Roteiro do Projeto 2: Supervisão de Percurso de Pedágio Aberto

1.Introdução

O objetivo do projeto 2 do **Laboratório de Sistemas Digitais II** é desenvolver um sistema digital de Supervisão de Percurso de Pedágio Aberto, também conhecido como pedágio por quilômetro rodado. O aluno deve desenvolver um projeto utilizando a estrutura ***RTL*** proposta, descrevendo o projeto em linguagem **VHDL** e implementando-o em um **FPGA**.

Esse laboratório também tem como objetivo exercitar o desenvolvimento de projetos digitais em ***RTL*** e a sua realização utilizando as ferramentas de engenharia estudadas em teoria. Esse processo envolve a compreensão do problema, seu planejamento, desenvolvimento da solução lógica, integração dos subsistemas, implementação no ambiente computacional, simulação, testes, depuração do projeto, implementação física e registro dos resultados.

Sistema Ponto a Ponto (ARTESP, 2021)

Fonte: Divulgação/Artesp

2.Planejamento do Projeto

Aula 10 – Entendimento do Problema (AVA):
- Discussão do roteiro do projeto e definição de características de projeto específicas para cada grupo;
- Entendimento do projeto lógico do sistema digital (elementos do Fluxo de Dados em ***VHDL***)
- Elaboração do **diagrama da máquina de estados de alto nível (*FSMD*)**; (**Avaliação Formativa**)

Aula 11 – Desenvolvimento da Unidade de Controle (AVA):
- Desenvolvimento da lógica da ***UC*** (diagrama de estados da ***FSM***) e Tabela de Sinais de Controle do ***FD***;
- Desenvolvimento do projeto lógico do sistema digital (diagrama em ***RTL*** e diagrama de estados);
- **Apresentação do diagrama de estados da *UC* (*FSM*)** e da **Tabela de Sinais de Controle do Fluxo de Dados** ao professor; (**Avaliação Formativa**)

Aula 12 – Configuração dos Módulos em VHDL (AVA):
- Implementação da ***UC*** e do ***FD*** em componentes descritos em ***VHDL*;**
- **Apresentação da descrição dos componentes *UC* e *FD* em *VHDL*** ao professor; (**1 ponto**)

Aula 13 – Simulação Funcional do Projeto (AVA):
- **Apresentação da simulação funcional da *UC+FD*** ao professor; (**3 pontos**)
- Inclusão dos decodificadores e associação de pinos para configuração do ***FPGA***.

Aula 14 – Configuração do FPGA (CGI) e Entrega do Relatório Final:
- **Configuração do *FPGA*** e teste projeto na placa **DE10-Lite;** (**se for possível acesso ao CGI**)
- **Apresentação do projeto** com a implementação no ***FPGA***; (**1 ponto***)
- **Entrega do relatório do projeto** (conforme instruções no final deste do roteiro). (**2 pontos**)
  Observação: Caso não haja possibilidade de aula presencial a pontuação da configuração do FPGA será incorporada na avaliação do relatório (que passará a valer 3 pontos).

