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

Genilson Cristiano Braga dos Santos
Kauê Cano

TURMA 03202A

genilsoncbsantos@hotmail.com
kauecano@gmail.com

Relatório de Projeto Final EEL5105
2016.1

Florianópolis, 10 de Julho de 2015.

Conteúdo

1. Introdução

O projeto implementado é simplóriamente definido como um simulador interno de ligações telefônicas.

Primeiramente, após ligar o aparelho, o usuário estará perante um estado de verificação de senha. Neste caso, a senha correta será os quatro últimos componentes do número de matrícula do aluno Kauê Cano, 1142. Utilizando as *switches SW(9 downto 0)* para selecionar um dos componentes do *array* por vez (em ordem mais significativa à esquerda, de nove a zero) e o botão *BTN3* para confirmar sua seleção, o usuário enviará ordenadamente cada um dos quatro números da senha ao bloco de Selectores. Lá, a leitura de *10 bits* do *array* de *switches* será decodificada em números binários de *5 bits* que em seguida serã o concatenados ordenadamente e passarão por um MUX 4:1 para serem enviados ao bloco de Registradores.

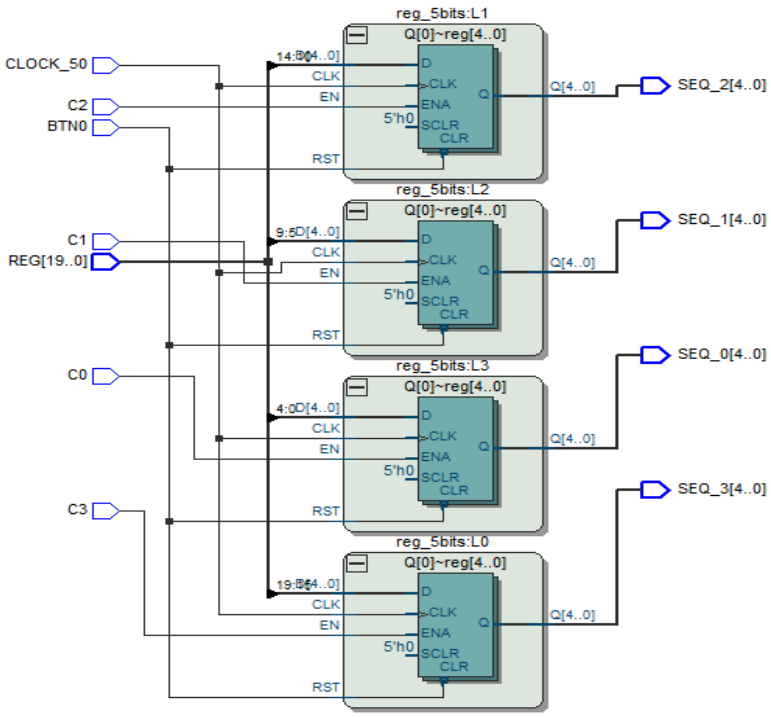
Neste bloco, os 20 bits concatenados são divididos entre os quatro registradores, os quais têm seu sinal de enable ordenada e encadeadamente controlado pelo bloco Controlador. As sequências de saída dos registradores são enviadas ao bloco Selector, para que assim os números possam aparecer no *Display* após passar por decodificadores de sete segmentos, e paralelamente ao Comparador, responsável por verificar se as sequências inseridas correspondem à correta e emitir um sinal de confirmação*PASS_CERTO*.

Após este processo de inserção de senha, o bloco Controlador chega ao estado número seis, no qual acontecerá a bifurcação condicional: caso a senha esteja errada e já tenham estourado quatro tentativas, voltará para o estado zero (desligado); caso a senha esteja errada e ainda tenha tentativas sobrando, retorna para o estado um (inserir nova senha) e, caso *PASS_CERTO* seja 1, irá avançar para o estado sete. Neste estado, a agenda (que possui 16 nomes predeterminados por nós) será acessada e o usuário poderá navegar pelos contatos selecionando um (novamente com o *BTN3*) para ligar. Assim que selecionado o contato, terá início a ligação, oitavo e último estado. Nele, o diplay mostrará a duração da ligação no formato MINUTOS:SEGUNDOS, enquanto o *array* de leds abaixo mostrará a depletação dos créditos.

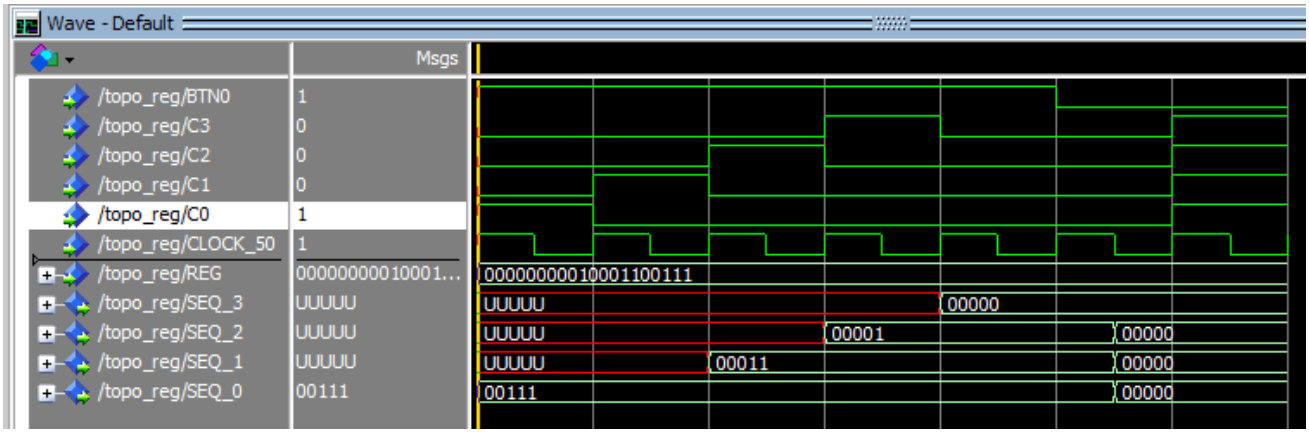
O bloco Contador é vital neste processo, já que pega o CLOCK_50 de 50 MHz e transforma no CLK1, de 1Hz, para utilização na contagem crescente do tempo, e no CLK2, de 4Hz para a contagem descrescente dos créditos. A ligação continuará até o esgotamento total dos créditos (verificado pelo bloco Comparador) , causando o desligamento do sistema, ou até o usuário apertar o *BTN3* novamente, o que retorna ele ao sétimo estado, de seleção.

