Universidade Federal de Pernambuco - UFPE
CIN - Centro de Informática

Projeto de Sistemas Digitais
Cronômetro Digital

Williams Douglas José dos Santos (wdjs@cin.ufpe.br)
Vinícius Felipe Barbosa (vfb2@cin.ufpe.br)
Lucas Yule Rocha de Melo Araújo (lyrma@cin.ufpe.br)
Carlos Eduardo Mendonça Clark (cemc@cin.ufpe.br)

RECIFE
2021

Sumário:

1.Introdução:

Esse projeto tem como objetivo o desenvolvimento de um Cronômetro Digital, desenvolvido em Verilog e utilizando máquina de estados finitos, conta com a estrutura básica de uma máquina de estados, também desenvolvido em verilog.

O projeto foi validado via simulação no Quartus II.

O Cronômetro mostra 4 dígitos, sendo 3 dígitos de segundos e 1 dígito de décimo de segundos. O cronômetro consegue contar até 999 segundos. Chegando ao final da contagem (999.9 segundos), o cronômetro reinicia do 000.0.

2.Visão geral do projeto:

- Inputs para o Clock e simulação dos botões do cronômetro(contar, pausar, resetar e parar)
- Outputs para a descrição dos estados da máquina e para a simulação do display do Cronômetro.
- Registradores para guardar os valores do estado atual e dos contadores de décimo de segundo e de segundo.
- Dois blocos always, um que representa a parte sequencial e o outro a parte combinacional.
- Estruturas de “case” que auxiliam na parte lógica do código.
- Definição “Initial” que seta os valores padrão do início do código.

Circuito projetado (Automatizado pelo Quartus):

