

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

Pedro Machado Santos Rohde - 16101085

TURMA 02220B

<pedrorohde@hotmail.com>

**Relatório de Projeto Final EEL5105
2016.2**

Florianópolis, 29 de novembro de 2016.

Conteúdo

1. Introdução.....	3
1.1 Predefinição	4
1.2 Comparador	5
1.3 Contador	6
1.4 Seletores	8
1.5 Registradores	10
2. Controlador	12
3. Resultados e conclusões	14
Anexo A – Observações.....	15

1. Introdução

Neste projeto da disciplina de Circuitos e Técnicas Digitais, aliando os conhecimentos adquiridos nas aulas teóricas e as técnicas da aula prática, foi elaborado um micro-ondas simplificado (sem a parte do aquecimento propriamente dito), a ser implantado na placa de FPGA DE1 da Altera. A construção do projeto se deu na linguagem de descrição de *hardware* VHDL, com o auxílio dos programas Quartus para compilação e ModelSim para simulação.

O micro-ondas criado utiliza 4 botões (*reset*, *enter*, *predef* e *pause*), 10 *switches* que possibilitam a entrada de valores numéricos binários, 10 *LEDs* e 6 *displays* de sete segmentos capazes de representar números e letras. Todos esses componentes estão disponíveis na placa e o projeto se constituiu em criar a lógica para associá-los de forma a fazer com que se comportem da forma desejada, isto é, como um micro-ondas. Além disso, fez-se uso do *clock* interno da placa *CLOCK_50*, de 50 MHz.

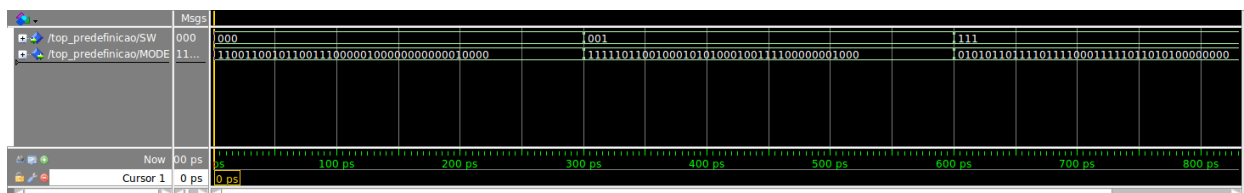
Essa lógica foi dividida em sete partes, os blocos. São eles: **controle**, máquina de estados que cujas saídas são usadas para controlar os outros blocos; **contadores**, que define os *clocks* *CLK1* (1 Hz) e *CLK2* (10 Hz) que serão usados por outros blocos, além de fazer a contagem regressiva de tempo; **registradores**, que servem para armazenar dados; **seletores**, utilizados para selecionar entre um número de entradas qual delas deve ser a saída; **comparador**, que compara valores; **predefinição**, onde tem-se a memória dos modos de aquecimento predefinido; e **debouncer**, ferramenta de sincronismo dos botões.

Para formar cada bloco são utilizadas entidades como multiplexadores (bloco de seletores), registradores (bloco registradores) etc., sendo estes os elementos mais abaixo na hierarquia do projeto. Os principais blocos e suas especificidades serão discutidos mais adiante.

Acima de todos esses blocos e no topo da hierarquia do projeto, está a entidade principal *topo_microondas*, responsável por fazer a ligação entre cada um dos blocos. De maneira simplificada, o que ela faz é: ligar as saídas de *debouncer* com as entradas de blocos que fazem uso dos botões; ligar as saídas de *clocks* de contadores com as entradas de blocos que se utilizam dos mesmos; ligar as saídas de controle com as entradas de blocos que dependem do estado atual da máquina de estados para definir sua funcionalidade; ligar as saídas de contador com as entradas de comparador e as saídas deste com controlador, para que a máquina de estados possa saber quando a contagem chegou a zero e defina a como devem operar os outros blocos; ligar as entradas de seletores às saídas dos blocos que definem diferentes valores para os *LEDs* e *displays*; e ligar as saídas de seletores aos próprios *LEDs* e *displays* da placa. Demais definições de funcionalidades e ligações específicas dos blocos serão apresentadas mais adiante.

1.1 Predefinição

No bloco Predefinição tem-se uma memória cujas entradas são os *switches* de 0 a 2 e cuja saída é o barramento interno MODE, de 40 bits (20 para os quatro caracteres do modo de aquecimento, 10 para o tempo, sendo 4 para os minutos e 6 para os segundos, e mais 10 para a potência). A entrada fornece um endereço da memória (linha 24 em diante em *ROM_pre.vhd*). A saída é determinada pelo conteúdo apontado pelo endereço (linhas 13 a 20 em *ROM_pre.vhd*).



Simulação de top_predefinicao. A entrada SW determina a saída MODE como definido na memória ROM_pre.