Durante toda a execução do projeto, o display hexadecimal HEX5 fica setado em E, enquanto HEX4 demonstra o número do estado em questão. Há também a disponibilidade de executar um reset assíncrono no sistema ao apertar o BTN0.

1.1 Registradores

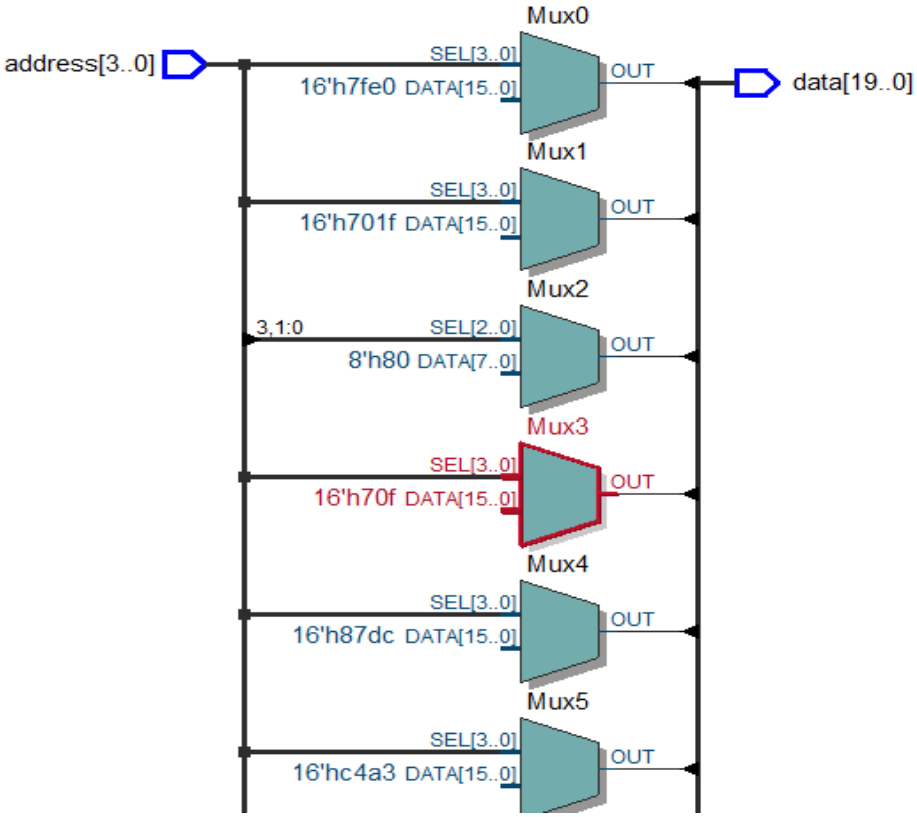


Pelo diagrama vemos a existência de quatro registradores semelhantes no bloco Registrador, diferenciados apenas pelo *enable* e *string* de *bits* que recebem. Cada um possui uma entrada clock e uma reset (BTN0) igual e funcionam da maneira convencional de um registrador: na passagem de uma borda de subida do clock, com enable sendo 0 e reset sendo 1, o componente pegará seu input D (uma partição de REG de 20 bits) e passará ao seu output Q (SEQ_X, sendo X o número individual do regitrador chamado no topo).



Forçamos a sequência “00000000010001100111” em REG, e habilitamos apenas o enable de C0 inicialmente, deixando BTN0 como 1 por hora. O Model Sim considerou o primeiro run já como um rising edge do clock, o que imediatamente fez SEQ_0 receber sua parcela de REG “00111”. Nos próximos ciclos, C0, C1, C2 e C3 receberam, respectivamente, 0,1,0,0; 0,0,1,0; 0,0,0,1, e 0,0,0,0 o que fez que cada um dos SEQs recebesse sua parcela de REG ordenadamente, com o “atraso” do registrador em rising edge do clock. Em seguida, BTN0 foi acionado em 0, o que negou, corretamente todas as saídas, mesmo, como visto posteriormente, com C0,C1,C2 e C3 indo a 1.

1.2 Agenda

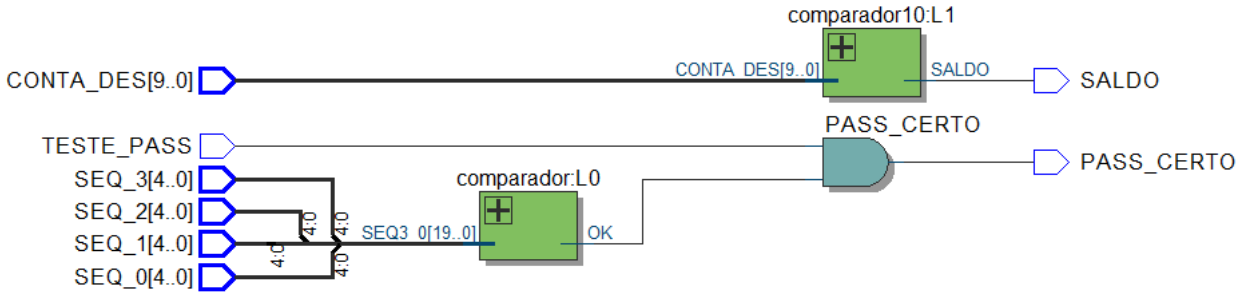


O funcionamento da agenda é bem representado pelo RTL acima. O bloco ROM recebe um address de 4 bits, uma requisição de visualização do que está naquele endereço. Após achado o addres em questão neste emparelhamento de dados, a informação data armazenada sob àquele código será permitida para sair como output.