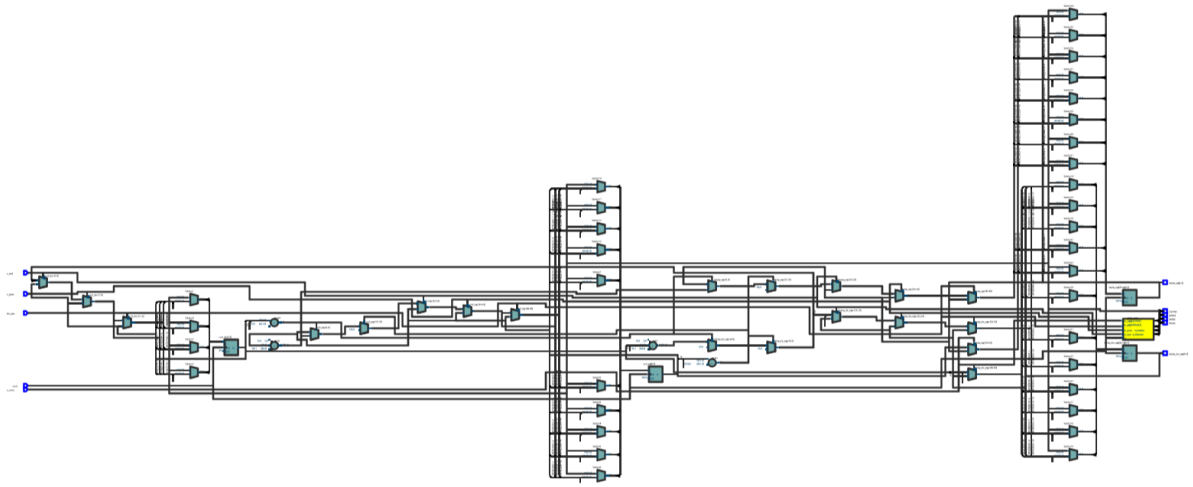
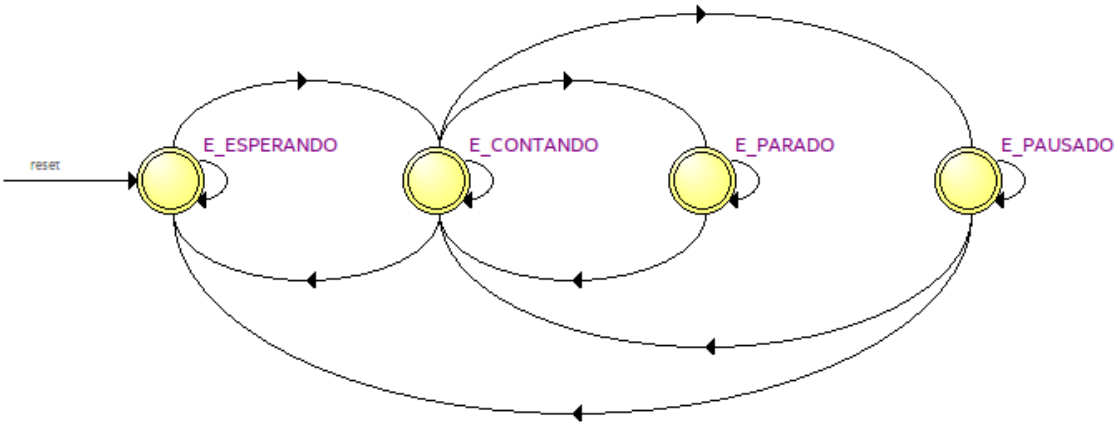
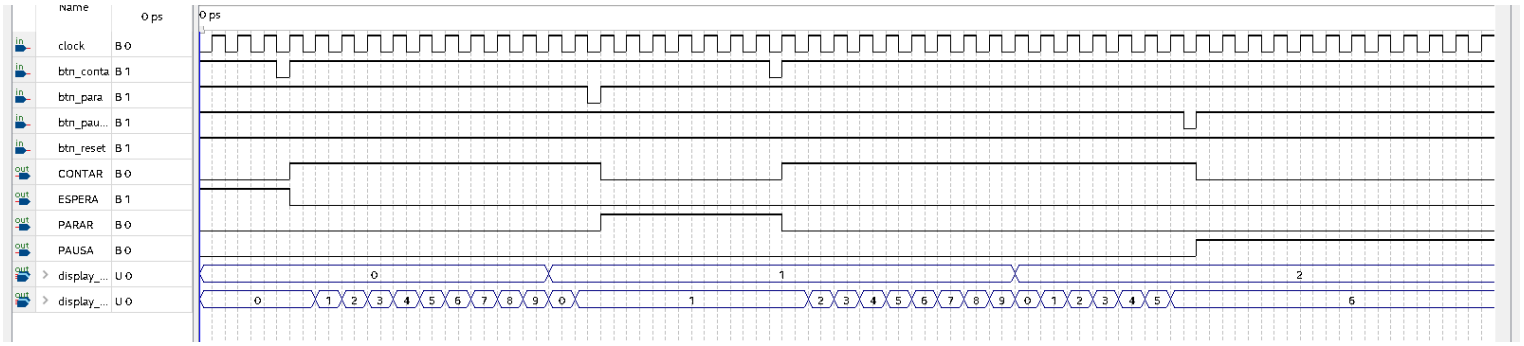
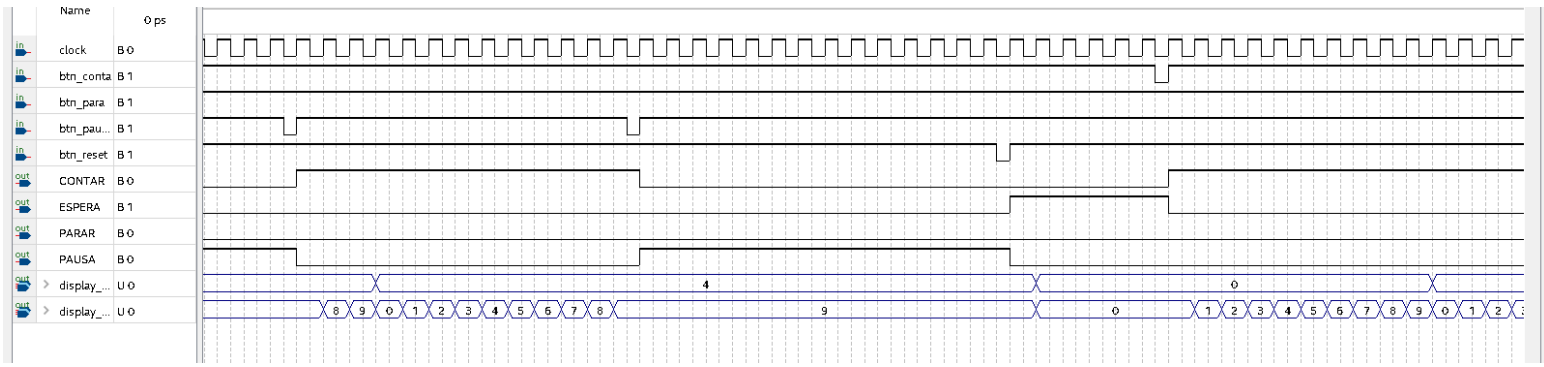


