



UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO
CONCEPÇÃO ESTRUTURADA DE CIRCUITOS INTEGRADOS
CURSO DE ENGENHARIA DE COMPUTAÇÃO

HENRIQUE ELPÍDIO RUFINO ARAÚJO

RELATÓRIO PROJETO – UNIDADE 01

JOÃO PESSOA

2022

HENRIQUE ELPÍDIO RUFINO ARAÚJO

RELATÓRIO PROJETO – UNIDADE 01

Relatório apresentado à disciplina de Concepção Estruturada de Circuitos Integrados, como requisito de avaliação do curso de Engenharia de computação do Centro de Informática da Universidade Federal da Paraíba – UFPB.

Prof. Dra. Verônica Maria Lima Silva

JOÃO PESSOA

2022

SUMÁRIO

01. INTRODUÇÃO.....	4
02. MATERIAIS E MÉTODOS.....	4
03. RESULTADOS.....	8
04. CONCLUSÃO.....	11

01. INTRODUÇÃO

O projeto consiste na implementação de um relógio digital através da linguagem de descrição de hardware System Verilog. O relógio tem como entradas um sinal de clock de 50Mhz e um botão de reset e terá como saídas seis displays de sete seguimentos. A metodologia de projeto usada foi a top-down, e junto a interface global (modulo topo da hierarquia) também foi nos dado alguns macro-blocos internos e suas interfaces, é o caso dos módulos enable_1hz (que nada mais é que um divisor de clock) e bcd_7seg, que como o nome indica, recebe como entrada um código em bcd e na saída acende os leds de um display de sete seguimentos de maneira a formar o número recebido através do código. Ficando a implementação dos demais módulos, o módulo que conta segundos, o que conta minutos e o que conta horas sob nossa responsabilidade.

02. MATERIAIS E MÉTODOS

Durante todo o desenvolvimento desse projeto foram utilizadas duas ferramentas, o Quartus II Lite Edition na versão 20.1 e o ModelSim, o primeiro é um software de projeto para dispositivos lógicos programáveis, o segundo é um ambiente multilinguagem para simulação de linguagens de descrição de hardware.