Neste caso, data está codificada em concatenações de quatro letras de cinco bits (formato posteriormente decodificado para o display de sete segmentos), sendo que o componente teve 16 nomes manualmente inseridos por nós, podendo ser selecionados pelos seletores.

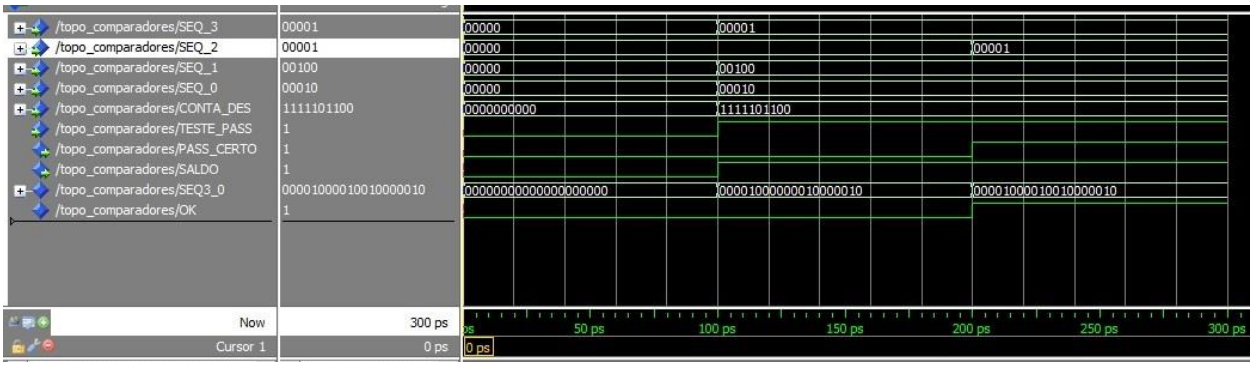
.

1.3 Comparador



Em nosso código, utilizamos a abordagem comportamental. Basicamente, colocamos qual a sequência tido como “correta” em cada um dos comparadores e, a partir daí, somente comparamos com o sinal de entrada para assim se definir as duas saídas do bloco Comparador.

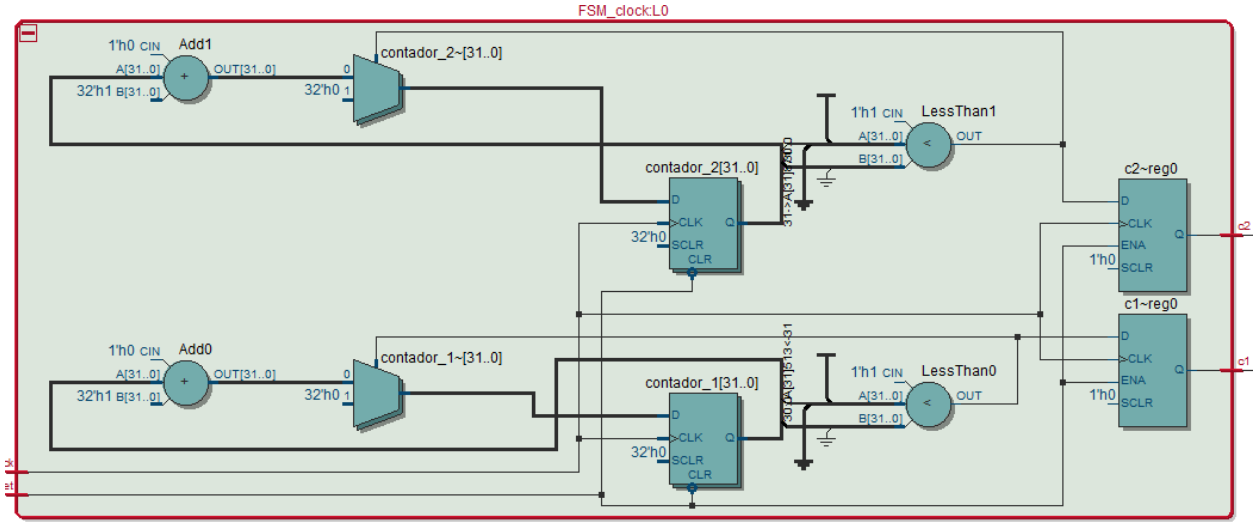
No comparador de senha, também foi utilizado um and com TESTE_PASS, enviado do bloco Controlador para que o teste seja feito no momento certo.