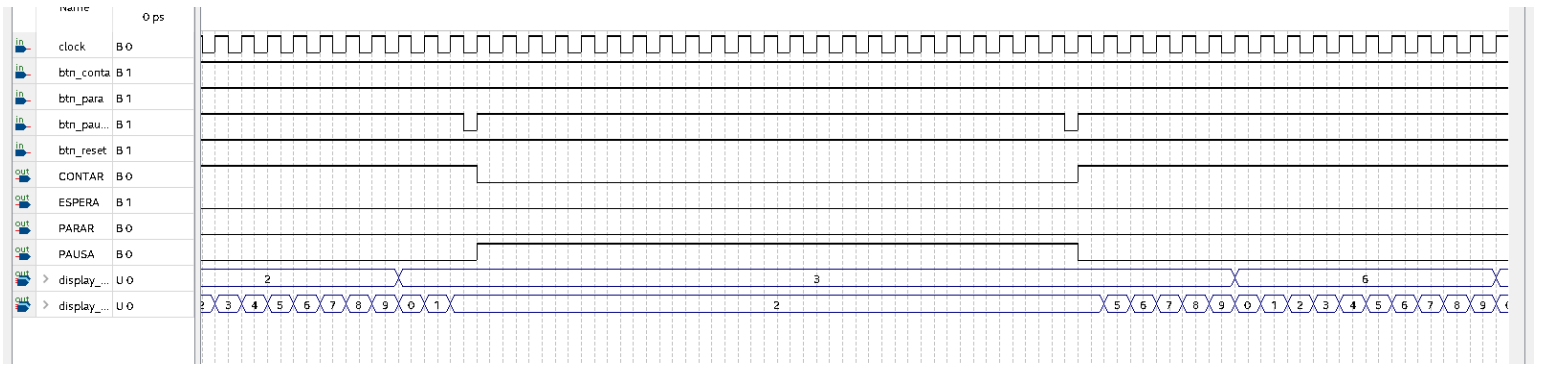
Diagrama de Estados (Automatizado pelo Quartus):

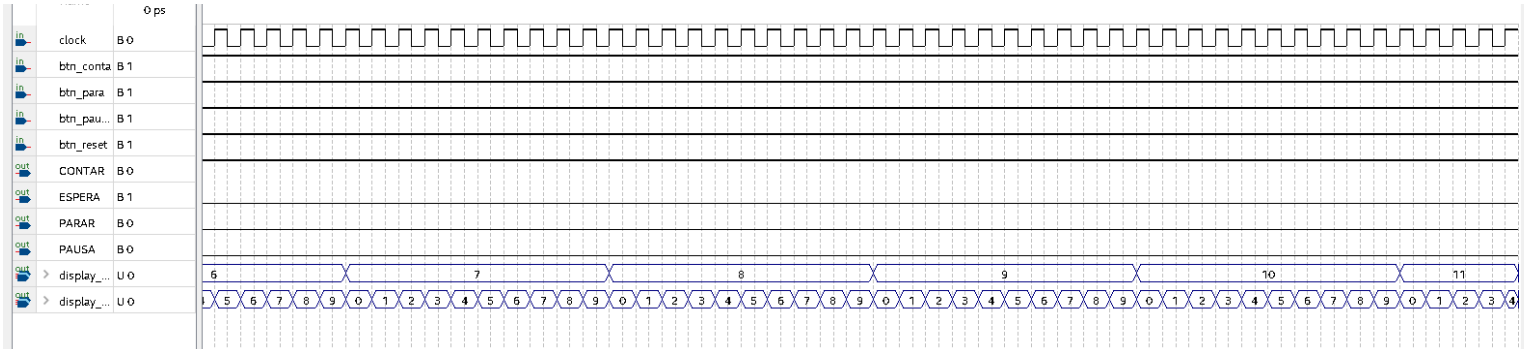


Waveforms (simulação dos botões):









3.Detalhamento de cada módulo:

A Máquina de Estados Finitos em Verilog é uma estrutura que agrega dois blocos always, que serão executados a partir das lógicas sequenciais e combinacionais.

