Trabalho prático 2 - Urna eletrônica em verilog

Arthur De Bellis (3503), Leandro Lázaro (3513), Mateus Pinto (3489), Pablo Ferreira (3480)

Campus Florestal – Universidade Federal de Viçosa (UFV-F)
Rodovia LMG 818, km 06, s/n – Bairro do Lago, Florestal – MG, 35690-000

{arthur.bellis, leandro.lazaro, mateus.p.silva, pablo.ferreira}@ufv.br

Resumo. *Este trabalho descreve uma urna eletrônica completa em verilog, com botão de corrige, confirma, e um enter (chamado de valid). O código permite fácil alteração dos candidatos, permitindo colocar quantos forem necessários e/ou mudar seus números. Foram criados: um modulo para ser o processador da urna, um para gerar informações do display em BCD, vários contadores BCD em série, um transformador de BCD para display de sete segmentos e um demultiplexador para gerar informação de voto válido. Também criamos um módulo de clock para fins de simulação de ondas e um redutor de clock para adaptar a urna para qualquer tecnologia. Para isso, foi implementado usando um modelo de Máquina de Estados de Meanly síncrona e depois essa urna foi implementada na tecnlogia Intel FPGA, modelo Altera DE2-115.*

1. Introdução

Este trabalho prático tem como objetivo a implementação de uma urna eletrônica capaz de computar votos para candidatos usando o número de matrícula como número eleitoral, com botão de corrige, confirma, um *enter* e capacidade de computar até 999999 votos (acreditamos ser mais do que o necessário).

Para padronizar, usamos todos os valores guardados como BCD (binário convertido para decimal).

2. Visão Geral

A urna tem seu funcionamento divido em pequenos módulos. Ela apresenta um processador chamado “urna.v” que emite pulsos para cada voto computado, e esses pulsos são recebidos por vários contadores ligados em série chamados “contadorbcd.v”, os quais guardam quantos votos cada candidato tem.

Para criar o display da urna, incorporamos um modulo chamado “displayModule.v” que tem como entrada a saída de todos os contadores - isto é, a quantidade de votos de cada candidato - e a dos números já digitados - que vem do processador. Ao fim, codificamos um conversor que transforma um número BCD para a entrada de um display de sete segmentos chamado “bcd7convert.v”.

3. Módulos e seus funcionamentos

Descreveremos resumidamente o funcionamento de cada módulo e suas características. Todos, exceto o de clock para a simulação, são sintetizáveis para diferentes tecnologias.

3.1. Processador - “urna.v”

O processador “urna.v” é uma máquina de estados de Meanly com reset síncrono que apresenta seis estados que são alterados com o *posedge* (borda de subida) da entrada valid, que funciona como uma espécie de *“enter”* da urna.

Os quatro primeiros estados são os necessários para inserir os quatro digitos de cada candidato, e a saída da transição desses para qualquer outro estado é 0 para qualquer candidato. O quinto estado foi apelidado de “aguardando confirma”, e ele computa o voto usando uma estrutura *SwitchCase* (uma forma encurtada da estrutura “se, então”) que compara os números digitados dos estados anteriores concatenados com os guardados na urna que mostram qual número é de cada candidato. Dependendo da entrada, a saída pode ser 1 para algum candidato (incluindo o candidato “Nulo”, que acontece caso o usuário vote num número eleitoral inexistente na memória da urna).

Também foi necessário implementar um estado apelidado “resetando”, que zera as saídas na urna para evitar qualquer tipo de problema de voto duplicado.

3.2. Contador BCD - “contadorbcd.v”

O contador BCD funciona somando um a cada pulso da entrada do seu clock. Caso o seu valor passe de nove, ele gera um pulso na sua saída “overflow”, o que indica que o número armazenado não cabe mais em apenas em algum algarismo, então ele zera seu valor e soma um número na próxima casa de algarismos (guardada por outro contador), transformando 10, que nao cabe em apenas um algarismo, em 1 na casa decimal + 0 na casa das unidades. O contador apresenta reset assíncrono.