Como dito anteriormente, a metodologia utilizada para desenvolvimento do projeto foi a top-down, cabendo ao aluno a implementação dos módulos maq_s (módulo que conta os segundos), maq_m (módulo que conta os minutos) e maq_h (módulo que conta as horas), além, é claro, de fazer a instanciação e a ligação desses módulos com os demais módulos no módulo topo (pai). Sendo assim, comecei com a implementação do módulo maq_s (Imagem-01).

```

1 module maq_s (input maq_s_clock,
2 input maq_s_reset,
3 input maq_s_enable1hz,
4 output logic [3:0] maq_s_bcd_s_lsd,
5 output logic [2:0] maq_s_bcd_s_msd,
6 output logic maq_s_incremento_minuto);
7
8     always_ff @(posedge maq_s_clock)
9     begin
10         if(!maq_s_reset)
11         begin
12             maq_s_bcd_s_lsd <= 4'd0;
13             maq_s_bcd_s_msd <= 3'd0;
14             maq_s_incremento_minuto <= 1'b0;
15         end
16     else
17     begin
18         if(maq_s_enable1hz)
19         begin
20             if(maq_s_bcd_s_lsd == 4'd9)
21             begin
22                 if(maq_s_bcd_s_msd == 3'd5)
23                 begin
24                     maq_s_incremento_minuto <= 1'b1;
25                     maq_s_bcd_s_lsd <= 4'd0;
26                     maq_s_bcd_s_msd <= 3'd0;
27                 end
28                 else
29                 begin
30                     maq_s_incremento_minuto <= 1'b0;
31                     maq_s_bcd_s_lsd <= 4'd0;
32                     maq_s_bcd_s_msd <= maq_s_bcd_s_msd + 3'd1;
33                 end
34             end
35         else
36         begin
37             maq_s_incremento_minuto <= 1'b0;
38             maq_s_bcd_s_lsd <= maq_s_bcd_s_lsd + 4'd1;
39         end
40     end
41 end
42 end
43 endmodule

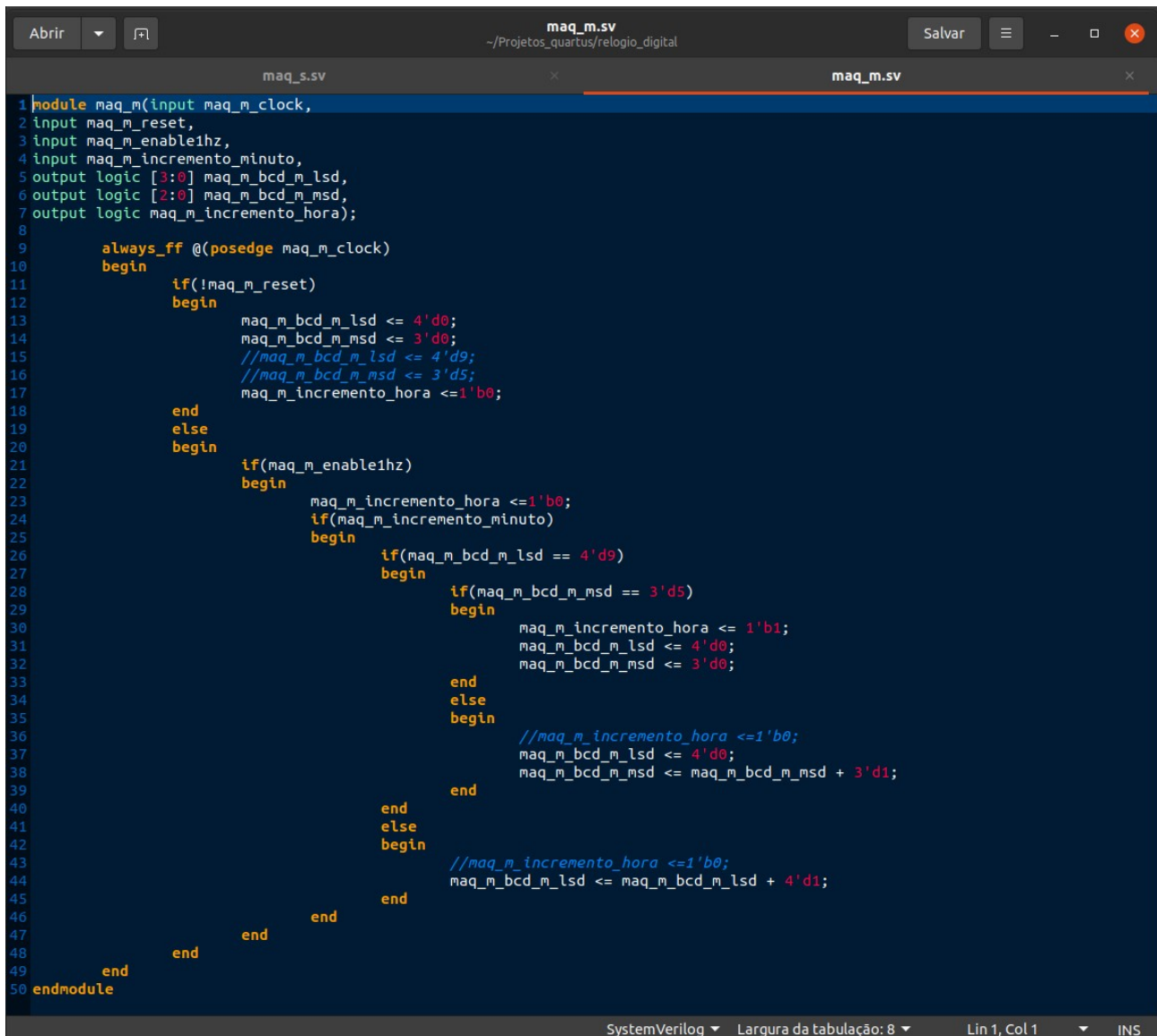
```