- **Bloco Always Sequencial:**

  Esse bloco é executado sempre que ocorrem mudanças no Clock (posedge ou negedge), e consiste na parte sequencial do projeto, ou seja que executa uma parte do código (um case da estrutura "case"), de maneira contínua até que receba instruções para executar outra parte do código (outro case da estrutura "case"). Dessa forma é garantido que cada estado da máquina tenha suas especificidades bem definidas.

```
always @(posedge clock) begin
 case (estado_atual)

        E_ESPERANDO  : begin //ESTADO INICIAL DO CRONOMETRO ESPERA O
BOTAO(btn_conta) INICIAR O CRONOMETRO
                cont_dec<=0;
                cont_seg<=0;
                display_dec_segs<=0;
                display_seg<=0;

                if(btn_conta==0)begin // SE O BOTAO CONTAR ESTIVER ATIVO TROCA PARA
O ESTADO DE CONTAR
                    estado_atual <= E_CONTANDO; //TROCA DO ESTADO PARA CONTANDO
                     //CONTADOR = 0
                     // DISPLAY = 000.0
            end

        end

        E_CONTANDO  : begin
                if (btn_para==0) begin
                    estado_atual<=E_PARADO;
                end else  if (btn_pausa==0) begin
                    estado_atual<=E_PAUSADO;
                end else if (btn_reset==0) begin
                    estado_atual<=E_ESPERANDO;
                end else begin
                     if(cont_dec == 9) begin
                       cont_seg <= cont_seg + 1;
                       cont_dec <= 0;
                       display_dec_segs <= 0;
                       display_seg <= cont_seg +1;
                    end else begin
                       cont_dec <= cont_dec + 1;
                       display_dec_segs <= cont_dec + 1;
                    end
                     if(cont_seg == 999) begin
                          cont_seg <= 0;
                          cont_dec <= 0;
                          display_dec_segs <= 0;
                          display_seg <= 0;
                     end
                end
                 //CONTADOR = ++
                 // DISPLAY = CONTADOR
        end

            E_PARADO  : begin
                if (btn_conta==0) begin
                    estado_atual<=E_CONTANDO;
                end else begin
                    cont_seg<=cont_seg;
                    cont_dec<=cont_dec;
                    display_seg<=cont_seg;
                    display_dec_segs<=cont_dec;
                end
```

```
                E_PAUSADO   : begin
                    if (btn_pausa==0) begin
                       estado_atual<=E_CONTANDO;
                    end else if (btn_reset==0) begin
                       estado_atual<=E_ESPERANDO;
                    end else begin
                          if(cont_dec == 9) begin
                       cont_seg <= cont_seg + 1;
                       cont_dec <= 0;
                    end else begin
                       cont_dec <= cont_dec + 1;
                    end
                    if(cont_seg == 999) begin
                       cont_seg <= 0;
                       cont_dec <= 0;
                    end
                    end
                // LÓGICA DO CONTADOR PAUSADO (QUE CONTINUA INCREMENTANDO O REGISTRADOR
contador)
                // LÓGICA DO DISPLAY PAUSADO (QUE NAO RECEBE O VALOR DO CONTADOR NO
REGISTRADOR display,
                //APENAS RECEBE QUANDO VOLTAR A CONTAGEM COM O BOTAO CONTAR(btn_contar))
                        //CONTADOR = ++
                        // DISPLAY = PAUSADO
                end
            default: ;
        endcase;
    end
```

- **Bloco Always Combinacional:**

  Esse bloco é executado sempre que ocorrer alguma alteração no âmbito dos botões. Ele gera como saída a partir dos inputs (botões) um sinal que mostra qual/quais estado(s) está/estão ativo(s) . Por exemplo, ao pressionar o botão de "CONTAR", o bloco Combinacional é executado e ao checar o registrador "estado_atual" (que guarda informação sobre qual estado está ativo), executa o trecho de código do case relacionado ao estado de "CONTAR" e gera os sinais de saída CONTAR=1,ESPERA=0,PAUSA=0,PARAR=0.

```
always @(posedge btn_conta,posedge btn_reset,posedge btn_pausa,posedge btn_para) begin
    case (estado_atual)
        E_ESPERANDO : begin
            CONTAR=0;
            PARAR=0;
            PAUSA=0;
            ESPERA=1;
        end
        E_CONTANDO : begin
            CONTAR=1;
            PARAR=0;
            PAUSA=0;
            ESPERA=0;
        end
        E_PARADO : begin
            CONTAR=0;
            PARAR=1;
            PAUSA=0;
            ESPERA=0;
        end
        E_PAUSADO : begin
            CONTAR=0;
            PARAR=0;
            PAUSA=1;
            ESPERA=0;
        end
         default: ;
    endcase;
end
```

4.Conclusão:

O desenvolvimento desse projeto foi de fundamental importância para consolidação do conhecimento adquirido durante a segunda metade da disciplina de Sistemas Digitais, nos dando a visão e a responsabilidade de um projetista e também nos exigindo aplicação direta de tudo aquilo visto em aula. A integração e o trabalho em grupo também gerou uma certa experiência, pois nos mostrou que mesmo em tempos difíceis, é possível se desenvolver em conjunto.