Os modos de aquecimento predefinido escolhidos foram:

<i>Switches</i>	<i>Modo</i>	<i>Display</i>	<i>Tempo</i>	<i>Potência</i>
000	Pipoca	“PIPO”	02:00	5
001	Chá	“CHA”	02:30	4
010	Legumes	“LEgU”	06:00	7
011	Sopa	“SOPA”	05:30	6
100	Miojo	“MIOJ”	04:10	3
101	Bolo	“bOLO”	03:15	10
110	Pudim	“PUdI”	10:00	8
111	Arroz	“ArrO”	15:45	9

1.2 Comparador

Para o comparador, foi utilizada uma abordagem estrutural, por conta de sua relativa simplicidade (não são necessários *process* ou *if/else*). Em *comparador*, a comparação de cada bit das duas entradas é feita com uma porta lógica *XNOR*, cuja tabela verdade mostra que sua saída terá nível lógico alto apenas quando as duas entradas forem iguais. Se isso ocorrer entre cada um dos 10 bits de cada entrada, tem-se que a saída EQ será 1, por conta das portas *AND*.

Em *top_comparador*, a entrada A do comparador está definida como “0000000000”, para que a saída *READY* seja 1 quando a entrada *CONTA* chegar em zero.

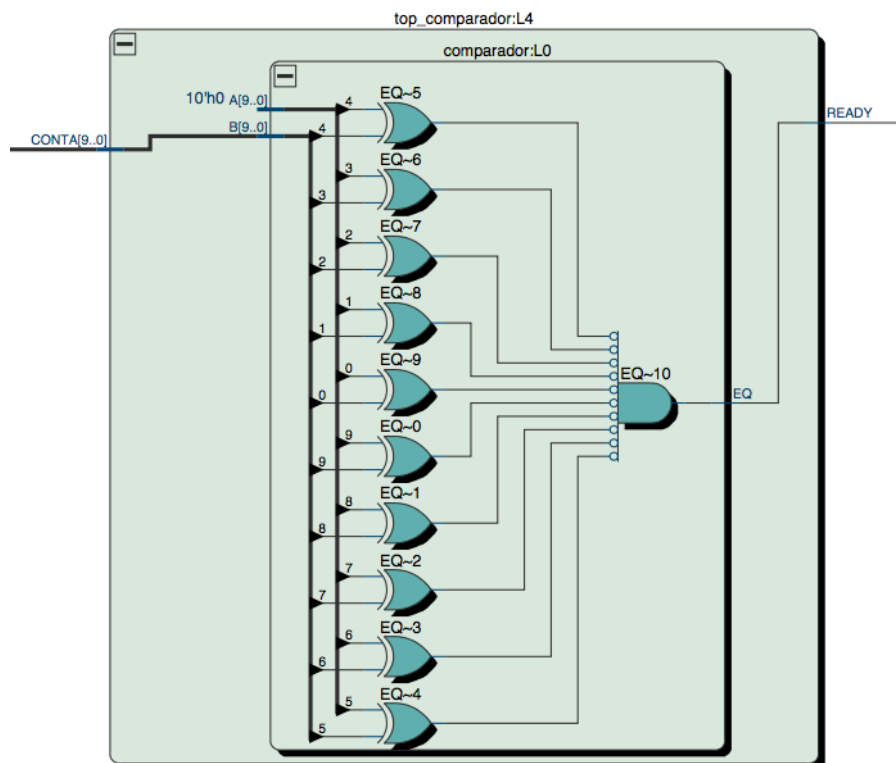





Diagrama de blocos de top_comparador. Tem-se a entrada A de comparador como “0000000000” e B como CONTA, as portas XOR, cujas saídas são invertidas (tornando-se XNOR) na entrada da porta AND e, finalmente, sua saída EQ, que sai de top_comparador como READY.

		Msgs																																												
 /comparador /A  /comparador /B  /comparador /EQ	-No Data-	<table><tr><td>0000000000</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>1111111111</td><td>0000000001</td><td>0000000000</td><td></td><td>1000000000</td><td>1010101010</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>										0000000000												1111111111	0000000001	0000000000		1000000000	1010101010																	
	0000000000																																													
	1111111111	0000000001	0000000000		1000000000	1010101010																																								
-No Data-																																														
-No Data-																																														

Simulação de comparador. Observa-se que EQ é 1 apenas quando A e B são iguais.

1.3 Contador

Em *FSM_clock*, são definidos os *clocks* *CLK1*, de 1 Hz, e *CLK2*, de 10 Hz. O *CLOCK_50* da placa tem 50 MHz, ou seja, período de 20 ns.

Para *CLK2* foi utilizado um contador (sinal *contador10hz* em *FSM_clock*), incrementado a cada transição positiva de *CLOCK_50*, ou seja, a cada 20 ns. Quando a contagem de *contador10hz* chega a 4.999.999 (4C4B3F em hexadecimal), ou seja, a cada 0,1 segundo, *CLK2* passa a nível lógico alto, a contagem é reiniciada e *CLK2* volta a nível lógico baixo logo em seguida.

Como o período de *CLK1* é dez vezes o de *CLK2*, o clock de 1 Hz foi definido a partir do de 10 Hz da seguinte maneira: um contador (*contador1hz* em *FSM_clock*) é incrementado a cada transição positiva de *CLK2*, que ocorre a cada 0,1 segundo. Assim que este contador, inicializado em zero, chega a 10, ou seja, imediatamente após o décimo ciclo de *CLK2* (equivalente a 1 segundo), *CLK1* passa a nível lógico alto, sua contagem é reiniciada e *CLK1* volta a nível lógico baixo logo em seguida.