Imagem-01

Ele tem três entradas e três saídas, sendo uma entrada para o sinal clock, uma entrada para o botão de reset e uma entrada para o sinal de clock convertido de 1hz, uma saída de quatro bits para o display de sete seguimentos do dígito menos significativo dos segundos, uma saída de três bits para o display de sete seguimentos do dígito mais significativo dos segundos e, por fim, uma saída para o incremento do minuto. Esse módulo consiste de um único bloco procedural do tipo `always_ff` construído com lógica sequencial. Este bloco é sensível a passagem para borda positiva do clock, ele testa se o botão de reset está pressionado, caso esteja, ele zera as saídas, caso contrário, ele testa novamente se o sinal de 1hz está na borda positiva para aí sim decidir qual(is) saída(s) serão incrementadas ou zeradas, ao contar 59 segundos ele incrementa a saída `maq_s_incremento_minuto`.

A lógica utilizada na implementação do módulo `maq_m` (Imagem-02) é semelhante a usada no módulo `maq_s`, só que agora além de testar as entradas do clock, do botão de

reset e do enable1hz, também será necessário testar se a entrada `maq_m_incrementa_minuto` está em 1, para aí sim decidir qual(is) incrementar ou zerar.



```

1 module maq_m(input maq_m_clock,
2 input maq_m_reset,
3 input maq_m_enable1hz,
4 input maq_m_incrementa_minuto,
5 output logic [3:0] maq_m_bcd_m_lsd,
6 output logic [2:0] maq_m_bcd_m_msd,
7 output logic maq_m_incrementa_hora);
8
9     always_ff @(posedge maq_m_clock)
10    begin
11        if(!maq_m_reset)
12        begin
13            maq_m_bcd_m_lsd <= 4'd0;
14            maq_m_bcd_m_msd <= 3'd0;
15            //maq_m_bcd_m_lsd <= 4'd9;
16            //maq_m_bcd_m_msd <= 3'd5;
17            maq_m_incrementa_hora <= 1'b0;
18        end
19        else
20        begin
21            if(maq_m_enable1hz)
22            begin
23                maq_m_incrementa_hora <= 1'b0;
24                if(maq_m_incrementa_minuto)
25                begin
26                    if(maq_m_bcd_m_lsd == 4'd9)
27                    begin
28                        if(maq_m_bcd_m_msd == 3'd5)
29                        begin
30                            maq_m_incrementa_hora <= 1'b1;
31                            maq_m_bcd_m_lsd <= 4'd0;
32                            maq_m_bcd_m_msd <= 3'd0;
33                        end
34                        else
35                        begin
36                            //maq_m_incrementa_hora <= 1'b0;
37                            maq_m_bcd_m_lsd <= 4'd0;
38                            maq_m_bcd_m_msd <= maq_m_bcd_m_msd + 3'd1;
39                        end
40                    end
41                    else
42                    begin
43                        //maq_m_incrementa_hora <= 1'b0;
44                        maq_m_bcd_m_lsd <= maq_m_bcd_m_lsd + 4'd1;
45                    end
46                end
47            end
48        end
49    end
50 endmodule

```

Imagem-02

Por sua vez, o módulo `maq_h` (Imagem-03) é semelhante aos dois módulos anteriores, com a diferença que agora não temos mais uma entrada do tipo `incrementa_minuto`, agora temos uma entrada `incrementa_hora`, que quando ativa ele decide qual(is) saída(s) incrementar ou zerar.