3.3. Módulo de display - “displayModule.v”

A lógica do módulo de display se divide em duas, uma quando Finish está desligado, isto é, quando as votações ainda estão ocorrendo, e uma quando Finish está ligado, ou seja, as eleições acabaram e o usuário deseja ver os resultados. Todos os resultados são síncronos com o clock.

Durante as eleições, o módulo de display se baseia no estado do processador para colocar nas saídas os números já digitados pelo eleitor. Já depois de encerradas as eleições, o módulo de display mostra cada candidato por um número de identificação seguido dos seus votos (para um candidato de número 2 com 5 votos, por exemplo, é exibido C5000002). Caso sejam mostrados os votos nulos, é mostrado “CN” seguido dos votos. Para exibir os outros, basta modificar a entrada *escolhaCandidato.*

3.4. Conversor de BCD para display de sete segmentos - “bcd7convert.v”

Este módulo converte um número BCD para a entrada de um display de sete segmentos sincronizadamente com o clock. Cada display necessita de um módulo diferente, e isso permite que sejam usados quantos displays forem necessários, fazendo com que nossa urna seja usada desde eleições de pequeno, médio até grande porte. Além dos

convencionais algarismos de 0 à 9, usamos um algarismo teórico 10 para desligar o display, 11 para criar o C (usado para mostrar o número de identificação do candidato) e o 12 para mostar um N (de nulo).

3.5. Redutor de clock - “divClock.v”

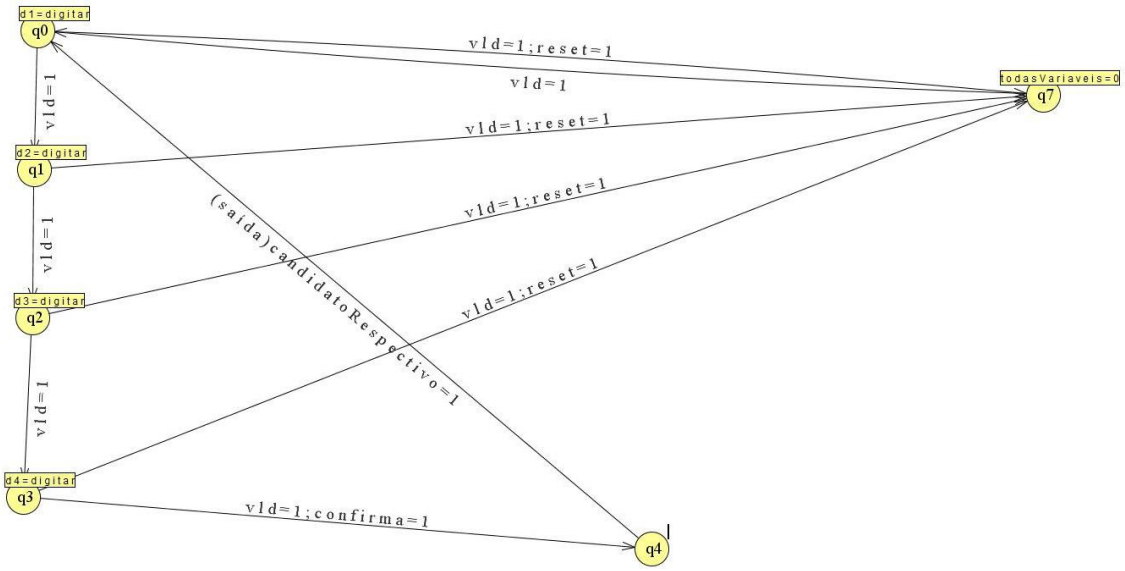
Este módulo faz com o que clock da tecnologia a ser implementada seja reduzido a fim de poder executar os outros módulos de forma síncrona sem problemas. Ele funciona contando bordas de subidas para criar uma borda de subida no novo clock. Exemplificando, caso sejam necessários contar cinco bordas de subida para gerar uma borda de subida no novo clock, o clock antigo fica cinco vezes menor.