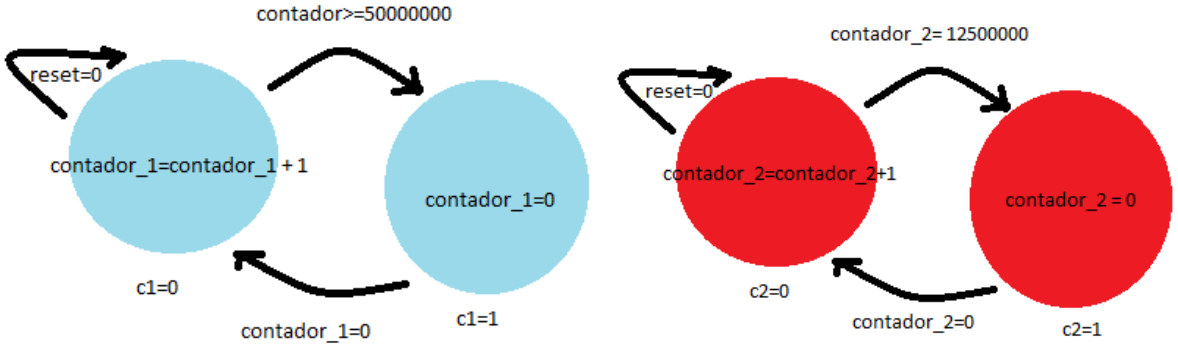


Na simulação dos comparadores, colocamos inicalmente os valores de SEQ3 à SEQ0 de 5 bits em “00000” e TESTE_PASS igual a ‘0’. Então o signal SEQ3_0 de 20 bits recebe todas as entradas e o signal OK mostra ‘0’ indicando que a senha está incorreta e a saída PASS_CERTO mostra ‘0’. Somente quando colocamos a senha correta e TESTE_PASS = ‘1’ a saída OK mostra ‘1’ PASS_CERTO mostra ‘1’.

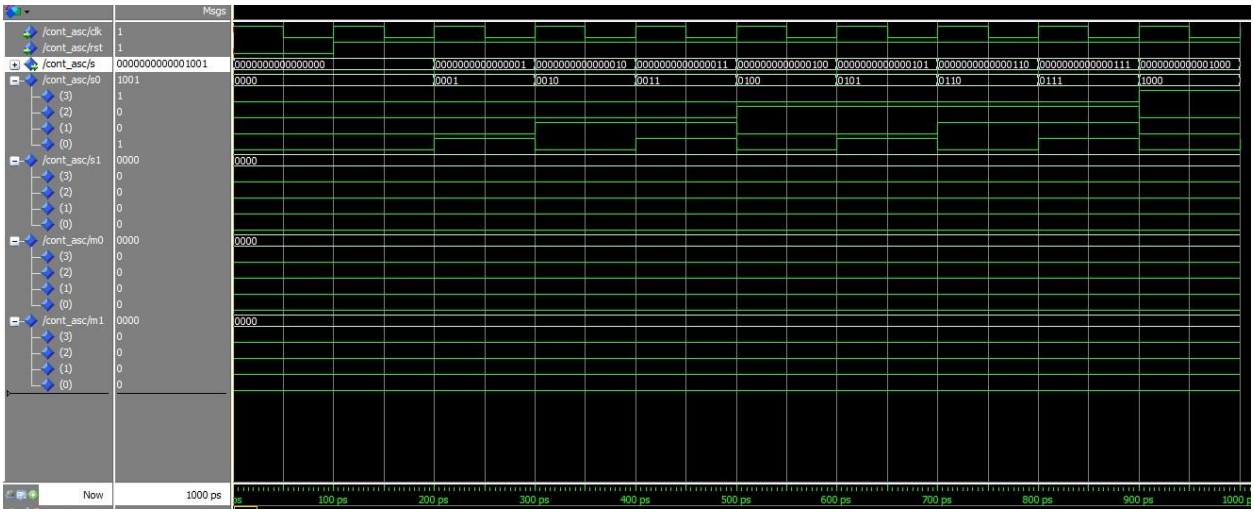
A entrada CONTA_DES de 10 bits foi inicialmente colocada em “000000000” então a saída SALDO mostra ‘0’, em seguida colocamos o valor inicial dos créditos “1111101100” na entrada CONTA_DES, então a saída SALDO mostra’1’.

1.4 Contadores





Com a utilização de dois registradores, foram criadas duas variáveis inteiras: um número limite para cada uma das contagens. Nestas, o novo clock fica congelado no estado 0 com a variável inteira incrementando de clock_50 em clock_50 (de 50 MHz), até atingir o denominador desejado para a nova frequência (respectivamente 50M e 12,5 M, em 1Hz e 4Hz para ClK1 e CLK 2), para que o novo clock seja setado para 1, a variável inteira setada para 0 e, por fim, o ciclo possa se repetir.



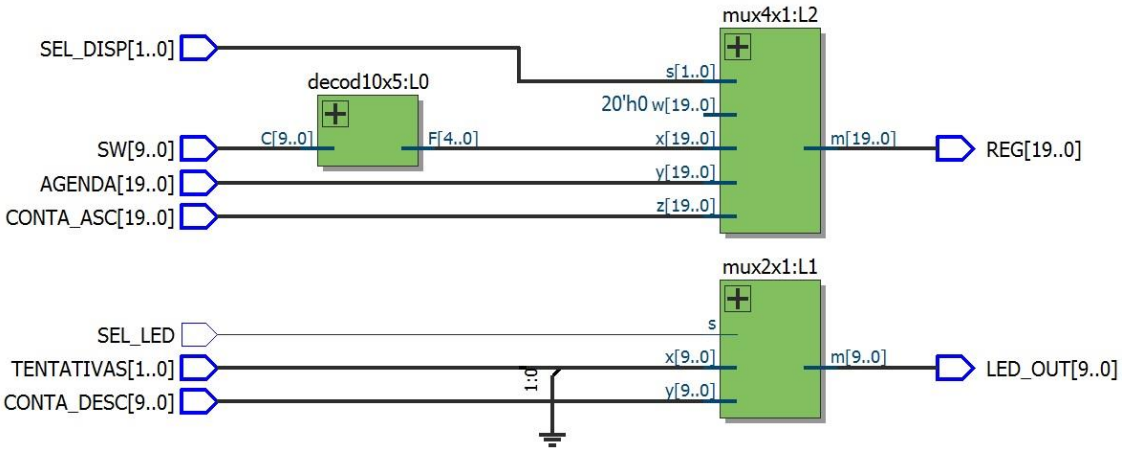
1.5 Selectores

A partir dos conhecimentos obtidos na aula de laboratório número 5, inclusive adotando o modelo explicado em aula, desenvolvemos um decodificador de binário para 7 segmentos que vai receber as entradas de 5 bits “00000” até “11100” ou seja 29 caracteres poderão ser escritos em cada display. Na ordem crescente cada entrada vai corresponder a uma saída de 7 bits que determinará um caracter. No nosso caso cada segmento acende com zero lógico.