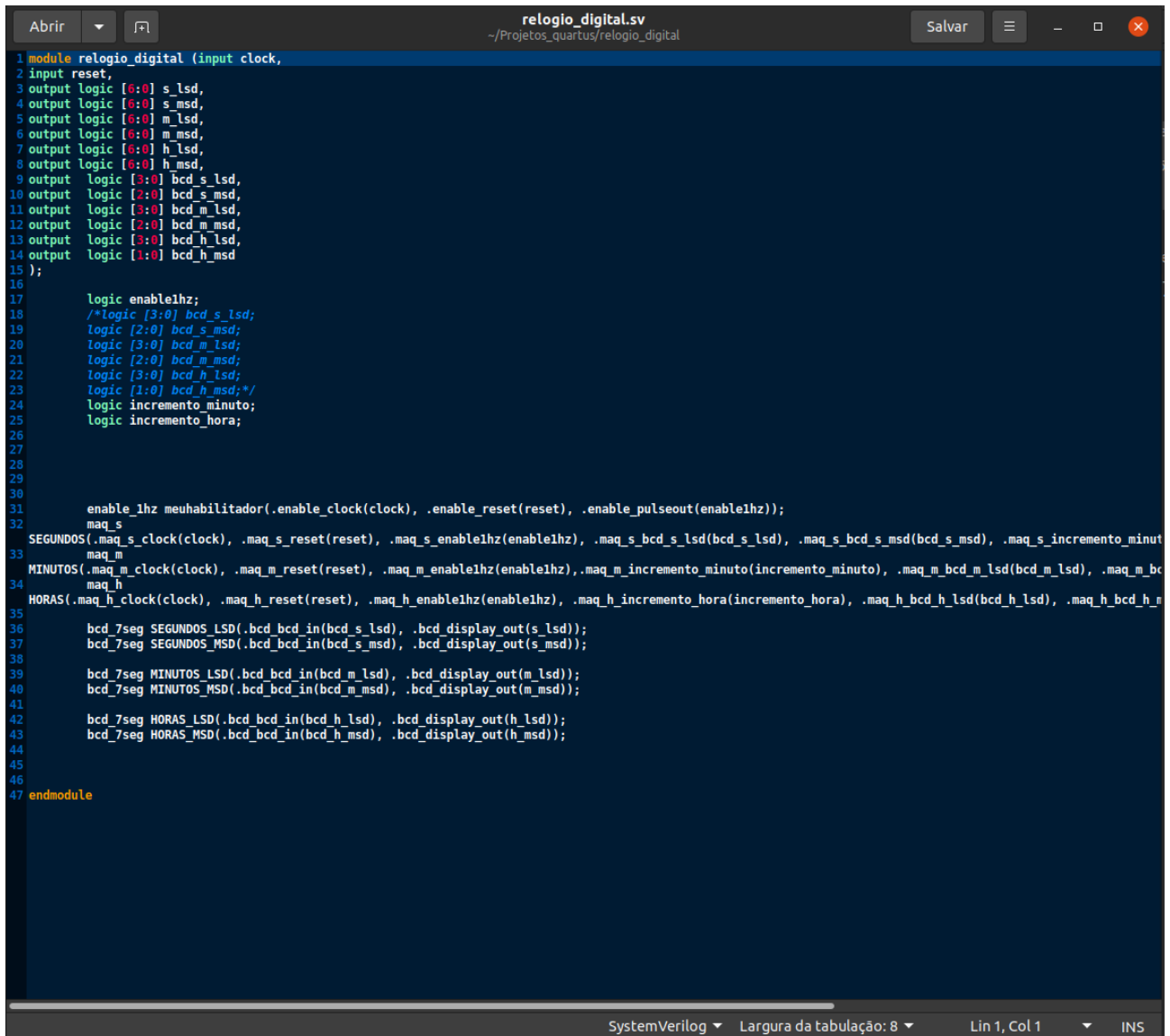
```

1 module maq_h(input maq_h_clock,
2 input maq_h_reset,
3 input maq_h_enable1hz,
4 input maq_h_incremento_hora,
5 output logic [3:0] maq_h_bcd_h_lsd,
6 output logic [1:0] maq_h_bcd_h_msd);
7
8     always_ff @(posedge maq_h_clock)
9     begin
10         if(!maq_h_reset)
11         begin
12             maq_h_bcd_h_lsd <= 4'd0;
13             maq_h_bcd_h_msd <= 2'd0;
14         end
15         else
16         begin
17             if(maq_h_enable1hz)
18             begin
19                 if(maq_h_incremento_hora)
20                 begin
21                     if(maq_h_bcd_h_msd == 2'd2)
22                     begin
23                         if(maq_h_bcd_h_lsd == 4'd3)
24                         begin
25                             maq_h_bcd_h_lsd <= 4'd0;
26                             maq_h_bcd_h_msd <= 2'd0;
27                         end
28                         else
29                         begin
30                             maq_h_bcd_h_lsd <= maq_h_bcd_h_lsd + 4'd1;
31                         end
32                     end
33                     else
34                     begin
35                         if(maq_h_bcd_h_lsd == 4'd9)
36                         begin
37                             maq_h_bcd_h_lsd <= 4'd0;
38                             maq_h_bcd_h_msd <= maq_h_bcd_h_msd + 2'd1;
39                         end
40                         else
41                         begin
42                             maq_h_bcd_h_lsd <= maq_h_bcd_h_lsd + 4'd1;
43                         end
44                     end
45                 end
46             end
47         end
48     end
49 endmodule