Desta forma ficam definidos os clocks de 1 e 10 Hz que serão utilizados pelos outros blocos do projeto.

Em *CONT_DESC*, tem-se a contagem regressiva que servirá de *timer* para o micro-ondas. Como o tempo usa apenas minutos e segundos, *CLK1* de 1 Hz foi usado para essa contagem.

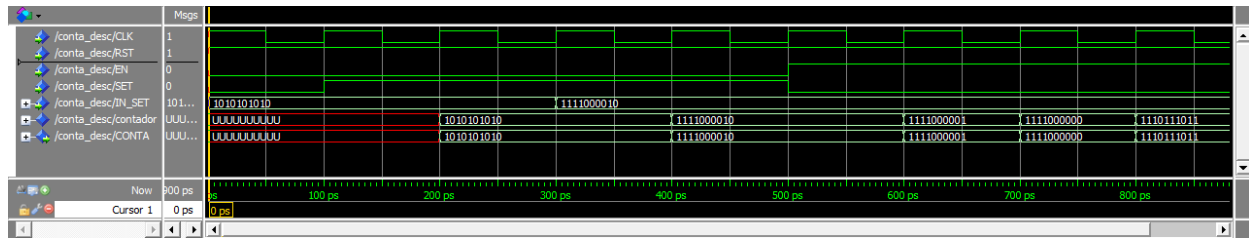
O sinal *RST_CONT* em nível lógico baixo, ou seja, botão de *RESET* pressionado ou *RESET_TIME* (definido no bloco de controle) igual a 1, zera a contagem a qualquer momento (assíncrono).

A entrada *SET_TIME* do bloco, quando em nível lógico alto, permite uma mudança “manual” do sinal *contador*, que define a saída *CONTA*, definindo-o como a entrada *SEL_TIME*. Se isso não ocorre, *contador* pode continuar com o mesmo valor ou ser decrementado.

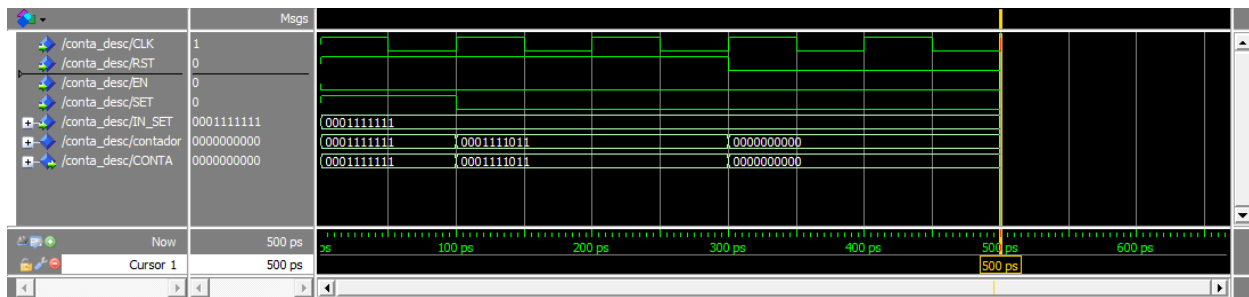
É a entrada *EN_TIME* que define se a contagem decresce (se 1) ou não (se 0). Quando ela está habilitada, os bits 0 a 5 de *contador*, representativos dos segundos, são decrementados de 1 a cada ciclo de *CLK1*. Quando essa contagem de segundos chega a zero, no próximo ciclo tem-se um decremento nos minutos (bits 9 a 6) e os segundos vão para 59 (111011 em binário).

Independentemente de *SET_TIME* e *EN_TIME*, quando *contador* é definido com o valor de segundos mais alto que 59, esse valor passa a 59 e o valor dos minutos permanece o mesmo, impedindo a entrada de um valor proibido.

Nas seguintes simulações de *CONTA_DESC*, *RST* é *RST_CONT*, *EN* é *EN_TIME*, *SET* é *SET_TIME* e *IN_SET* é *SEL_TIME*.



Observa-se que contador e, conseqüentemente, *CONTA*, só são carregados com *IN_SET* quando *SET* é 1. A partir do momento em que *EN* é 1, começa a contagem regressiva. Quando os segundos chegam a “000000”, passam a “111011” no ciclo seguinte, com um decremento também nos minutos.



Com o valor dos segundos de *IN_SET* maior que 59 (111011), os segundos do contador passam a 59 logo em seguida. Quanto *RST* é 0, independentemente do clock, a contagem é zerada.

1.4 Seletores

Os multiplexadores foram descritos fazendo-se uso de uma abordagem comportamental, por sua facilidade de implementação e eventual correção de falhas no projeto.