O decodificador de 10 para 5 receberá uma entrada de 10 bits que virá dos switches da placa DE1-SoC que será usada para testes e simulação e fornecerá uma saída de 5 bits. Esse decodificador será responsável por receber a entrada correspondente a senha. Portanto cada SW de 0 a 9 em 1, enquanto as outras estão em zero, corresponderá uma saída binária correspondente no decodificador de 10x5 transformado os valores de 0 a 9 decimal em “00000” a “01001” binário. O sinal de saída do decoder 10x5 foi concatenado quatro vezes em um sinal S1 que é uma das entradas do multiplexador 4x1 e será mostrado nos displays de sete segmentos após passar pelo bloco de registradores e pelos dedodificadores de 7 segmentos, sempre que a entrada de seleção do Mux4x1, que vem da saída SEL_DISP da máquina de controle, estiver em “01”.

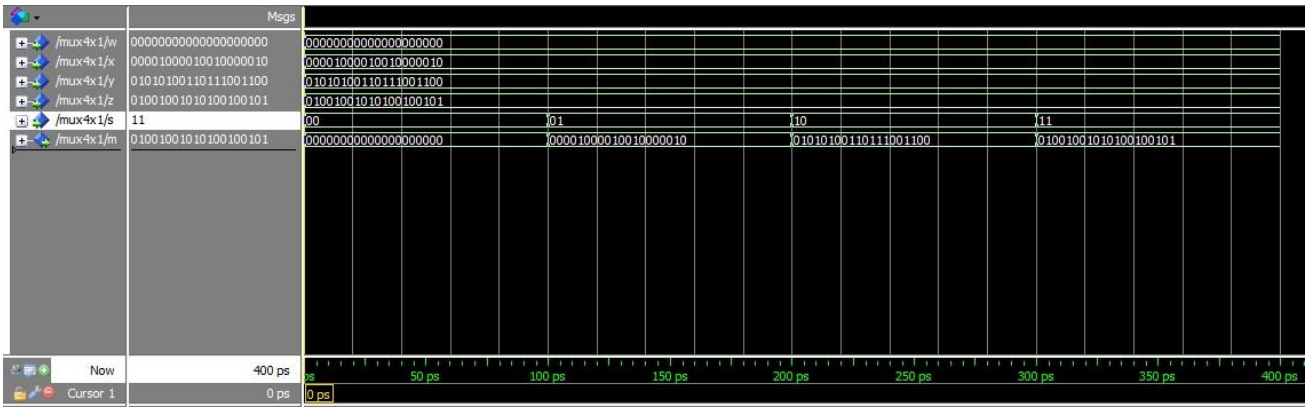
De acordo com o que aprendemos na aula de laboratório número 4, onde aprendemos projetar o multiplexador de 2 x 1 e o de 4 x 1 de forma estrutural, onde definíamos exatamente qual porta lógica deveria ser usada, e de forma comportamental, onde definíamos apenas qual entrada deveria ser mostrada na saída a partir de uma entrada seletora, entendemos que seria mais viável e menos

trabalhoso construir os nossos multiplexadores de forma comportamental, apesar de não termos o controle de quais e quantas portas lógicas seriam criadas pelo Quartus. No topo criado para os selectores concatenamos a saída “TENTATIVAS” de 2 bits, da máquina de controle com oito zeros e colocamos na entrada do multiplexador 2 x 1.



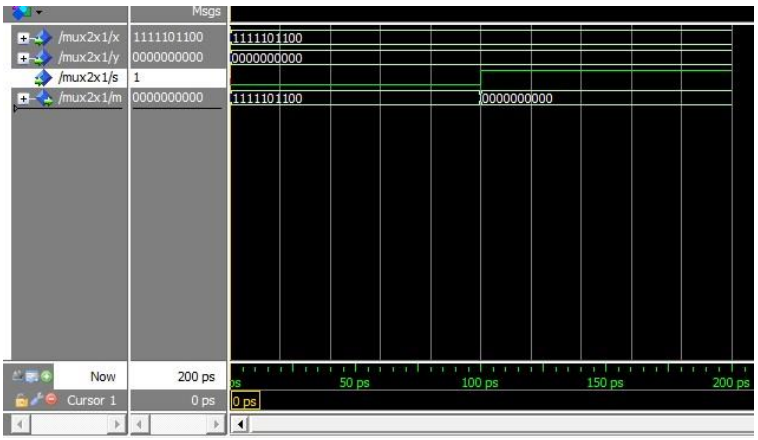
No diagrama de blocos dos Seletores vemos as entradas SEL_DISP e SEL_LED que vão definir qual entrada nos multiplexadores vai aparecer na saída.