3.6. Clock para a simulação de ondas - “clock1unit.v”

Este módulo puramente usado para a simualação gera um clock de 1hz. É apenas usado para criar as simualações de onda no computador, e não é sintetizável para nenhuma tecnologia.

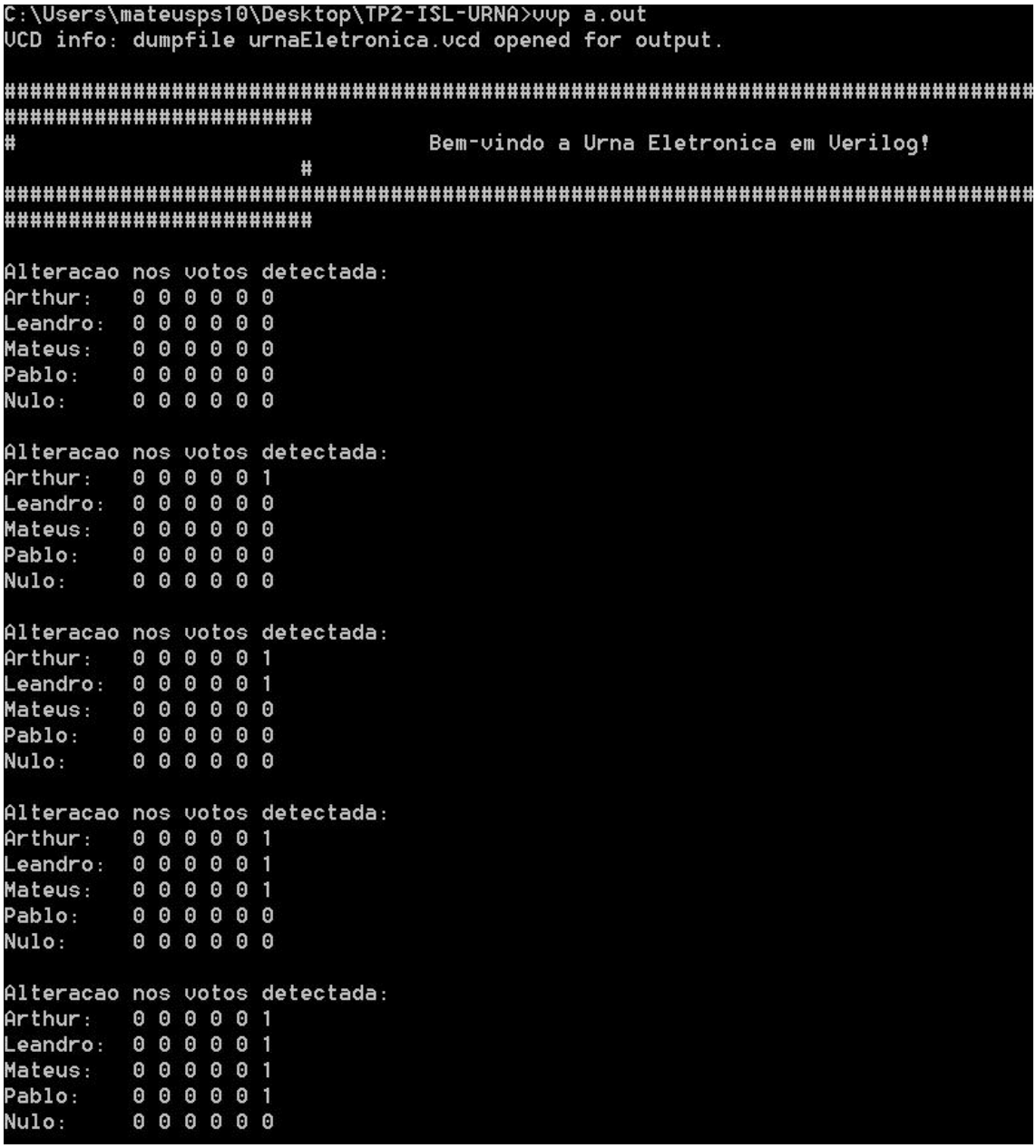
3. Diagrama de estados de Meanly

4.



