|  |
| --- |
| **Disciplina:** PCS 3335 – Laboratório Digital A |
| **Prof.:** *Anarosa* **Data:** *03/04/2020* |
| **Turma:** *01*  **Bancada:** *A2* |
| **Membros:** |
| *Andre Devay Torres Gomes (10770089)* |
| *FIZ O EXPERIMENTO SOZINHO* |



## Introdução

Esta experiência traz o conceito das máquinas de estados descritas em VHDL e como utilizá-las em circuitos digitais. A ideia é conseguirmos desenvolver sistemas digitais mais complexos, compostos por fluxo de dados e unidade de controle.

## Objetivo

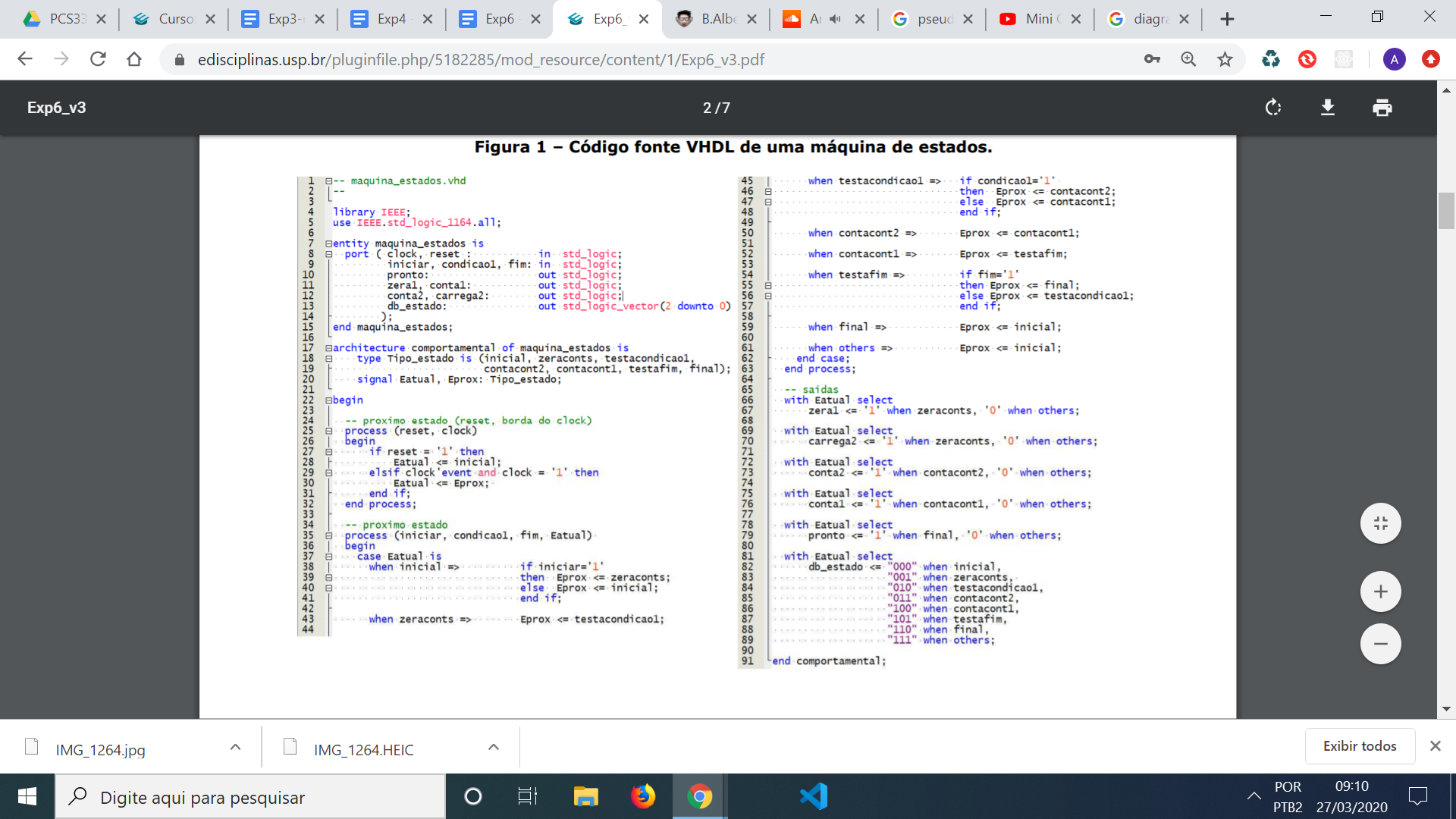
O intuito desta experiência é que nós consigamos desenvolver e entender a descrição de máquinas de estados em VHDL e a aplicação de máquinas de estados como unidade de controle de um circuito digital, tudo isso com circuitos mais simples.

## Planejamento

**Projeto de uma Máquina de Estados em VHDL**

1. Projeto
2. Descrição funcional (e respectivo Pseudocódigo)

Acordando com o tema do experimento 6 (Máquinas de Estados em VHDL), foi dado para nós o código descrito em VHDL descrito abaixo:



Ao analisarmos tal código, podemos perceber que ele é um circuito sequencial (pela maneira de estruturação de seus ‘process’) e, olhando mais a fundo, reconhecemos que se trata de uma máquina de estados de Moore (pois seu estado está relacionado com a saída).

Conseguimos ver também que há três ‘inputs’ utilizados como condicionais para nossa máquina de estado (‘iniciar’; ‘condicao1’ e ‘fim’) e que há 7 estados em nosso sistema.

Podemos, portanto, descrever o programa em VHDL acima por um pseudocódigo que demonstra bem sua funcionalidade:

**Programa ‘maquina-estados.vhd’**

Início:

A qualquer momento, se reset = 1, então o estado atual se torna ‘inicial’.

Senão, se clock estiver em subida, o próximo estado se tornará o estado atual.

//A respeito dos estados atuais e futuros:

Quando o estado atual for ‘inicial’:

‘db\_estado’ é = ‘000’.

Se ‘iniciar’ = 1, então próximo estado é ‘zeraconts’

senão próximo estado é inicial.