```

Imagem-03

O modulo topo (Imagem-04), além de contar com as saídas e entradas contidas na requisição, tomei a liberdade de adicionar mais 6 saídas (bcd_s_lsd, bcd_s_msd, bcd_m_lsd, bcd_m_msd, bcd_h_lsd e bcd_h_msd), que além de servirem como variáveis de ligação entre os módulos também facilitarão a visualização dos resultados.



```

1 module relogio_digital (input clock,
2 input reset,
3 output logic [6:0] s_lsd,
4 output logic [6:0] s_msd,
5 output logic [6:0] m_lsd,
6 output logic [6:0] m_msd,
7 output logic [6:0] h_lsd,
8 output logic [6:0] h_msd,
9 output logic [3:0] bcd_s_lsd,
10 output logic [2:0] bcd_s_msd,
11 output logic [3:0] bcd_m_lsd,
12 output logic [2:0] bcd_m_msd,
13 output logic [3:0] bcd_h_lsd,
14 output logic [1:0] bcd_h_msd
15 );
16
17     logic enable1hz;
18     /*logic [3:0] bcd_s_lsd;
19     logic [2:0] bcd_s_msd;
20     logic [3:0] bcd_m_lsd;
21     logic [2:0] bcd_m_msd;
22     logic [3:0] bcd_h_lsd;
23     logic [1:0] bcd_h_msd;*/
24     logic incremento_minuto;
25     logic incremento_hora;
26
27
28
29
30
31     enable_1hz meuabilitador(.enable_clock(clock), .enable_reset(reset), .enable_pulseout(enable1hz));
32     maq_s
33     SEGUNDOS(.maq_s_clock(clock), .maq_s_reset(reset), .maq_s_enable1hz(enable1hz), .maq_s_bcd_s_lsd(bcd_s_lsd), .maq_s_bcd_s_msd(bcd_s_msd), .maq_s_incremento_minuto(
34     maq_m
35     MINUTOS(.maq_m_clock(clock), .maq_m_reset(reset), .maq_m_enable1hz(enable1hz), .maq_m_incremento_minuto(incremento_minuto), .maq_m_bcd_m_lsd(bcd_m_lsd), .maq_m_bcd_m_msd(bcd_m_msd),
36     maq_h
37     HORAS(.maq_h_clock(clock), .maq_h_reset(reset), .maq_h_enable1hz(enable1hz), .maq_h_incremento_hora(incremento_hora), .maq_h_bcd_h_lsd(bcd_h_lsd), .maq_h_bcd_h_msd(bcd_h_msd),
38
39     bcd_7seg SEGUNDOS_LSD(.bcd_bcd_in(bcd_s_lsd), .bcd_display_out(s_lsd));
40     bcd_7seg SEGUNDOS_MSD(.bcd_bcd_in(bcd_s_msd), .bcd_display_out(s_msd));
41
42     bcd_7seg MINUTOS_LSD(.bcd_bcd_in(bcd_m_lsd), .bcd_display_out(m_lsd));
43     bcd_7seg MINUTOS_MSD(.bcd_bcd_in(bcd_m_msd), .bcd_display_out(m_msd));
44
45     bcd_7seg HORAS_LSD(.bcd_bcd_in(bcd_h_lsd), .bcd_display_out(h_lsd));
46     bcd_7seg HORAS_MSD(.bcd_bcd_in(bcd_h_msd), .bcd_display_out(h_msd));
47
48 endmodule