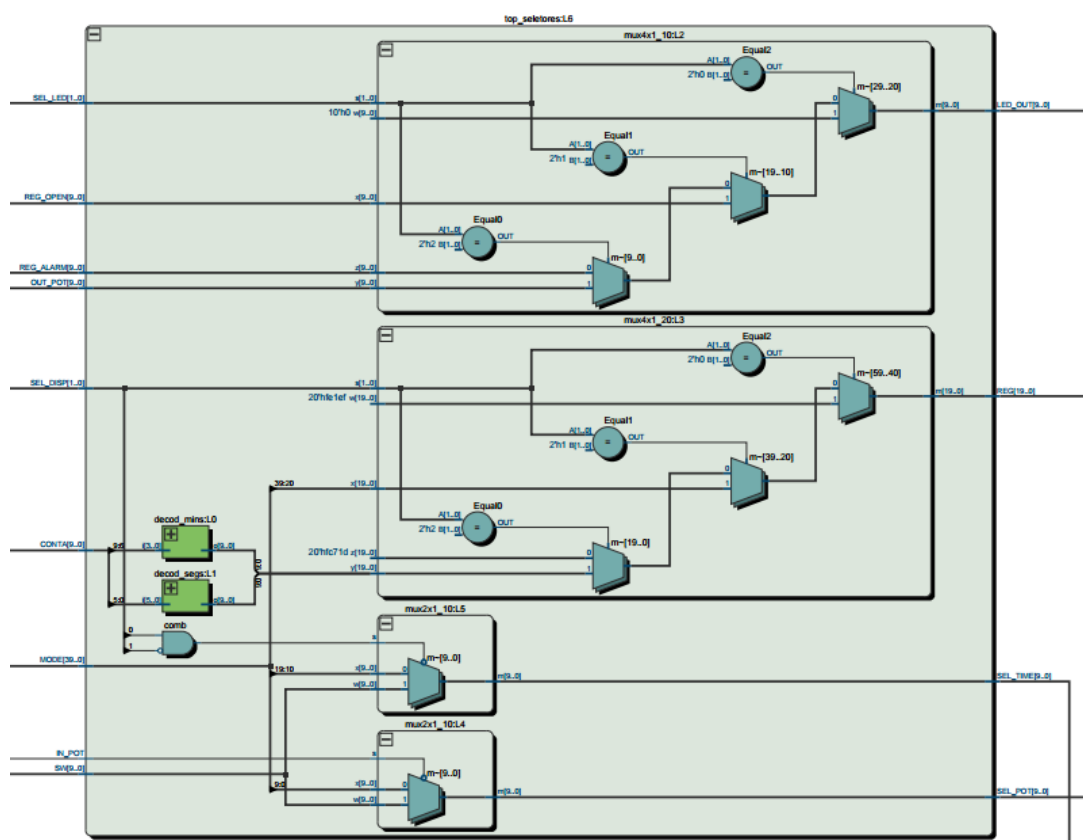
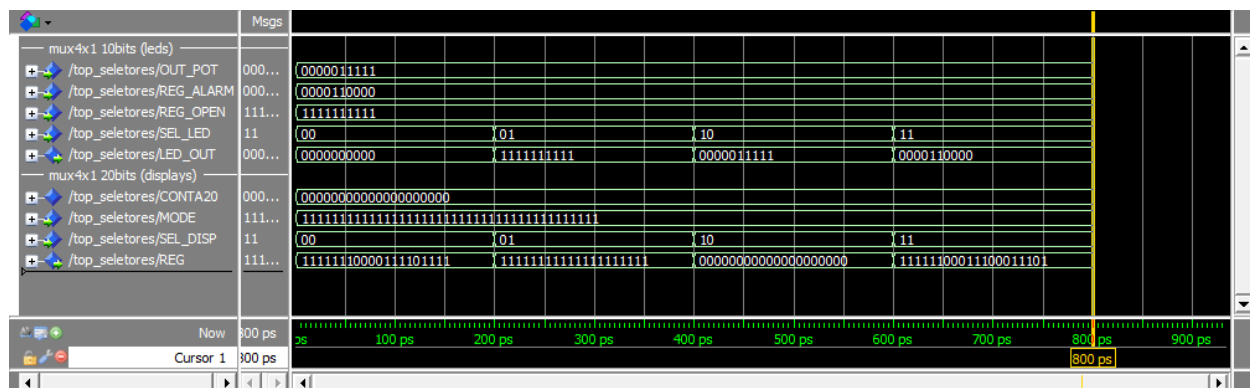