Quando o estado atual for ‘zeraconts’:

‘db\_estado’ é = ‘001’.

O próximo estado será ‘testacondicao1’.

Quando o estado atual for ‘testacondicao1’:

‘db\_estado’ é = ‘010’.

Se ‘condicao1’ = 1, então próximo estado é ‘contacont2’.

senão próximo estado é ‘contacont1’.

Quando o estado atual for ‘contacont2’:

‘db\_estado’ é = ‘011’.

O próximo estado é ‘contacont1’.

Quando o estado atual for ‘contacont1’:

‘db\_estado’ é = ‘100’.

O próximo estado é ‘testafim’.

Quando o estado atual for ‘testafim’:

‘db\_estado’ é = ‘101’.

Se ‘fim’ = 1, então próximo estado é ‘final’

senão próximo estado é ‘testacondicao1’’.

Quando o estado atual for ‘final’:

‘db\_estado’ é = ‘110’.

O próximo estado é ‘inicial’.

Quando o estado atual for outro diferente dos estados citados acima:

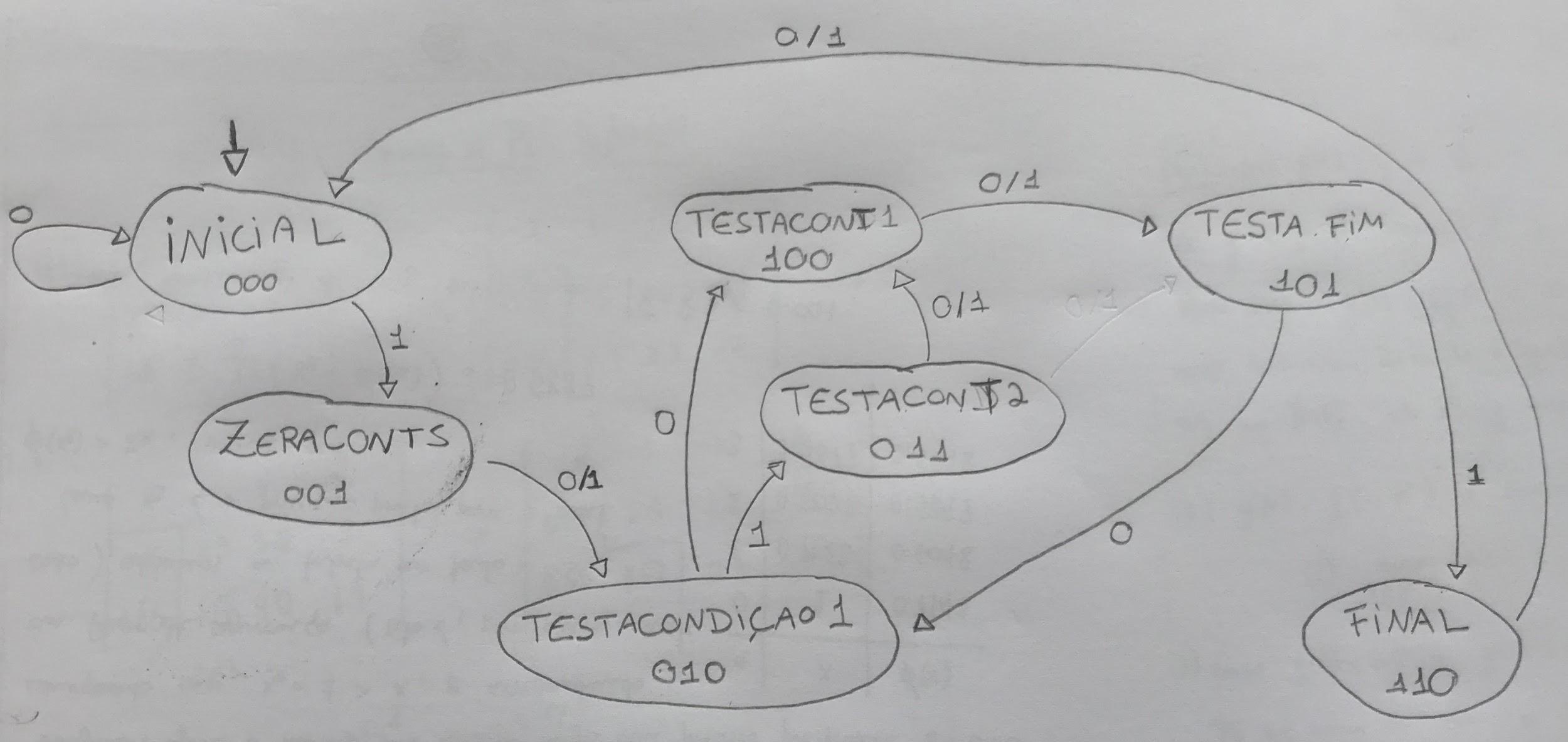
‘db\_estado’ é = ‘111’.

O próximo estado é ‘inicial’.