```

Imagem-04

03. RESULTADOS

Utilizando a ferramenta RTL Viewer do Quartus II, obtemos a representação de um diagrama de blocos de como ficou o projeto (Imagem-05). Partindo agora para a simulação, foi utilizado a ferramenta ModelSim.

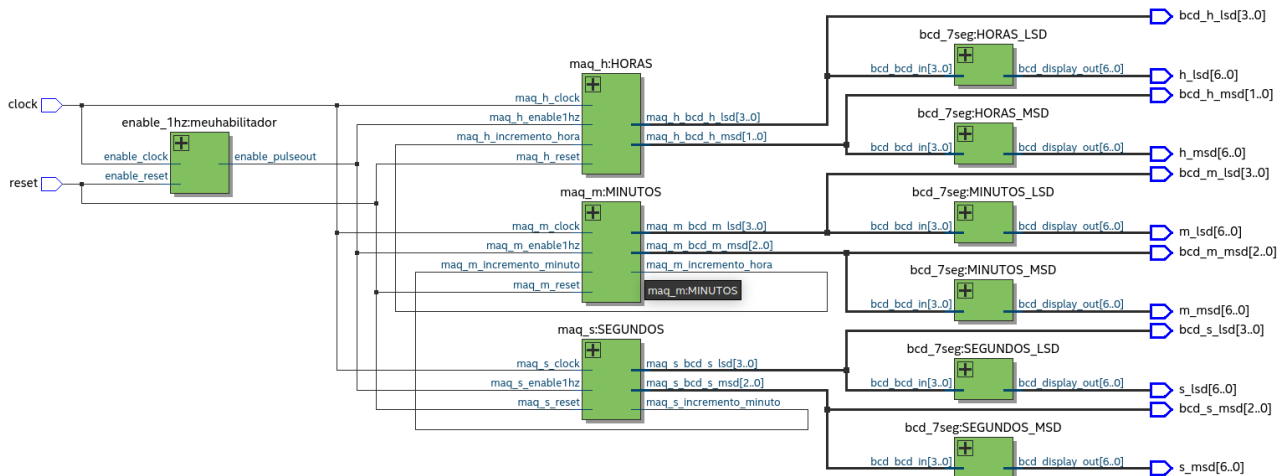


Imagem-05

os resultados a seguir (Imagem-06), foram obtidos com um sinal de clock cujo período é 0.2 ns (5Ghz) e cujo divisor de clock é 2:1, ou seja, a cada dois ciclos de clock do sinal de entrada, temos 1 ciclo no enable1hz, essas alterações foram necessárias para que fosse possível observar o comportamento do circuito, uma vez que o período de simulação do ModelSim é muito curto e um divisor de clock grande, como o proposto inicialmente, iria proporcionar um resultado com poucas mudanças nos sinais de saída. Fazendo isso conseguimos observar mudanças até o dígito mais significativo do minuto.