MULTIPLEXADOR 4 x 1:



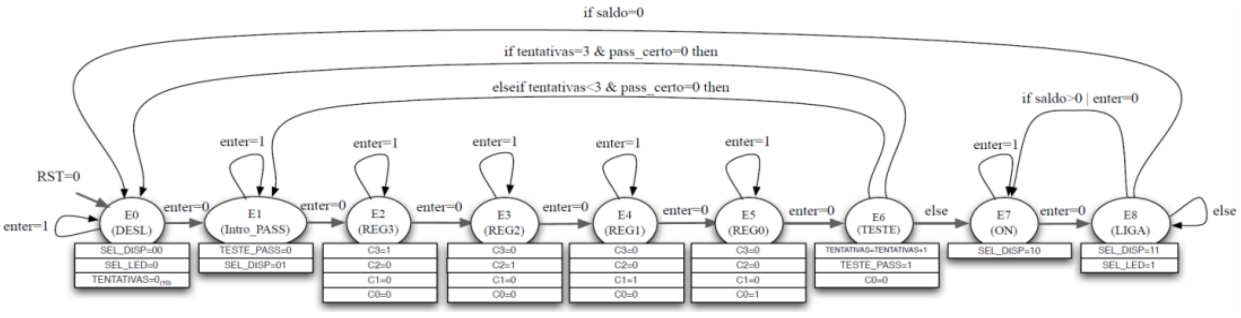
Na simulação do Mux 4x1, fixamos os valores das entradas “w, x, y e z” de 20 bits e variamos a entrada seletora “s” de 2 bits. Vericamos que para cada valor de “s” a saída “m” também de 20 bits, correspondia à uma das entradas, conforme mostrado na simulação.

MULTIPLEXADOR 2 x 1:



Na simulação do Mux 2x1, fixamos os valores das entradas “x e Y” de 10 bits e variamos a entrada seletora “s”. Verificamos que para cada valor de “s” a saída “m” também de 10 bits, correspondia à uma das entradas, conforme mostrado na simulação.

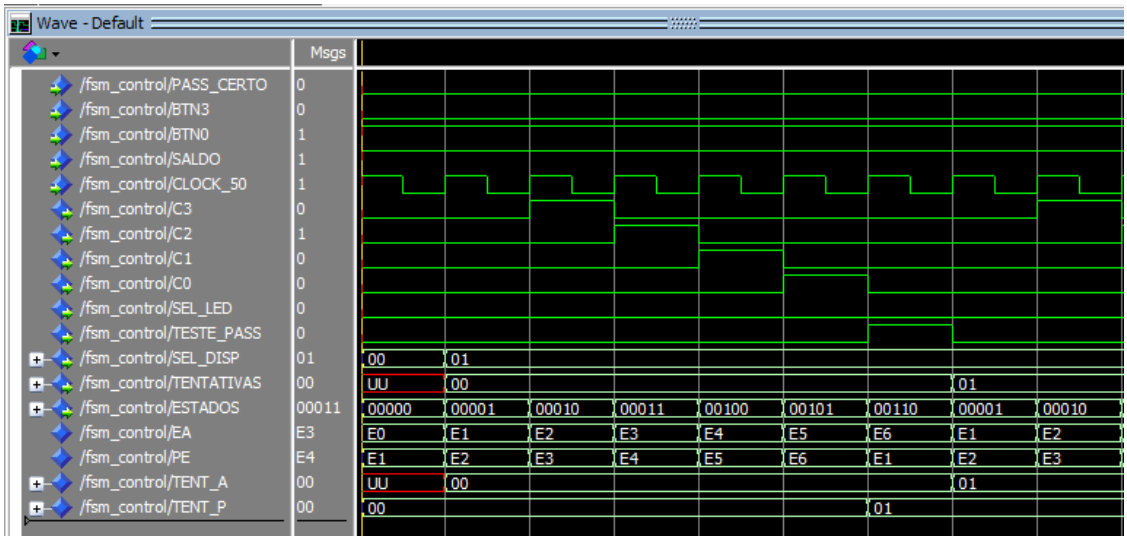
2 Controlador



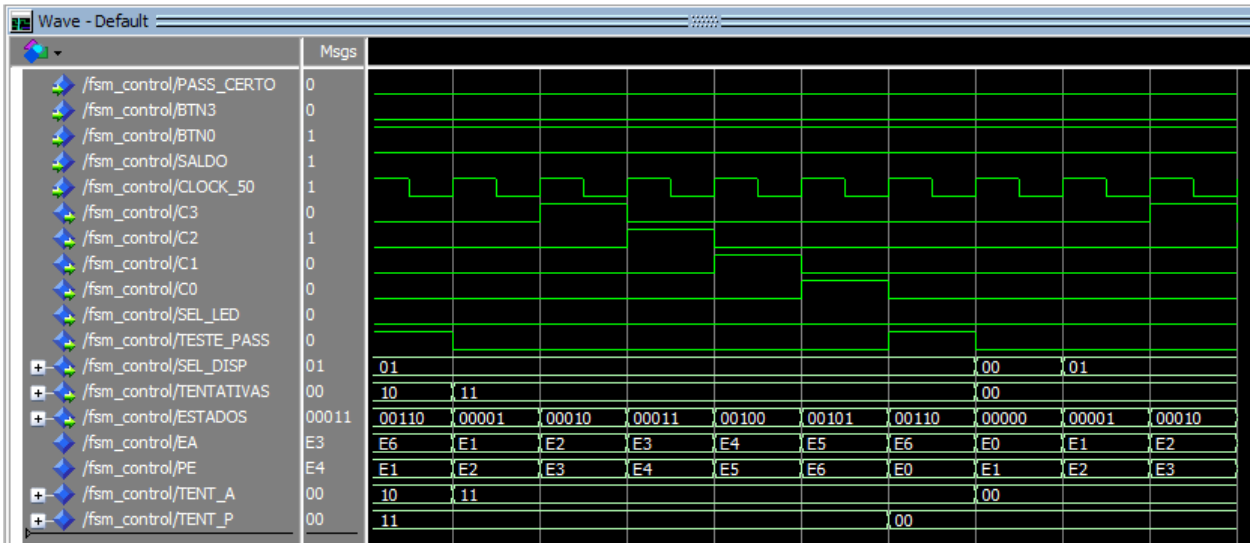
Cronológicamente: o sistema fica desligado em E0, incialmente com todas as variáveis de controle em 0, até que enter (BTN3) seja acionado. Em E1, o display, além de indicar o estado atual(como em todos os outros estados), terá SEL_DISP como 01 (mostrará nos próximos estados a senha que o usuário está inserindo) e TESTE_PASS como 0 (para que a senha só seja testada em estado futuro. Assim, em E2, E3, E4 e E5 (avançando pelo botão enter), o celular receberá cada um dos quatro

dígitos inseridos pelo usuário, sendo que cada estado habilita um registrador por vez por meio de C0, C1, C2 e C4. Chegando em E6, ocorrerá o teste da senha (TESTE_PASS como 1), o número de tentativas será incrementado e C0 voltará a zero para desabilitar o último registrador usado.