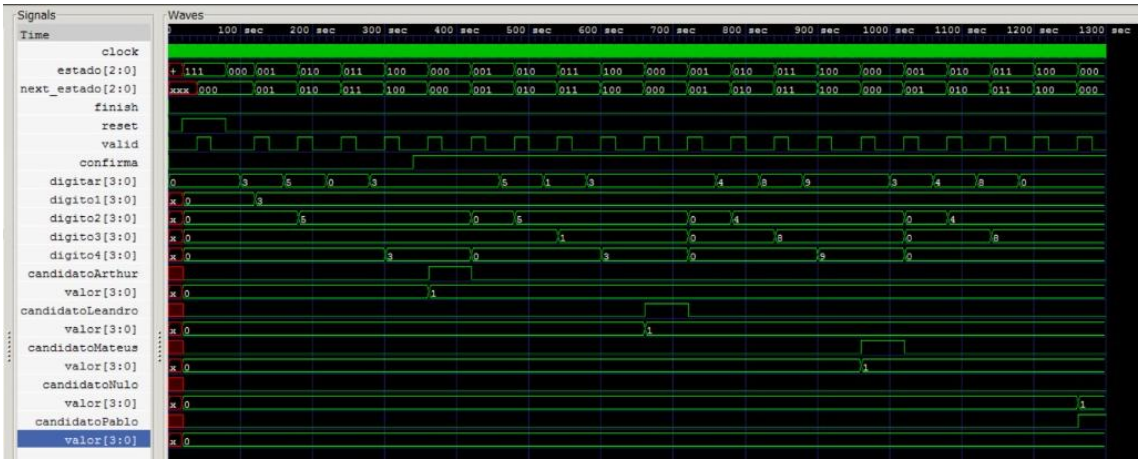
Os estados q0, q1, q2, q3 são, respectivamente, *“Aguardando dígito 1, 2, 3 e 4”*. O estado q4 é o *“Aguardando confirma”,* que cria uma borda de subida para o candidato votado ou para os votos nulos e o q7 é o estado *“Resetando”,* que torna todas as variáveis iguais a 0.

4. Simulação em linha de comando



Nesta simulação, foram feitos quatro votos, um para o candidato Arthur, outro para Mateus, Leandro e ao final para Pablo.

4.1. Simulação em ondas



Para fins de diminuição do número de ondas, foram usados apenas cinco contadores BCDs, um para cada candidato incluindo o nulo, que também foi retirado da imagem com a mesma finalidade.

5. Implementação na tecnologia Intel FPGA

Criamos, então, um projeto para a tecnologia Intel FPGA cujo arquivo principal se chama “urnatp.qpf”. Nela, usamos como base, conforme solicitado, a placa Altera DE2-115 e não tivemos muitas dificuldades nessa implementação, visto que fizemos todos os códigos visando a capacidade de sintetizá-los.

5.1. Altera DE2-115

Usamos os switches de 17 a 14 para serem o número a ser digitado (em código binário). O SW0 é o reset, 2 o confirma, 3 o corrige, 4 o valid, 10 o finish e de 13 a 11 o escolhaCandidato.

Todos os displays de sete segmentos da placa foram utilizados, e os Leds verdes de 7 a 5 representam o próximo estado, 4 um voto válido. Os leds vermelhos de 17 a 15 mostram o estado atual e o 13 mostra que o voto foi nulo.