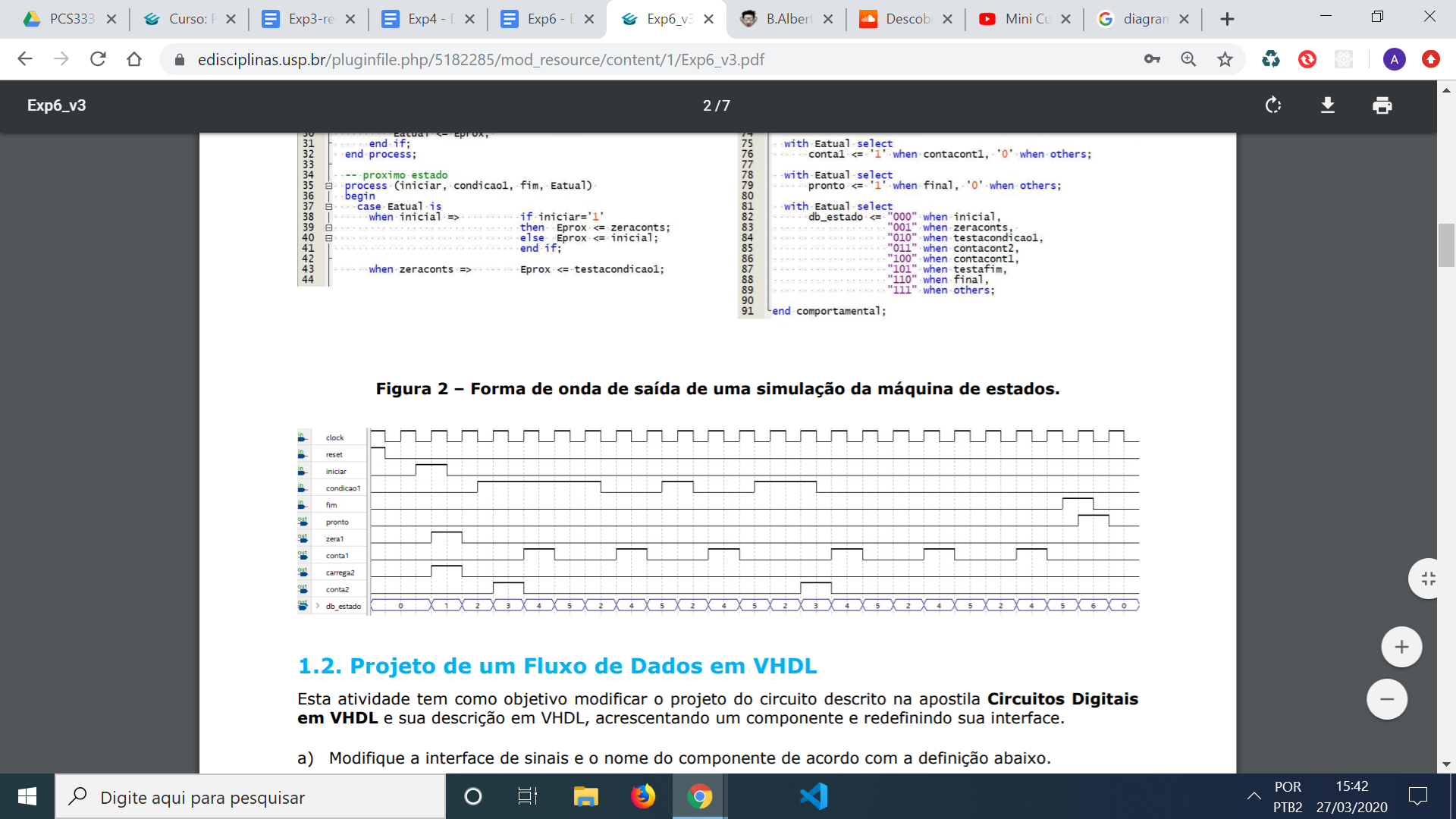
Fim

1. Diagrama de Transição de Estados

A partir dos estudos citados acima com o código fornecido, conseguimos ver que há uma similaridade da maneira em que o código foi escrito com o exemplo de Moore com dois process no documento do professor Bruno Albertini [1] e, dessa forma, conseguimos desenvolver um diagrama de transição de estados para tal código.



1. Simulação base

****

1. Simulações novo projeto no Intel Quartus Prime

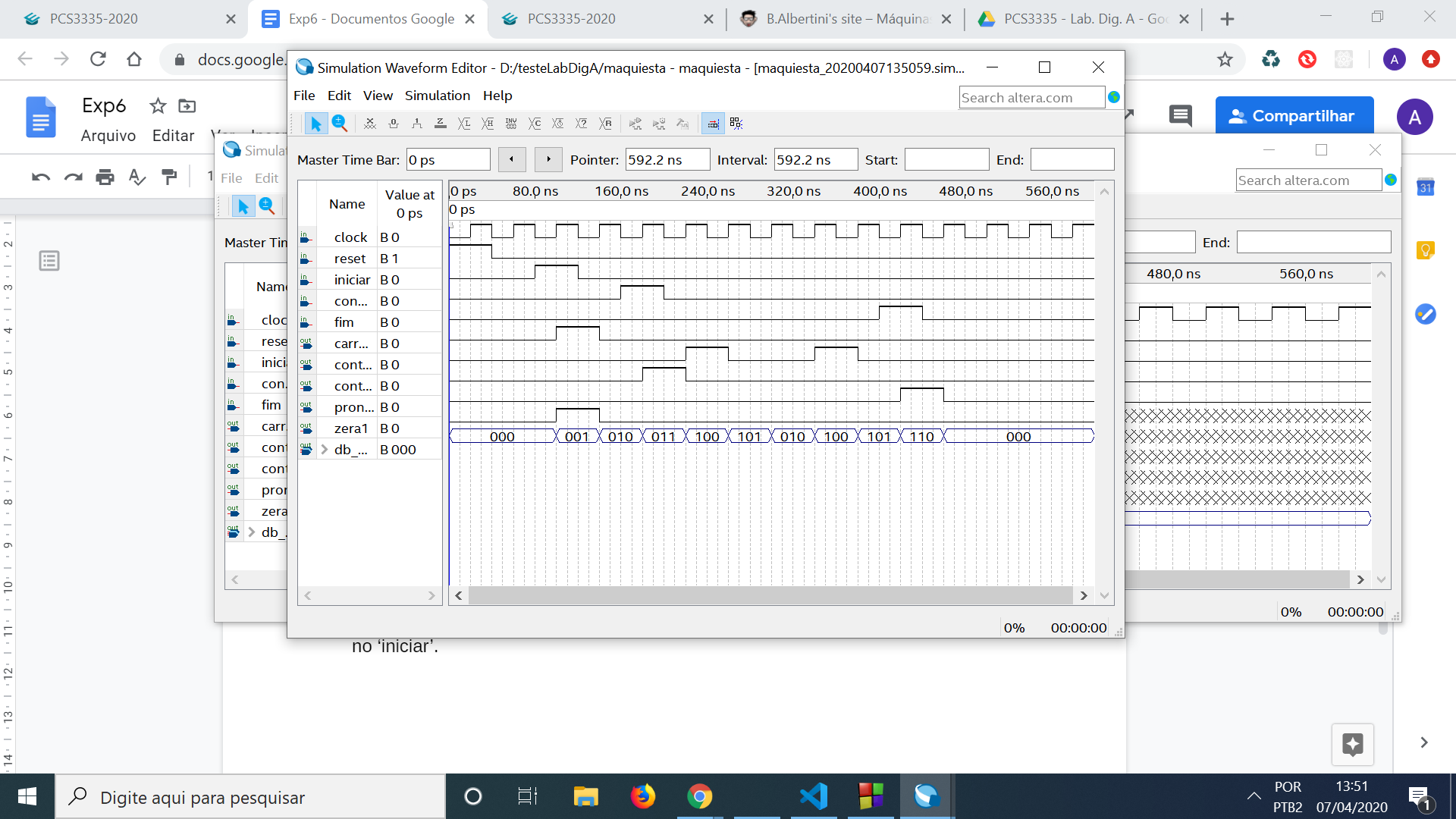
Após criarmos um projeto dentro do Quartus e reescrever o código mostrado na imagem acima, iremos fazer um plano de testes para simular os sinais de onda da saída e compreender se tudo ocorre como o esperado.