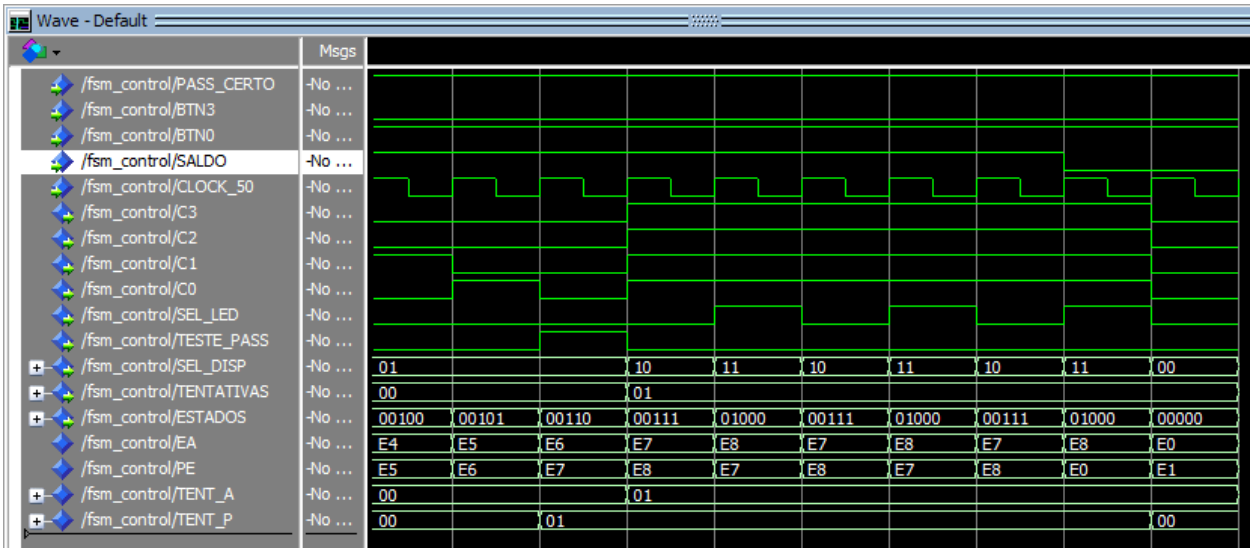
Caso a senha inserida esteja errada (PASS_CERTO=0) e as tentativas forem inferiores a quatro, volta-se ao estado E1 para reiniciar a captação de senha; caso as tentativas forem iguais a quatro, o sistema desliga e as variáveis são reiniciadas (E0); em outros casos, ao apertar enter, o sistema irá para E7. Em E7 SEL_DISP irá para 01, habilitanto o display a mostrar os contatos da agenda. Selecionado o contato, o sistema irá para E8 (a ligação em si), com SEL_DISP em 11(contagem crescente da duração da ligação) e SEL_LED em 1 (contagem descrescente dos saltos). O sistema pode voltar a E7 caso enter seja acionado novamente ou finalizar em E0 quando o saldo atingir 0.



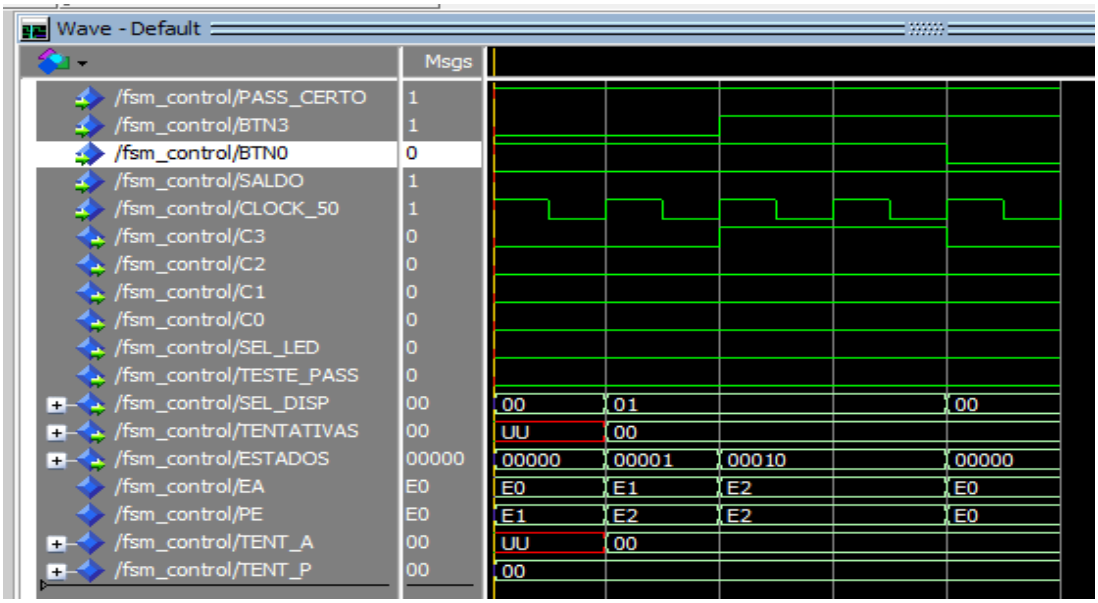
Primeiramente, com BTN0 em 1 (reset não acionado), PASS_CERTO em 0 (senha errada), BTN3 em 0 (enter pressionado em cada ciclo de clock), SALDO em 1 (possui saldo), verifica-se que a FSM passa de estado e, ao chegar em E6 com senha errada, volta a E1 e incrementa TENTATIVAS.