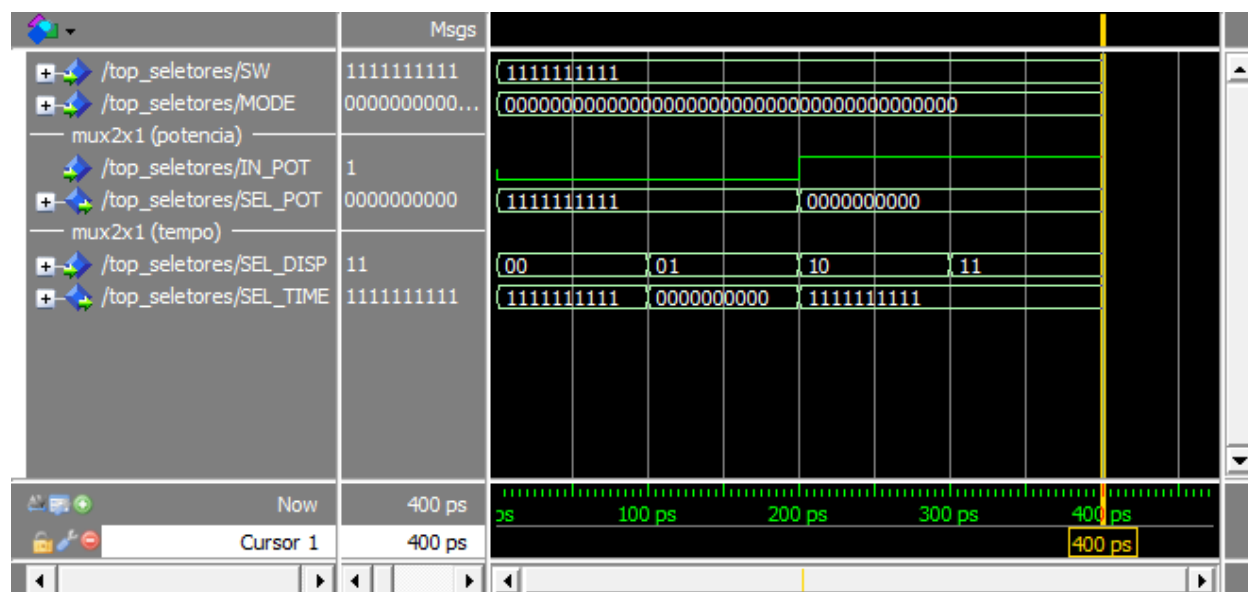
Diagrama de blocos de top_seletores. Tem-se os decodificadores que transformam CONTA em um formato adequado para exibição nos displays, a lógica combinacional com SEL_DISP cuja saída serve de entrada para um dos multiplexadores. Aqui, há quatro: dois 4x1, um de 10 e outro de 20 bits, e dois 2x1 de 10 bits, todos apresentados na sua forma estrutural no diagrama.



Simulação de top seletores mostrando mux4x1 10 e mux4x1 20.

No multiplexador 4x1 de 10 bits, a saída *LED_OUT*, que define o que será mostrado nos *LEDs*, pode ser zero, *REG_OPEN* (*LEDs* quando a porta está aberta), *OUT_POT* (*LEDs* mostrando a potência) ou *REG_ALARM* (*LEDs* no modo de alarme), se a entrada *SEL_LED* é 00, 01, 10 ou 11, respectivamente.

No multiplexador 4x1 de 20 bits, a saída *REG*, que define o que será mostrado nos *displays* de sete segmentos, depende da entrada *SEL_DISP*. Se *SEL_DISP* é 00, *REG* fará com que os *displays* mostrem a palavra *off*; se 01, *REG* será os 20 bits mais significativos de *MODE*, a saída da predefinição, e os *displays* mostrarão a palavra referente ao modo de aquecimento escolhido (sopa, chá etc.); se 10, *REG* será *CONTA20*, e, portanto, os *displays* mostrarão a contagem; finalmente, se *SEL_DISP* for 11, *REG* fica definido de tal forma que os *displays* mostram a palavra *hot*.



Simulação de mux2x1_10 para duas funções diferentes.

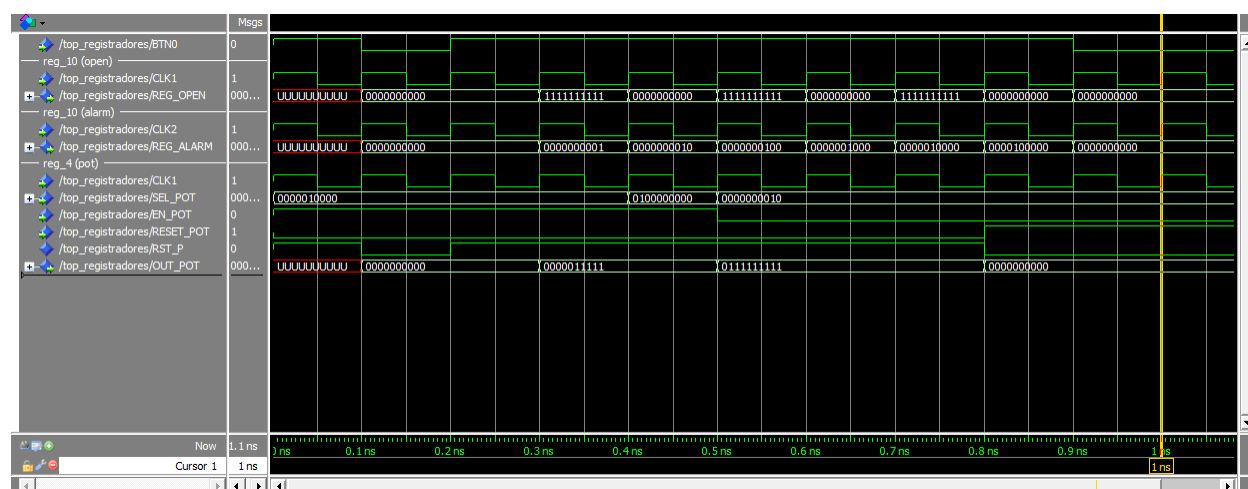
Para a definição de *SEL_TIME*, que define o tempo para a contagem decrescente, e *SEL_POT*, responsável por definir a potência de aquecimento, o mesmo multiplexador 2x1 de 10 bits é usado.