Primeiramente, ajustamos o ‘clock’ (com um ciclo de 40ns), setamos um end time para 0.6 micro segundos (tempo suficiente para o nosso teste) e colocamos o ‘clear’ em ‘1’ para garantir que nosso circuito não tem resquícios que possa atrapalhar nosso teste.

Em seguida, vamos introduzir o estado inicial aplicando a condição ‘0’ e ‘1’ e verificando se nosso programa se mantém no estado ‘inicial’ e vai para o estado ‘zeraconts’, respectivamente.

Logo, passaremos para o estado ‘testacondicao1’ e testaremos sua passagem para o estado ‘testacont1’. Iremos ao estado ‘testafim’ e colocaremos o input ‘1’ para ele retornar ao estado ‘testacondicao1’.

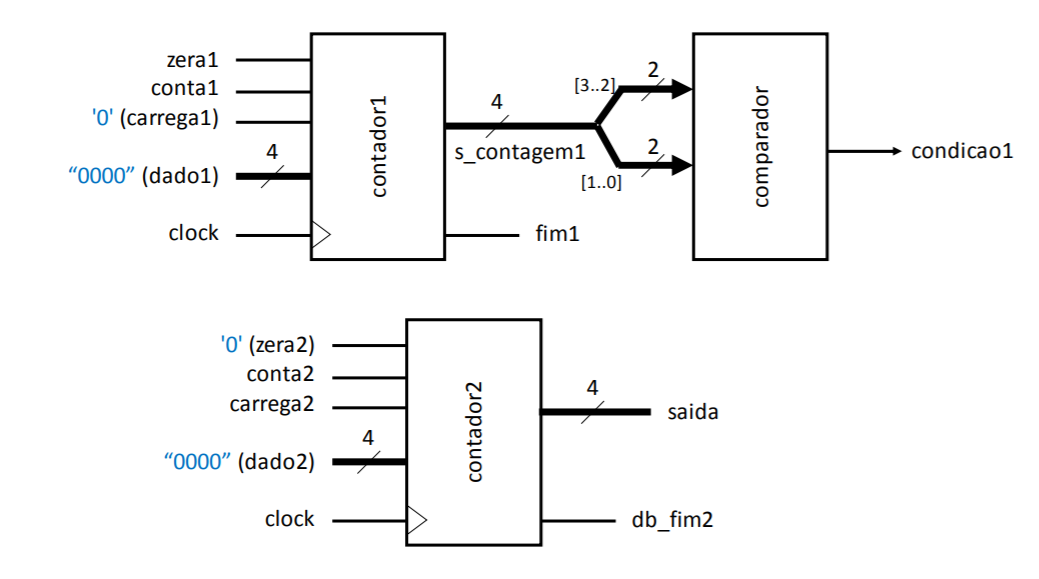
Após isso, iremos por outro caminho ao ‘testafim’, passando pelo ‘testacont2’ seguido do estado ‘testacont1’. Para terminar a nossa simulação, saíremos do ‘testafim’ para o estado ‘final’ e tentaremos retornar ao estado ‘inicial’. É perceptível que deixamos um tempo além do necessário no estado inicial para entender se ele continuaria lá (o que era desejado) até a indicação do input ‘1’ no ‘iniciar’.



**Projeto de um Fluxo de Dados em VHDL**

1. Descrição funcional

Nesse projeto foi pedido para modificarmos o circuito digital do Experimento 05 para que conseguíssemos montar um componente de fluxo de dados. Basicamente, o novo circuito será uma combinatória dos projetos desenvolvidos no experimento anterior, portanto, temos o seguinte diagrama de blocos:



Como podemos perceber na esquematização acima o fluxo de dados irá ativar a ‘condicao1’, quando os dois primeiros dígitos do vetor de contagem for igual ao dois últimos (em ‘0000’; ‘0101’; ‘1010’; ‘1111’), vale ressaltar que como fixamos a entrada do contador com o ‘carrega1’ zerado, o comparador atingirá a saída ‘1’ de forma periódica, pois não conseguiremos carregar valores (quebramos essa periodicidade somente se zerarmos).

Para o contador 2, percebemos que seu funcionamento é basicamente o mesmo do contador original. Contudo, esse não consegue ser zerado através do botão ‘zera2’, podemos fazer o mesmo efeito de zerar nesse contador através da ativação do ‘carrega2’, uma vez que o ‘dado2’ está fixado em ‘0000’.