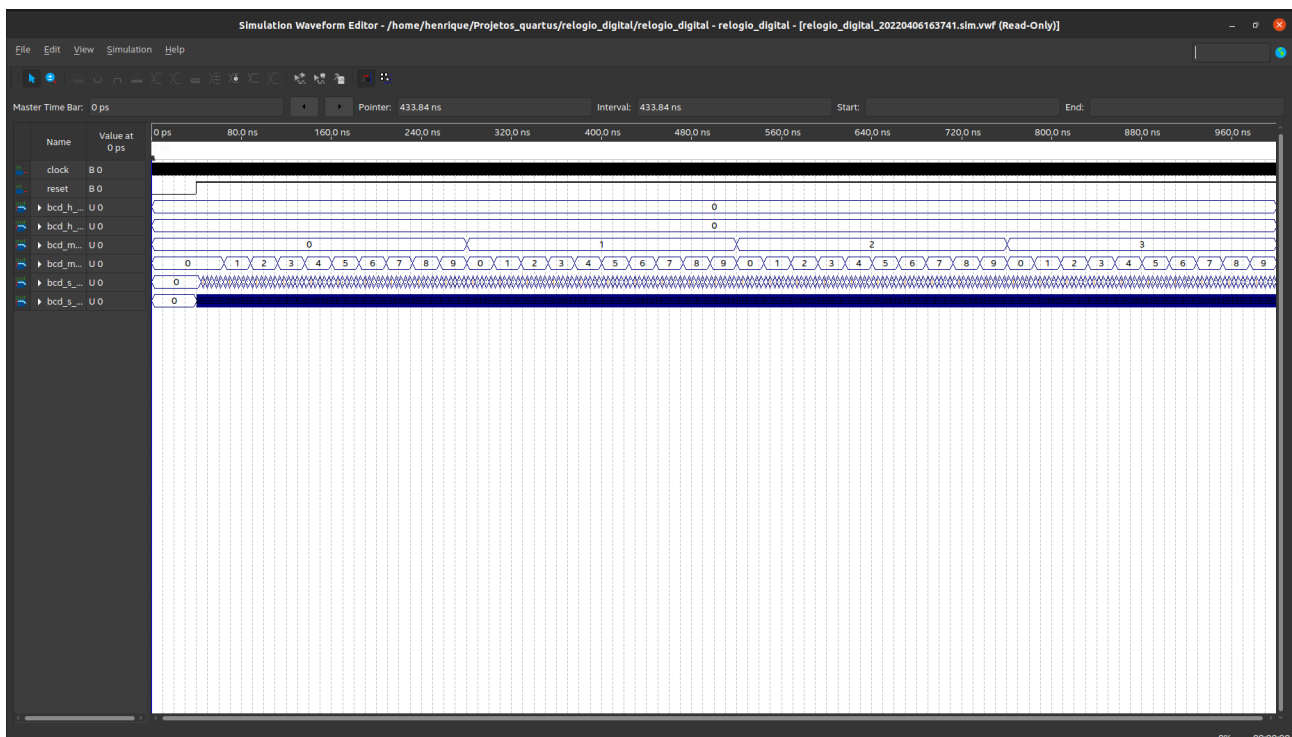


Imagem-06

Dando um zoom na imagem-06, conseguimos observar com mais clareza a mudança dos estados nas saídas:

