UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE – UFRN
INSTITUTO METRÓPOLE DIGITAL – IMD
BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO – BTI
LABORATÓRIO DE CIRCUITOS LÓGICOS

PROJETO
CIRCUITOS SEQUENCIAIS
Relógio com alarme

Hugo Thiago de Holanda Oliveira
Joel Felipe Ferreira Gomes

Natal, RN
2016

1. CIRCUITO PRINCIPAL

O circuito principal, escrito no arquivo "main_circuit.vhd", é composto pelos subcircuitos codificador de teclas, registrador e processador de dados, relógio, implementado como um contador, controle do alarme e, por fim, decodificadores de BCD (*Binary-coded decimal*) para *display* de 7 segmentos, além de multiplexadores (*Mux*) e portas *AND* para uso lógico.

As portas de entrada são basicamente as teclas de um teclado numérico de 0 a 9 (*key_0...9*), uma tecla SET (*key_S*) e uma ALARM/CANCEL (*key_A*), além de uma entrada *clk* para um gerador de pulsos de *clock*, para uso da lógica sequencial.

As portas de saída são quatro vetores de 7 bits (*Std_logic_vector*(0 to 6)) para representar quatro *displays* de 7 segmentos, no formato HH:MM, dois *displays* para horas e dois para os minutos.

A arquitetura foi descrita de forma comportamental, em sua maior parte. Seu desenho é mostrado na figura 1.

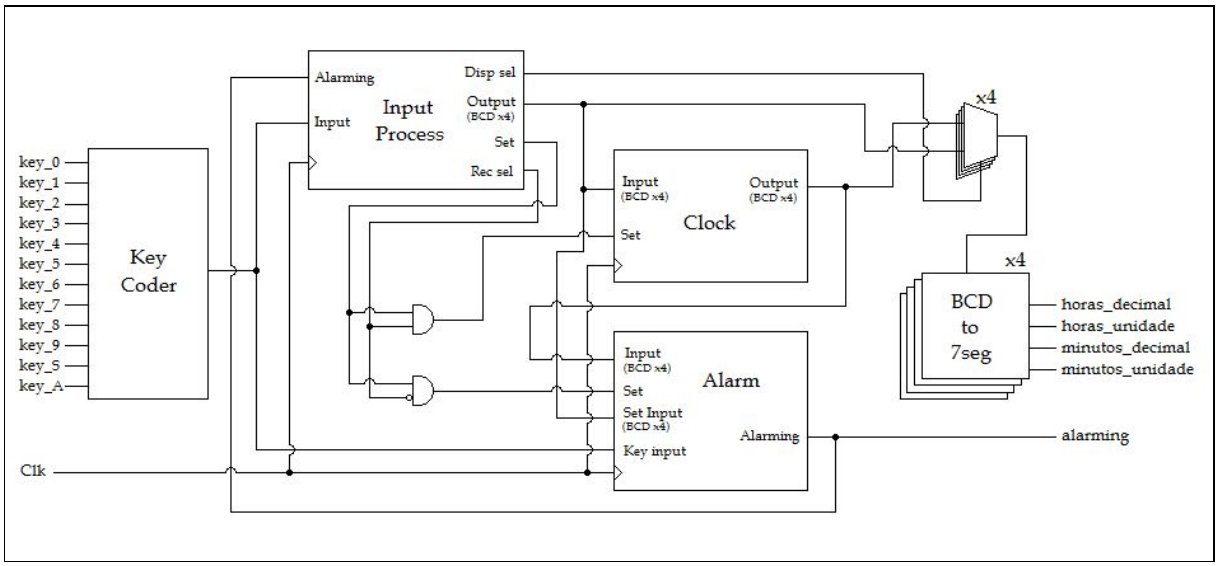

Figura 1. Diagrama de blocos do circuito principal.

Dos sinais (*signal*) declarados, podemos destacar ***set_selection*** e ***disp_selection***, que são utilizados como controles controles para o carregamento dos dados, seja no relógio ou alarme, no caso do ***set_selection***, e para seleção da saída nos *displays*, entre inputs ou relógio, no caso do ***disp_sel***.