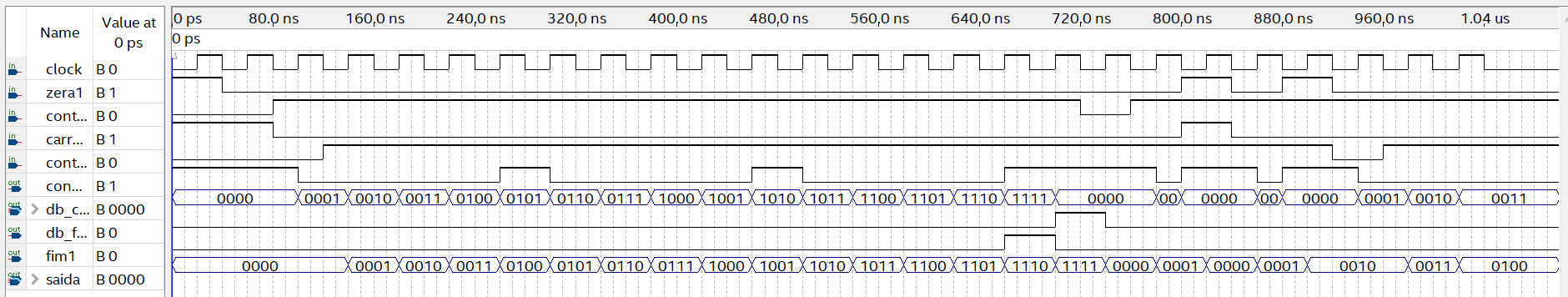
1. Plano de Testes e Simulação no Quartus

Após criarmos um novo projeto dentro do Quartus e ajustar o código dos sub componentes, iremos fazer um plano de testes para simular os sinais de onda da saída e compreender se tudo ocorre como o esperado.

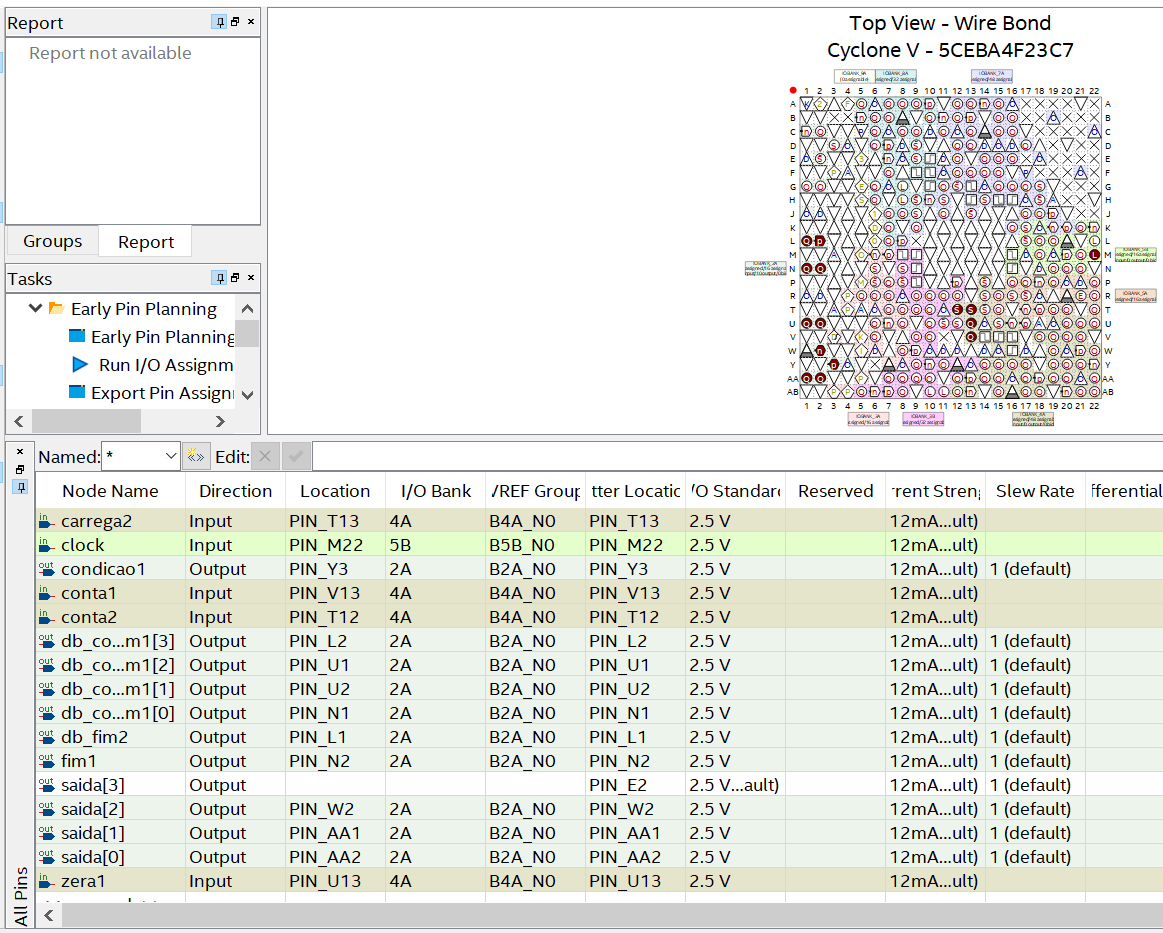
Primeiramente, ajustamos o ‘clock’ (com um ciclo de 40ns), setamos um end time para 1.1 micro segundos (tempo suficiente para o nosso teste) e colocamos o ‘zera1’ e ‘carrega2’ em ‘1’ para garantir que nosso circuito não tem resquícios que possa atrapalhar nosso teste (aplicamos a idéia similar ao reset comum).

Em seguida, vamos testar a funcionalidade do ‘contador1’ + ‘comparador’ e ver se eles funcionam corretamente um ciclo completo mostrando as saídas esperadas, faremos uma pausa na ativação do ‘conta1’ em alguns momentos e ver se o projeto responde ao que queríamos. Após um ciclo e meio, podemos testar se o zerar funciona corretamente.

Concomitantemente com o teste do ‘contador1’ + ‘comparador’, testaremos o ‘contador2’ em seu ciclo completo. Por fim, no meio do segundo ciclo, testaremos a pausa na ativação do ‘conta2’ e seu método de zerar (ativando o ‘carrega2’).