A potência depende da entrada *IN_POT*, de 1 bit. Quando ela está em nível lógico baixo, a potência fica definida pelos *switches*. Caso contrário, a predefinição (bits 0 a 9 de *MODE*) a define.

Já o tempo depende de *SEL_DISP*. Pela máquina de estados do bloco controlador (que será apresentada mais adiante), o tempo deve ser definido pela predefinição apenas quando *SEL_DISP* for 01. Caso contrário, ele vem dos *switches*.

1.5 Registradores

No bloco dos registradores, tem-se três registradores (sendo dois deles o mesmo com funções diferentes), um decodificador e uma memória que operam da maneira descrita a seguir.



Simulação de top_registradores mostrando as funções de reg_4 e reg_10. BTN0 é o botão de reset.

O registrador *reg_10* é usado para definir a saída para os *LEDs* em duas ocasiões no projeto: quando a porta está aberta e ele inverte o valor anterior a cada ciclo de *CLK1* (1 segundo), ou seja, *REG_OPEN*, a saída, vai de “0000000000” para “1111111111” e vice-versa continuamente, o que faz com que os *LEDs* pisquem todos ao mesmo tempo; e ao fim do aquecimento quando a saída *REG_ALARM* que vai para os *LEDs* faz com que eles apresentem o efeito de luz definido em *ROM_seq17.vhd* com uma frequência de 10 Hz (*CLK2*, na simulação, por simplicidade, tem período de 0,1 ns). Ambos têm seus valores resetados quando se pressiona o botão de *reset* (*KEY0*) (0,9 ns na simulação), independentemente do *clock*.

Já *reg_4* é um pouco diferente, pois tem-se a opção de habilitá-lo ou desabilitá-lo. Ele é usado para definir a saída para os *LEDs* quando estes mostram a potência de aquecimento (*OUT_POT*). Seu *clock* é *CLK1* (ilustrado na simulação com período de 0,1 ns) e ele só armazena um novo valor quando a entrada de *enable* *EN_POT* é 1. Seu valor é resetado quando a entrada *RST_P* é 0, e esta por sua vez é definida em *top_registradores* de tal forma que é 0 quando o botão de *reset* é pressionado ou quando a entrada *RESET_POT* (definida pelo bloco de controle) está em nível lógico alto (0,8 ns na simulação), independentemente do *clock*.

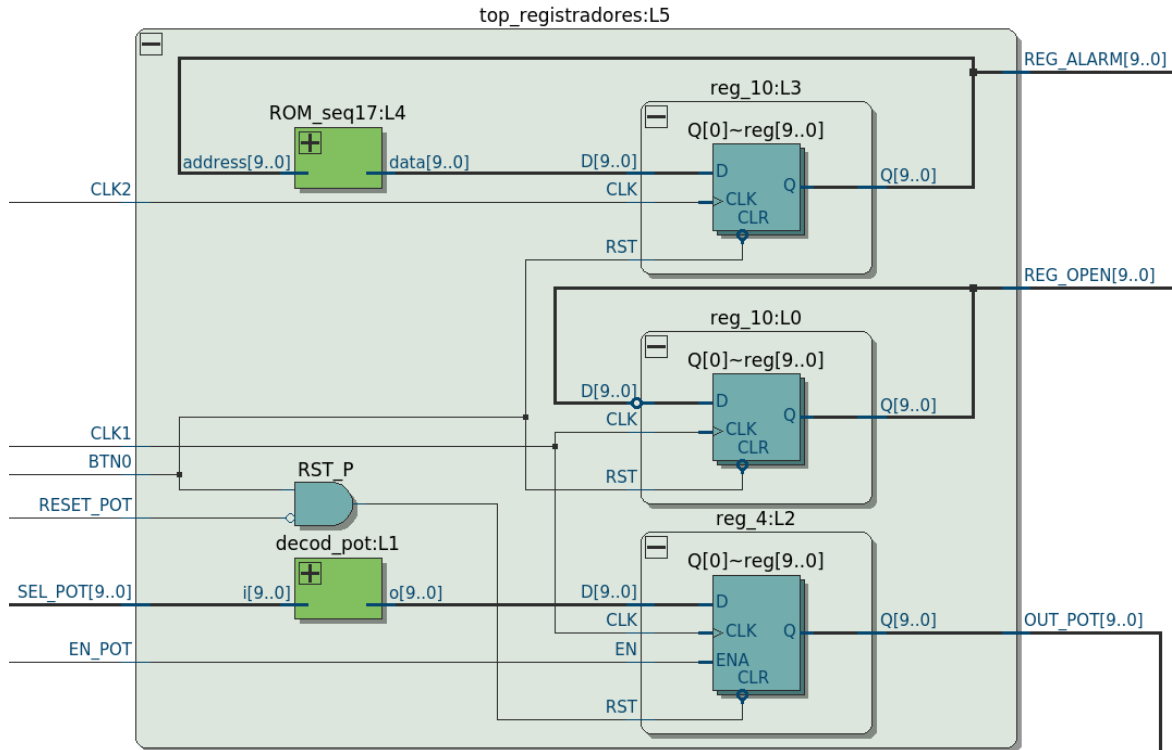


Diagrama de blocos de top_registradores.