2. CIRCUITO CODIFICADOR DE INPUTS

O codificador de inputs, escrito no arquivo “key_coder.vhd”, foi implementado de forma comportamental a partir da tabela verdade abaixo (tabela 1).

Tabela 1. Tabela verdade inputs para 4 bits

Entradas												Código			
0	1	2	3	4	5	6	7	8	9	Set	Alarm/ Cancel	A	B	C	D
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1
0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0
0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

A escolha desta tabela verdade foi feita pela facilidade que ela proporciona para a implementação do circuito planejado, oferecendo os valores das teclas numéricas já em BCD, além de sinalizar quando nenhuma tecla foi apertada com o valor “1111” que é útil para a detecção de uma circunstância de parada do alarme.

3. CIRCUITO DE REGISTRO DAS ENTRADAS E PROCESSAMENTO DE DADOS

Este circuito foi implementado como uma máquina de estados finitos, seu arquivo fonte é o “input_process.vhd” .

O estado inicial é o ***idle*** (ocioso), nesse estado, a saída ***disp_sel*** está como ‘0’, sinalizando para o circuito principal que a saída deve mostrar o relógio.

No momento que o jogador aperta SET (cod. “1010”) ou ALARM/CANCEL (cod. 1011) o estado passa a ser ***h_d*** e o ***disp_sel*** passa para ‘1’, mostrando o registo dos inputs, que inicia mostrando todos os LEDs apagados. dependendo da tecla apertada no estado *idle*, seja SET ou ALARM/CANCEL o valor da saída ***out_sel*** será 1 ou 0, respectivamente, para sinalizar onde o valor registrado será gravado.

No estado ***h_d*** após o usuário apertar alguma tecla numérica (nota: é utilizado um *signal* ***lock*** para evitar que o valor da tecla seja lido mais de uma vez, sem a tecla ter sido desprecionada) e o seu valor estiver entre 0 e 2, este número será guardado como dezena das horas, e o estado passa para ***h_u***. Neste estado após inserção de um número passará para o estado ***m_d*** e este número será guardado no segundo registro (unidade das horas)

No estado ***m_d***, caso o número inserido esteja entre 0 e 5, este é registrado como dezena dos minutos, e por fim o estado passa para ***m_u***. Neste último estado, se o usuário apertar algum número, é desencadeada a lógica de regulação do alarme ou relógio, dependendo da tecla inicialmente apertada no estado ***idle***, a saída ***disp_sel*** é então mudada para ‘0’, mostrando o horário atual do relógio nos *displays* de sete segmentos, e a saída ***recording*** passa para ‘1’ sinalizando que o circuito alvo pode ser ajustado, e o estado da máquina passa para ***idle***, estado inicial, resetando o valor de ***recording*** no próximo pulso do ***clk***.

Vale resaltar que em qualquer dos estados de ***h_d*** a ***m_u*** se o usuário apertar ALARM/CANCEL, o ***disp_sel*** volta para ‘0’ e o estado para ***idle***, sinalizando o cancelamento da operação.

Foi inserida a entrada ***alarming*** para que, caso o relogio esteja alarmando, o input utilizado para desarmar o alarme não interfina na lógica de inserção de registros, estando bloqueada qualquer mudança nesse subcircuito até o fim do alarme.

4. CIRCUITO CONTADOR DO RELÓGIO

O subcircuito do relógio, do arquivo “main_clock.vhd” foi implementado com o uso de um contador de 0 até 60, representando os segundos. Para este projeto, a frequência foi estabelecida como 1Hz (um ciclo por segundo).

Quando o valor do contador passa de 59 para 0, entra em cena um outro contador, que é das unidades dos minutos, que é incrementada em 1, e este por sua vez, quando passa de 9 para 0, incrementa o contador seguinte, das dezenas e minutos, seguindo a mesma lógica para as unidades e dezenas das horas.