E assim, ao chegar novamente em E6 com PASS_CERTO em 0 pela quarta vez, o sistema é desligado (E0), com TENTATIVAS de volta a 00 e ESTADOS a 00000.

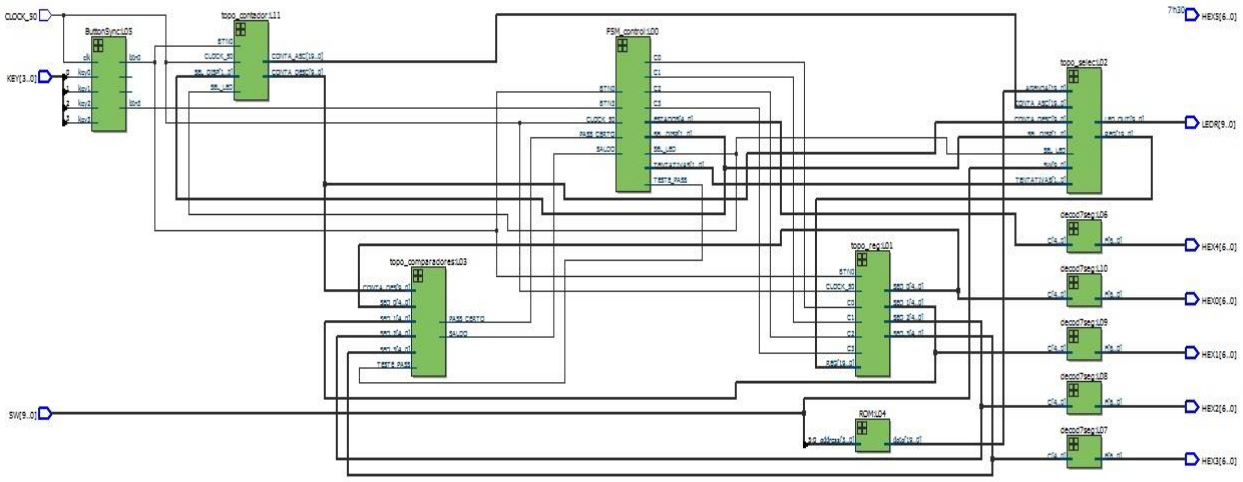


Por outro lado, quando o sistema chega em E6 com PASS_CERTO em 1 (senha correta), ele avança a E7 (seleção de ligação). Com BTN3 pressionado, avança para E8, onde fica até que o botão seja pressionado novamente (voltando para E7) ou até os créditos acabarem (voltando para E0 e zerando TENTATIVAS e ESTADOS).



Aqui oberva-se que o sistema fica estacionário quando BTN3 não é pressionado e é resetado de fato com BTN0 pressionado.

3. Resultados e conclusões



Por fim, após compilar com sucesso cada um dos blocos, teve início a união deles e formação do topo geral. Esta integração foi turbulenta, com diversos erros diferentes de compilação, mas notadamente localizados no bloco seletor (display não se comportando como desejado), contador (segundos demorando o dobro para passar e, como visto no Anexo A, lógica errônea de contagem decrescente) e controlador (diversos erros menores de sinais mal interpretados e falta de registradores para incrementar TENTATIVAS).

Após ajuda de professores e monitores da disciplina, simulações no Model Sim e testes na placa, nos livramos dos erros e tudo rodou corretamente no FPGA.

Este foi um projeto extenso, especialmente para quem não é fluente em VHDL. Porém, o Google e os monitores (especialmente o Rafael) ajudaram muito a superação dos erros, melhor domínio da lógica da linguagem e compreensão mais tactível de uma matéria que, apesar de já ser metade prática, foi muito melhor absorida (por nós, pelo menos) na hora de realizar pelas próprias mãos.

Anexo A – Observações

Por fim, uma crítica construtiva ao projeto. No contador descendente, seria interessante implementar uma lógica mais tactível e intuitiva ao apagamento dos leds: o gradual apagamento dos 10 leds proporcional ao saldo restante similar a uma "barra" de bateria encontrada nos celulares atuais. Inicialmente, achamos que esta era a intenção e escrevemos o snippet de código usando elsifs para isso.

```
begin
    process ( clk, rst)
        begin
            if rst = '0' then
                S <= "1111111111";
            elsif (CLK1'event and CLK1 = '1') then
                conta<=contp;
            if en = '1' then
            contp=conta + '1';
                                    --103                           207
                if conta > "0001100111" and conta < "0011001111" then
                S<="1111111110";

                                    --207                           307
                elsif conta > "0011001111" and conta < "0100110011" then
                S<="1111111100";

                                    --307                           407
                elsif conta > "0100110011" and conta < "0110010111" then
                S<="1111111000";

                                    --407                           507
                elsif conta > "0110010111" and conta < "0111111011" then
                S<="1111110000";
                    --507                          607
elsif conta > "0111111011" and conta < "1001011111" then
S<="1111100000";

                    --607                                  707
elsif conta > "1001011111" and conta < "1011000011" then
S<="1111000000";

                    --707                                  807
elsif conta > "1011000011" and conta < "1100100111" then
S<="1110000000";

                    --807                                  907
elsif conta > "1100100111" and conta < "1110001011" then
S<="1100000000";

                    --907                                  1007
elsif conta > "1110001011" and conta < "1111101111" then
S<="1000000000";

elsif conta > " 1111101111" then

S<="0000000000";
```