O decodificador *decod_pot.vhd* para a entrada do registrador de potência foi elaborado com uma abordagem estrutural. Sua saída foi definida com portas *OR* e de tal maneira que todos os bits à direita do mais significativo da entrada, e inclusive o próprio, tenham nível lógico alto.

Para o primeiro registrador *reg_10*, temos um *flip-flop* tipo *D* de 10 bits (ou 10 *flip-flops* de 1 bit) com *reset* assíncrono. Seu *clock* é *CLK2*, de 10 Hz, e ele armazena um novo valor quando há uma transição positiva do mesmo. A entrada de dados é definida pela memória do efeito luminoso (*ROM_seq17*), cuja entrada é a própria saída do registrador. A entrada de *reset* (ou *CLR*) é ativa baixa, sendo acionada quando o botão *BTN0* é pressionado.

O segundo *reg_10* opera de maneira semelhante, apenas com as entradas definidas por outros valores. O *clock* é *CLK1*, de 1 Hz, e o registrador é sensível a sua transição positiva. A entrada de dados *D* é definida pela própria saída do registrador invertida. *Reset* é também acionada por *BTN0*.

Em *reg_4*, tem-se uma entrada a mais, de *enable*, que é ativa alta e definida pela entrada do bloco *EN_POT*. A entrada de dados *D* do registrador é a saída do decodificador da potência e passa à saída *Q* na transição positiva de *CLK1*. *Reset* é assíncrono e definido pela lógica combinacional mencionada anteriormente, uma *AND* de entradas *BTN0* e *RESET_POT* invertida. *CLR* (*reset*) é ativa baixa, sendo, portanto, acionado somente quando *BTN0* é pressionado ou *RESET_POT* é 1.

2. Controlador

A máquina de estados no bloco de controle foi elaborada com oito estados. São eles: *INIT* (*S0*), estado inicial do micro-ondas desligado; *OPEN* (*S1*), micro-ondas desligado e com a porta aberta; *IN_TIME* (*S2*), onde é definido o tempo do aquecimento no método manual; *IN_POT* (*S3*), onde é definida a potência do aquecimento; *WARM* (*S4*), do aquecimento propriamente dito; *READY* (*S5*), quando o aquecimento já está encerrado e a comida está pronta; *IN_PREDEF* (*S6*), onde tem-se a escolha do modo de aquecimento predefinido; e *PAUSE* (*S7*), estado em que o aquecimento se encontra pausado.

