (State Machine Viewer), onde podemos conferir que todas as ligações entre os estados estão corretas.



3. Resultados e conclusões

No bloco dos registradores terão sete entradas: CLOCK_50, RST, C3, C2, C1, C0 e REG. O CLOCK_50 é um valor padrão para o uso do clock em FPGA utilizando o VHDL. O RST será representado pelo botão KEY0, que quando pressionado “reseta” o sistema, para que ele volte ao seu estado inicial. Já as entradas C3, C2, C1 e C0 são saídas do bloco controlador e serão os enable de cada um das quatro registradores presentes no bloco dos registradores. O REG é uma entrada de 20 bits que corresponde a saída do bloco dos selectores. O bloco dos registradores também possui três saídas: SEQ_3, SEQ_2, SEQ_1 e SEQ_0. Essas saídas serão as entradas de um dos comparadores e também entrada dos displays HEX3, HEX2, HEX1 e HEX0.

O bloco dos comparadores também possui outras quatro entradas: os 20 bits do password, TESTE_PASS, que é uma das saídas do controlador, CONTA_DESC, que é uma saída dos contadores, e ainda 10 bits no valor 0 para serem comparados com a entrada CONTA_DESC. Além de possuir duas saídas: SALDO e PASS_CERTO, que vão ser entradas para o controlador.

Além dessas duas entradas, o bloco controlador irá ter mais outras quatro: o CLOCK_50 e o RST, já citados anteriormente, o “enter”, que será o botão KEY3 da placa FPGA e ainda TENTATIVAS, uma entrada de 2 bits que representa o número de tentativas para inserir o password, essa entrada vem da saída deste mesmo bloco. Esse bloco possui outras oito saídas: C3, C2, C1, C0 e TESTE_PASS, já citados antes, ESTADOS, uma saída de 5 bits que entra no display HEX5, e também SEL_LED e SEL_DISP que serão entradas para o bloco dos selectores.

O bloco dos selectores possui outras quatro entradas: CONTA_DESC e CONTA_ASC, já citados anteriormente, os 20 bits vindos da AGENDA e 10 bits selecionados nos Switches. Com duas saídas REG, entrando nos registradores e LED_OUT para acender os LEDs. Esse bloco também possui algumas entradas e saídas internas (entre cada componente).

A agenda possui uma entrada, nos Switches(3 até 0), e uma saída de 20 bits, que será uma das entradas dos selectores.

O bloco dos contadores possui quatro entradas: RST, SEL_LED, SEL_DISP e CLOCK_50, e duas saídas: CONTA_DESC e CONTA_ASC todos já comentados anteriormente. Além de entradas e saídas entre componentes dos blocos.

A junção de todos os blocos foi bastante complicada, mas depois de muito trabalho e vários ajustes tudo funcionou tanto nas simulações quanto na placa FPGA.

Anexo A – Observações

Antes de começar a desenvolver o projeto, achamos que era só aplicar todos os métodos de descrição de hardware (VHDL) que nos foram ensinados durante a aula prática. Mas ao iniciar, percebemos que iria ser necessária muita concentração, planejamento e esforço para buscar as soluções e resolver todos os problemas que apareciam. Todas as dificuldades e problemas que enfrentamos durante a elaboração do código nos permitiram esclarecer muitas dúvidas que surgiram nesse caminho e não tínhamos percebido ao longo do semestre.

Ao longo da construção dos VHDLs, percebemos que tínhamos ainda muita dificuldade em escrever nessa linguagem de descrição de hardware. Mas aos poucos conseguimos entender cada vez mais e enfim terminamos o projeto, ou quase. Quando fomos fazer as simulações, o código funcionava perfeitamente até o estado E6, e depois não acontecia o que queríamos. Depois de muitos pensar em uma solução, foi possível finalizar o projeto com sucesso.

O projeto do funcionamento do telefone celular nos concedeu um maior aprendizado e com ele várias dúvidas foram esclarecidas.