1. Pinagem placa FPGA



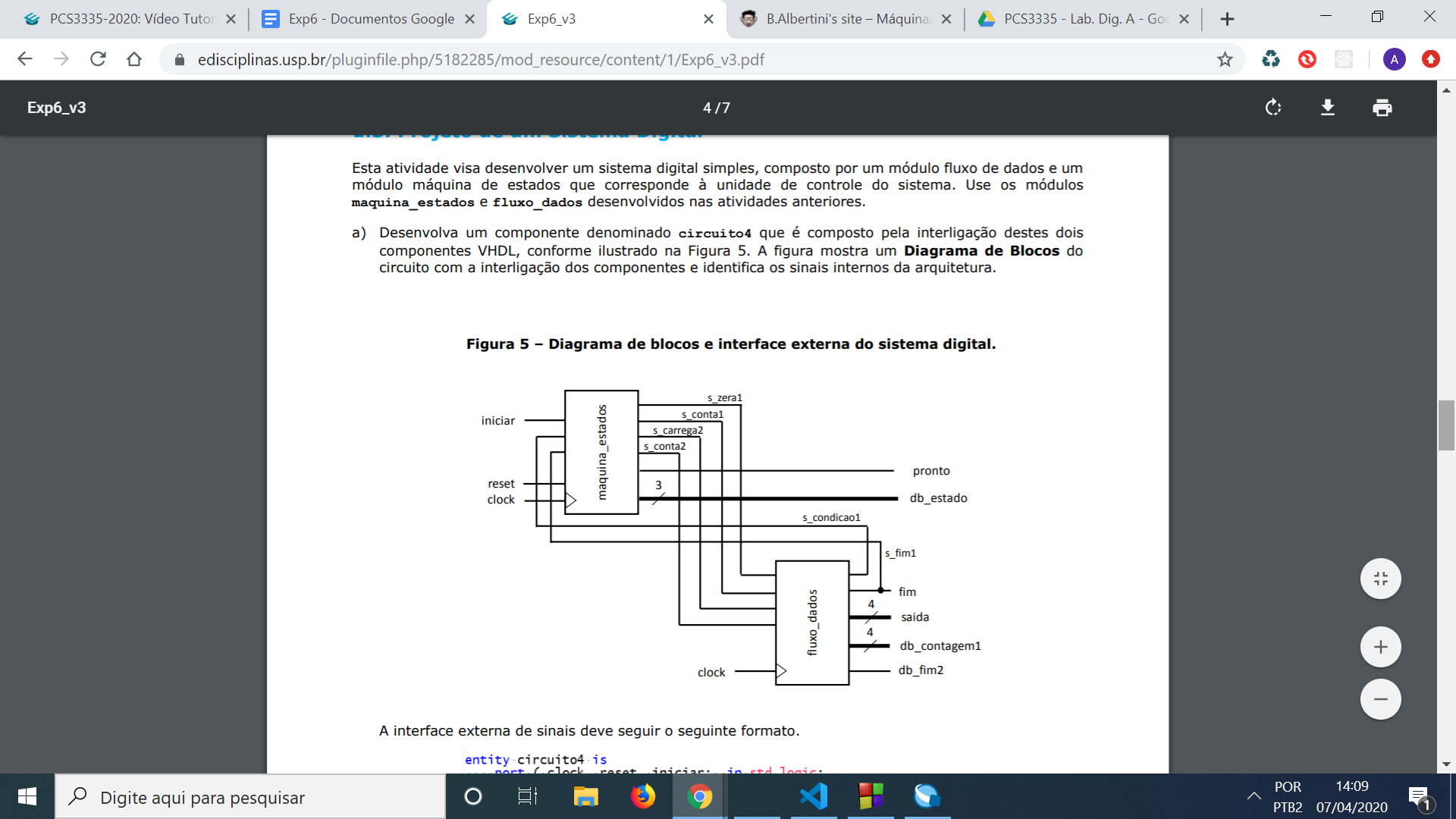
**Projeto de um Sistema Digital**

1. Projeto
2. Descrição Funcional

O ‘circuito4’ é composto pela interligação dos dois outros componentes VHDL feitos neste relatório (a máquina de estados + o fluxo de dados) e sua função é exatamente exemplificar como a maioria dos sistemas são construídos: com 1 unidade de controle + um local de passagem de dados.

1. Diagrama de Blocos

Neste projeto foi pedido para combinarmos o circuito do fluxo de dados com o componente da máquina de estados para que conseguíssemos montar um projeto completo de UC + Dados. Basicamente, o novo circuito será uma combinatória dos projetos desenvolvidos nesse experimento, portanto, temos o seguinte diagrama de blocos:

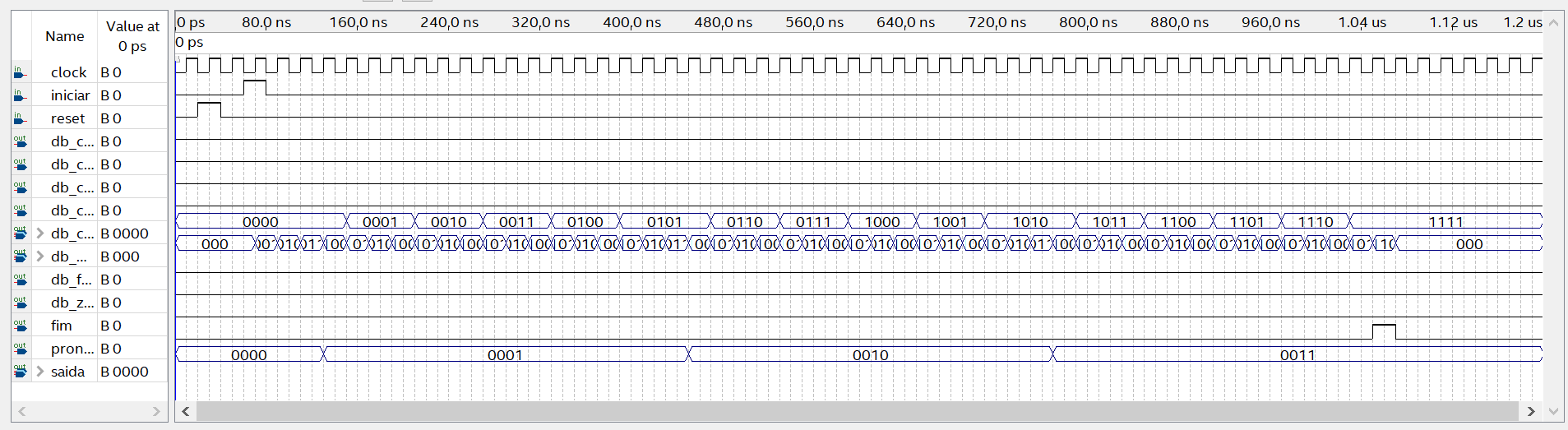


1. Plano de Testes e Simulação no Quartus

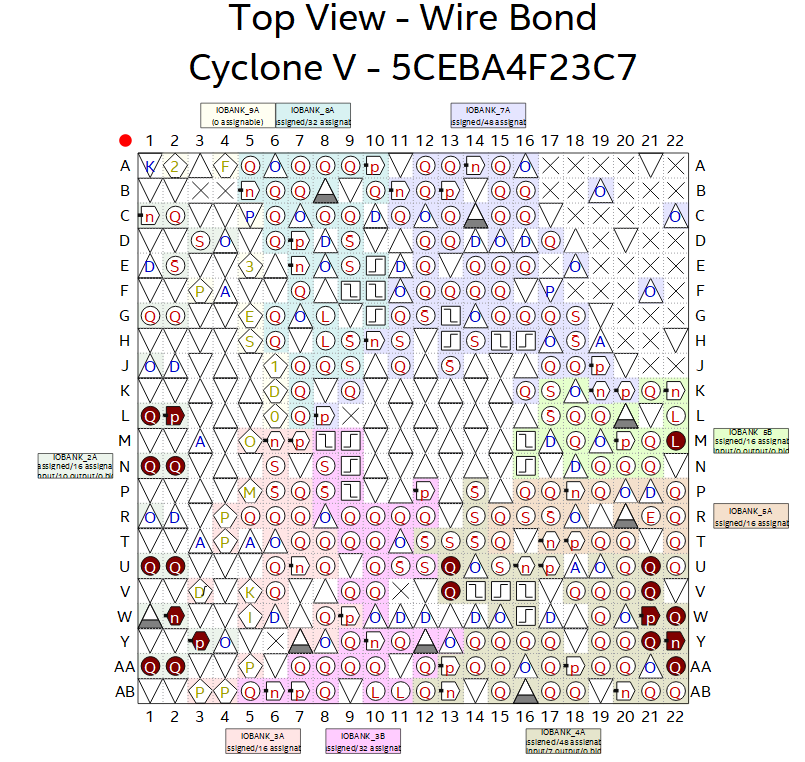
Após criarmos um último projeto dentro do Quartus e juntar o código do fluxo de dados e da máquina de estados, iremos fazer um plano de testes para simular os sinais de onda da saída e compreender se tudo ocorre como o esperado.

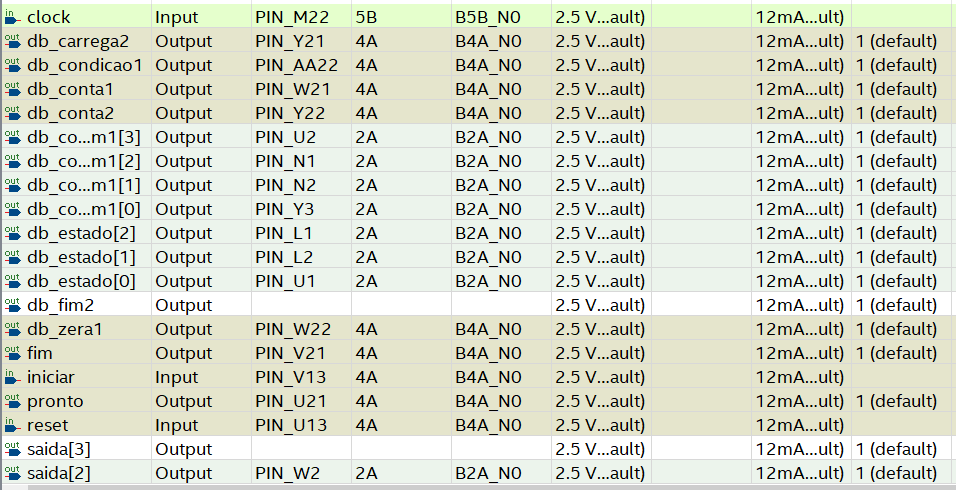
Primeiramente, ajustamos o ‘clock’ (com um ciclo de 20ns), setamos um end time para 1.2 micro segundos (tempo suficiente para o nosso teste) e colocamos o ‘reset’ em ‘1’ para garantir que nosso circuito não tem resquícios que possa atrapalhar nosso teste. Em seguida, aplicamos o input ‘iniciar’ e, como podemos ver, o sistema ocorre como o esperado nas etapas seguintes.

Podemos perceber que este projeto tem uma funcionalidade praticamente igual ao descrito nos outros experimentos. A ideia é que teremos a mesma tomada de decisão vista nas máquinas de estados e podemos acompanhar, através do output ‘db\_estado’, em qual estado o programa está. O output ‘pronto’ é acionado quando se tem passado pelo estado ‘FINAL’ , assim, o programador pode saber quando o seu ciclo nas máquinas de estados se finaliza somente olhando uma saída. Vale ressaltar também que caso não se acione o botão ‘iniciar’ novamente após o ‘pronto’, o sistema seguirá no estado inicial e, portanto, estará em stand-by.