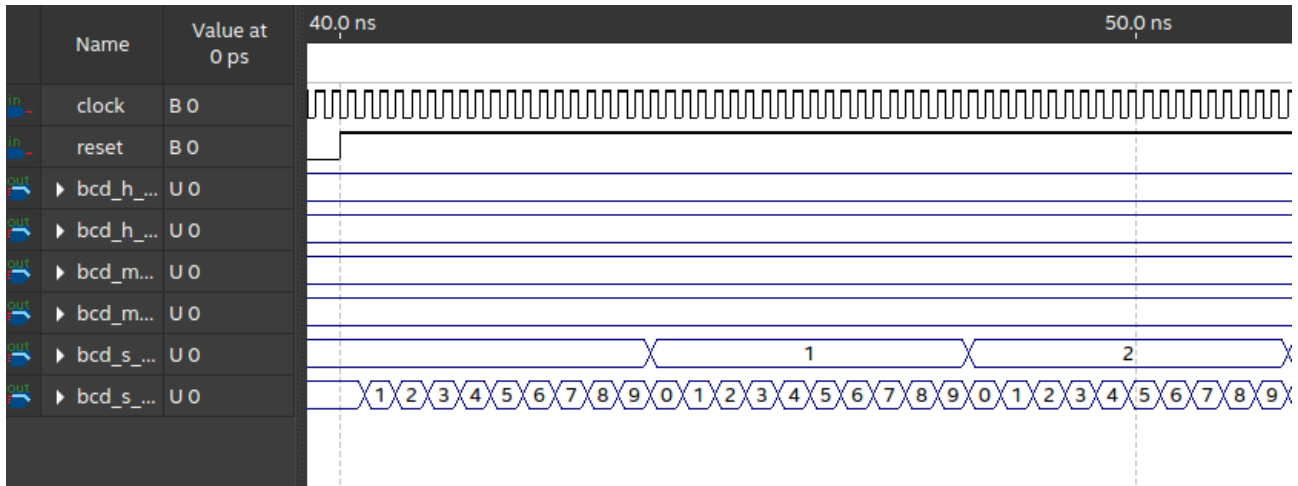


Imagem-07

Agora para forçar uma mudança de estado nas saídas dos dígitos das horas, uma vez que na simulação atual só conseguimos ir até 4 no dígito mais significativo dos minutos, comentei as linhas 13 e 14 do arquivo do módulo da máquina de minutos (maq_m) e descomentei as linhas 15 e 16, para forçar que o módulo ao ser iniciado, ele inicie contando em 59 em vez de 00. Obtemos o seguinte resultado:

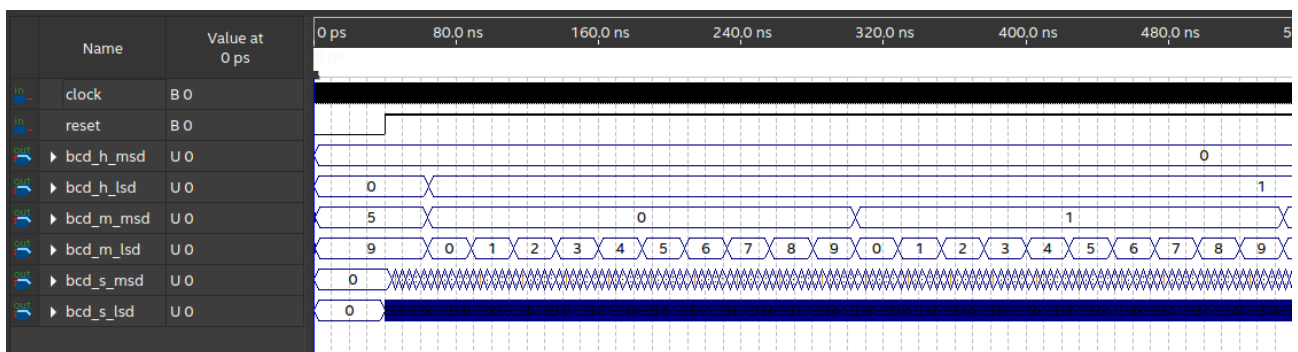


Imagem-08

como é possível notar, houve uma mudança de estado na saída bcd_h_lsd, configurando assim um incremento no dígito menos significativo das horas.

04. CONCLUSÃO

De acordo com os resultados obtidos, é possível afirmar que o dispositivo está funcionando de acordo com o planejado e que os requisitos especificados na descrição do projeto foram integralmente atendidos.