Neste subcircuito ainda foram inseridas entradas para quatro vetores de 4 bits BCD e um *Std_logic* ***set*** para ajuste da hora atual. Quando a hora é ajustada, o contador de segundos é resetado para 0.

5. CIRCUITO DE CONTROLE DO ALARME

O subcircuito de controle do alarme, do arquivo "alarm.vhd" foi implementado como uma máquina de estados finitos, que é manipulada pelo process ***clock_alrm.***.

O estado inicial é o ***idle*** (ocioso), nesse estado, o contador ***soneca*** (tempo de duração da soneca) é setado como '0', sinalizando que a soneca está desativada. Se o horário marcado no relógio, for o mesmo horário que está setado como horário do alarme, então o ***alarm*** passa a ser '1', fazendo com que o valor da saída ***alarming*** passe a ser '1', sinalizando que o alarme passa a estar ativo, além de setar o valor de ***sec*** (tempo de duração do alarme) para "1010".

No estado ***actived*** (tocando), o ***locked*** (sinal que indica se o alarme estava tocando em um determinado minuto) é setado como '1', e se ***sec*** for maior que 0, ele decrementa o ***sec*** e verifica se ele é igual a 1 porque se for, o ***alarm*** é setado como '0'.

No estado ***sleep*** (soneca ativada), é verificado se ***soneca*** é maior que 0 para poder decrementar a ***soneca***, e se for também verifica-se se ***soneca*** é igual é 1 para setar o ***alarme*** como '1' e ***sec*** como "1010".

No estado ***off*** (desligado pelos botões), o ***alarm*** é setado como '0', e se o horário marcado no relógio, for o diferente do horário que está setado como horário do alarme, ***locked*** é setado como '0'.

Além disso, o subcircuito possui outro process denominado ***inputs***, que serve para fazer a transição dos e também para setar o horário do alarme. Primeiramente ele verifica se ***set_alarme*** é igual a 1 para poder setar o horário do alarme como o horário passado como parâmetro do subcircuito, Depois verifica-se se ***alarm*** é igual a '1', ou seja, se o alarme está ativado, e se for, verifica se ***input_code*** é menor que 15, ou seja, algum botão foi apertado, pois se o botão apertado for o botão de alarme, o ***current_state*** é setado como ***sleep***, senão é setado como ***off***. Se nenhum botão foi apertado o ***current_state*** é setado como ***actived***. Se o ***alarm*** não for '1', é verificado se ***soneca*** é maior que 0 para poder setar o ***current_state*** como ***sleep,*** se não for maior, é verificado se ***locked*** é igual a '0' para poder setar o ***current_state*** como ***idle,*** se não for igual, é verificado se ***locked*** é igual a '1' para poder setar o ***current_state*** como ***off.***

6. CIRCUITO DECODIFICADOR DE BCD PARA *DISPLAY* DE 7 SEGMENTOS

Para a conversão dos dados de BCD para *display* de 7 segmentos, foi construida uma tabela verdade (tabela 2), e por ela, foram obtidas as equações boleanas para implementação estrutural.

Tabela 2. Tabela verdade BCD para 7 segmentos

BCD				7 segmentos						
A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

As equações boleanas foram obtidas pelo conceito de mintermos, ou seja: pela soma dos produtos dos termos das linhas que possuem o valor “1” para o segmento esperado. Em seguida as fórmulas foram simplificadas por mapas de Karnaugh. As expressões simplificadas, que foram implementadas de forma estrutural são mostradas abaixo:

a = A'.C + A'.B.D + B'.C'.D' + A.B'.C'

b = A'.B' + B'.C' + A'.C'.D' + A'.C.D

c = A'.B + B'.C' + A'.D

d = A'.C.D' + A'.B'.C + A.B'.C' + A'.B.C'.D + B'.C'.D'

e = A'.C.D' + B'.C'.D'

f = A'.B.D' + A'.B.C' + A.B'.C' + B'.C'.D'

g = A'.C.D' + A'.B'.C + A'.B.C' + A.B'.C'