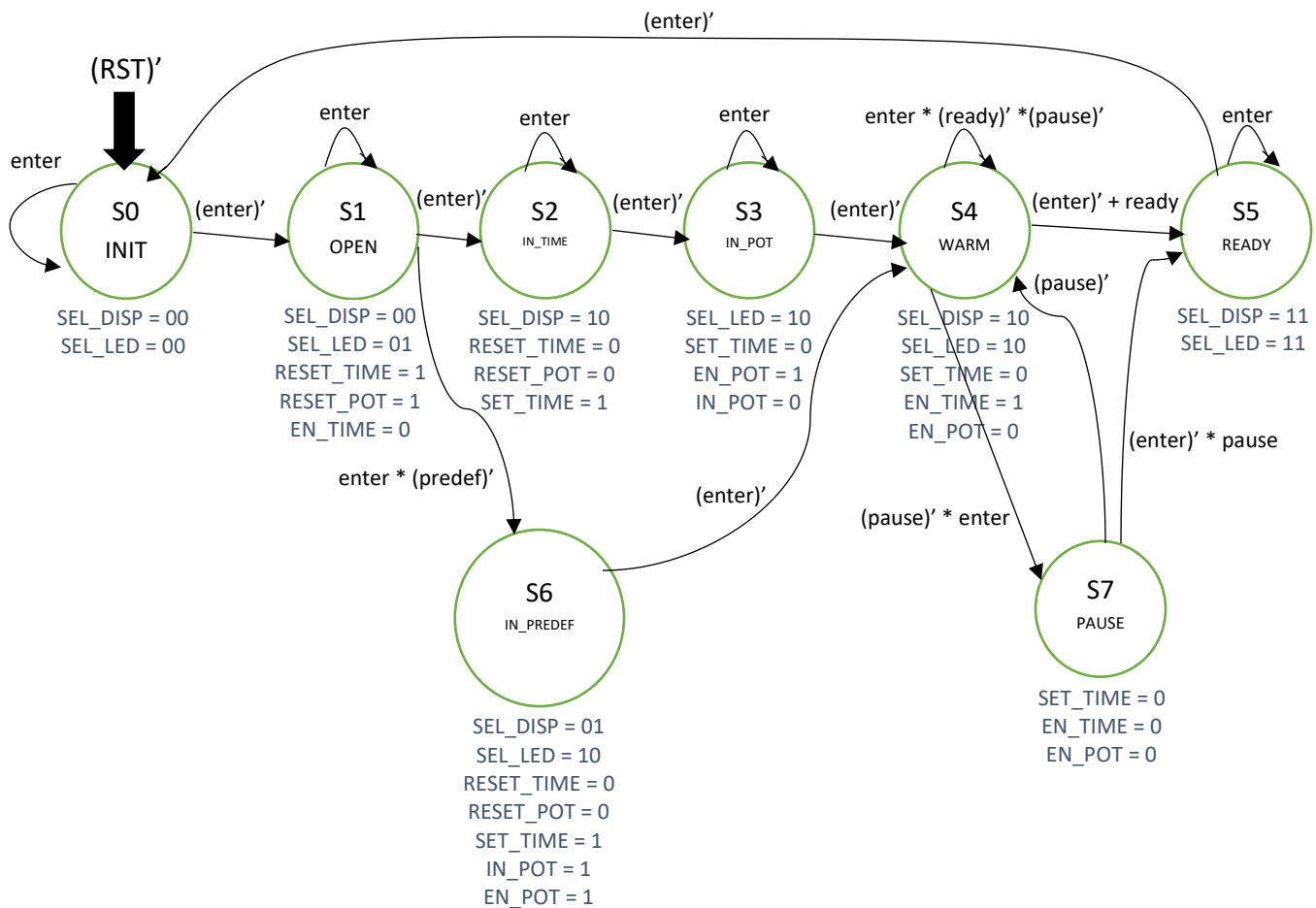


Diagrama da máquina de estados mostrando os estados, suas transições e os valores definidos por eles.



Simulação de FSM_control, que ilustra a máquina de estados do diagrama anterior.

O reset de *FSM_clock* é assíncrono e ativo baixo, e os demais botões são sincronizados com o *clock* (*CLOCK_50*, de 50 MHz). *READY* indica que a contagem do tempo chegou em zero.

Na simulação observa-se que pressionar *ENTER* passa do estado *S0* a *S1*, de *S1* a *S2*, de *S3* a *S4*, de *S4* a *S5* e de *S5* a *S1*. *S4* (*WARM*) é o único estado em que *PAUSE* e *READY* podem causar uma mudança de estado: o primeiro em nível lógico baixo faz a máquina passar para *S7* (*PAUSE*), e o segundo em alto leva a *S5* (*READY*). Quando se está em *S1* (*OPEN*), pressionar *PREDEF* passa a *S6* (*IN_PREDEF*). A qualquer momento, independentemente de ciclos de *clock*, pressionar *RESET* passa ao estado *INIT* (*S0*).

Em cada estado, as saídas estão definidas como no diagrama de estados. *SEL_DISP*, *SEL_LED* e *IN_POT* são entradas do bloco de seletores e indicam o que será mostrado nos *displays* e *LEDs* e se a potência deve ser definida manualmente ou pela memória da predefinição, respectivamente. *EN_POT* habilita o registrador da potência e *RESET_POT* o reseta. *EN_TIME* habilita a contagem regressiva de tempo, com *SET_TIME* habilitando a mudança manual de valor e *RESET_TIME* resetando a contagem, como já explicado na seção sobre o bloco dos contadores. A saída *ESTADOS* indica, em binário, o número do estado em que se encontra *FSM_control*, e é mostrada no *display* para facilitar a correção de erros.

3. Resultados e conclusões

Todos os blocos descritos no decorrer do trabalho foram interligados por *topo_microondas*. Ali definiu-se que saídas de quais blocos se ligam às entradas dos outros (algumas dessas foram descritas ao longo deste trabalho e nos comentários nos arquivos *VHDL*) e como as saídas e entradas da placa dialogam com o projeto. Mesmo que se trabalhe com um componente separado de cada vez, é preciso ter sempre a noção de onde ele se encaixa no contexto geral, para saber quais funcionalidades devem ser implementadas e quais determinam o que.

Uma vez compilados pelo Quartus, simulou-se cada bloco separadamente no ModelSim, onde alguns erros de lógica, principalmente na parte da contagem regressiva e do controlador, foram observados e consertados. Testando na placa DE1, fica ainda mais fácil observar as falhas de projeto. Ao final de todo o processo de simulação, teste e correção de *bugs*, o micro-ondas funcionou como esperado.

Conclui-se, portanto, que, tão importante quanto uma primeira análise do código à procura de erros são as simulações efetivas, onde se pode ver em tempo real o que está acontecendo e avaliar, por meio de testes de diferentes cenários (apertando vários botões, alterando os *switches* etc.), se tudo ocorre dentro do que se espera e tomar as medidas necessárias para normalizar qualquer situação que fuja do ideal.

Anexo A – Observações

Trabalhar com o FPGA e a linguagem VHDL se revelou um processo útil e divertido, apesar de, no começo, ter sido de difícil aprendizado. Depois que se aprende e se acostuma, percebe-se que é muito mais prático digitar algumas linhas de código do que usar diversos componentes físicos e um emaranhado de fios para desempenhar a mesma função.

O projeto do micro-ondas foi muito interessante de ser realizado e serviu bem para juntar os conhecimentos adquiridos ao longo do semestre tanto nas aulas práticas como nas teóricas. Desde a ideia de como realizar o projeto, passando pela descrição de cada componente em VHDL, suas ligações dentro dos blocos e finalmente as ligações entre blocos, as simulações e os testes na placa, tudo envolveu os conceitos chave da disciplina, que serão levados daqui para o resto do curso.