1. Pinagem na placa FPGA

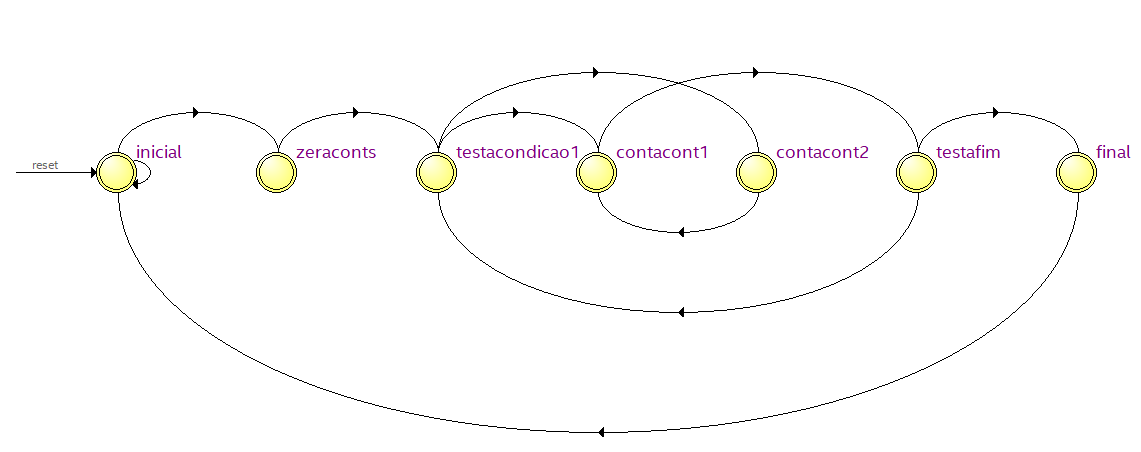


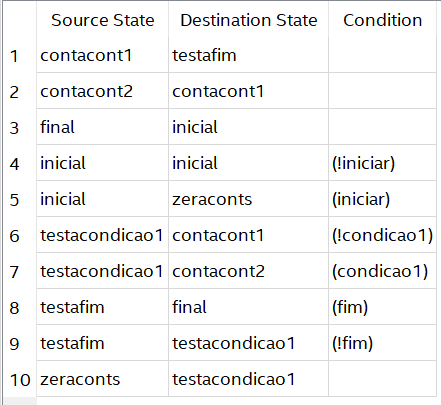


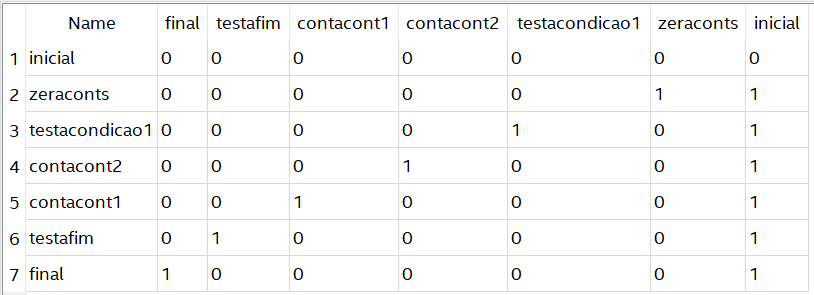


1. **Relatório**
2. State Machine Viewer para o ‘maquina\_estados’

Como podemos ver a nossa máquina de estados está de acordo com o que planejamos para o seu funcionamento.

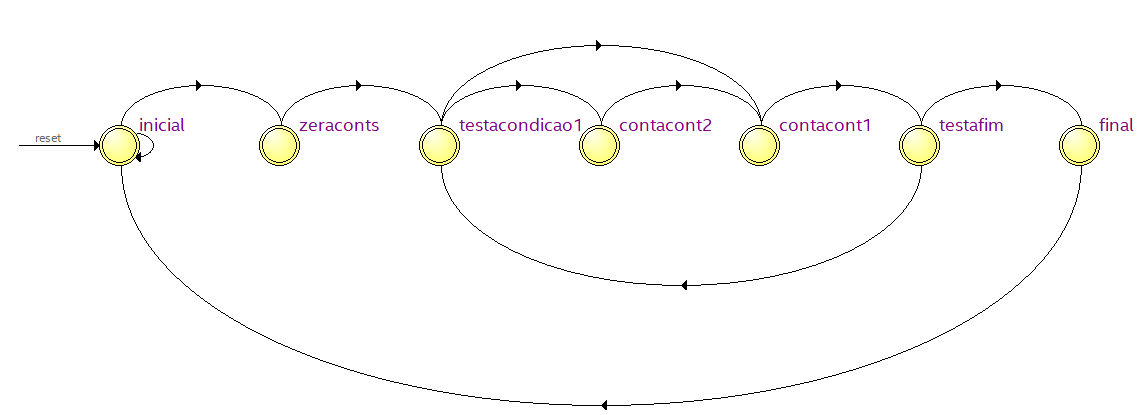


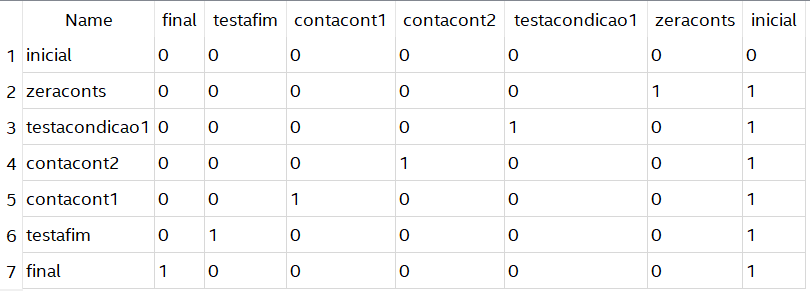
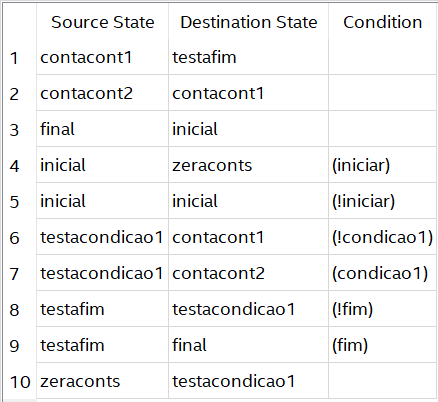




1. State Machine Viewer para o ‘circuito4’

Como podemos ver a nossa máquina de estados está de acordo com o que planejamos para o seu funcionamento.

****

****

## Apêndices

**Referências**

1. Apostilas e documentos de apoio do site ~/labdig do PCS.
2. Apostilas disponíveis na plataforma e-Disciplinas.

[1] <https://balbertini.github.io/vhdl_fsm-pt_BR.html>