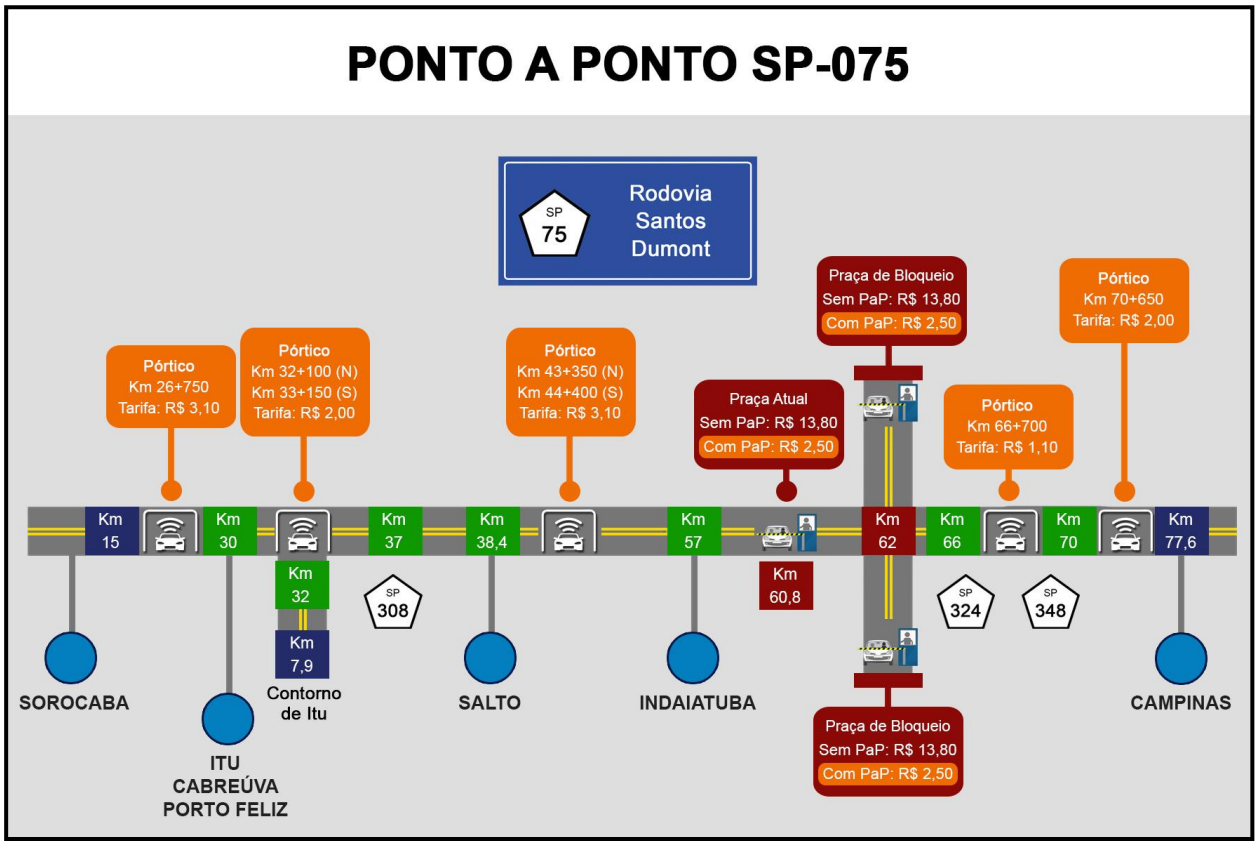
3.Descrição Funcional do Projeto

Criado em 2012 pelo Governo do Estado de São Paulo, o Sistema Ponto a Ponto consiste em uma nova forma de cobrança de pedágio nas rodovias paulistas, feita de forma eletrônica e com base no trecho percorrido pelo usuário. Esse sistema também é conhecido como **Sistema de Rodovia de Pedágio Aberto** ou "***free-flow***" (CNT, 2021).

O sistema de pedágio aberto consiste na instalação de dispositivos "Tags" (componentes passivos que respondem às ondas eletromagnéticas emitidas pelas antenas, emitindo uma identificação por radiofrequência - RFID) nos veículos que permite o monitoramento dos carros nas estradas e cidades. Esse sistema tem o objetivo de prevenir, fiscalizar e reprimir o furto e o roubo de veículos e cargas, bem como contribuir para uma cobrança mais justa de pedágio (pagamento por distância percorrida pelo usuário), reduzindo as distorções do modelo de cobrança com praças de pedágio. Além disso, melhora as condições operacionais das rodovias, proporcionando fluidez, além de reduzir o tempo das viagens, consumo de combustível e a emissão de gases poluentes na atmosfera Em São Paulo, os chips já estão sendo usados em algumas estradas privatizadas do interior do estado, onde pórticos fixos são instalados em pontos estrategicamente definidos nas rodovias.

Em 2012, o projeto foi instituído de forma experimental na Rodovia Engenheiro Constâncio Cintra (SP-360) e na Rodovia Santos Dumont (SP-75). Em 2013, a Rodovia Governador Adhemar Pereira de Barros (SP-340) recebeu o primeiro Ponto a Ponto aberto para qualquer usuário e veículo que trafegue naquela rodovia. E em 2014, a Rodovia Professor Zeferino Vaz (SP-332) recebeu o quarto Ponto a Ponto do Estado de São Paulo. Na **Figura 1** são apresentadas a localização e as tarifas cobradas em janeiro de 2021 na Rodovia Santos Dumont (SP-75), segundo a Agência Reguladora de Serviços Públicos Delegados de Transporte do Estado de São Paulo (ARTESP, 2021).

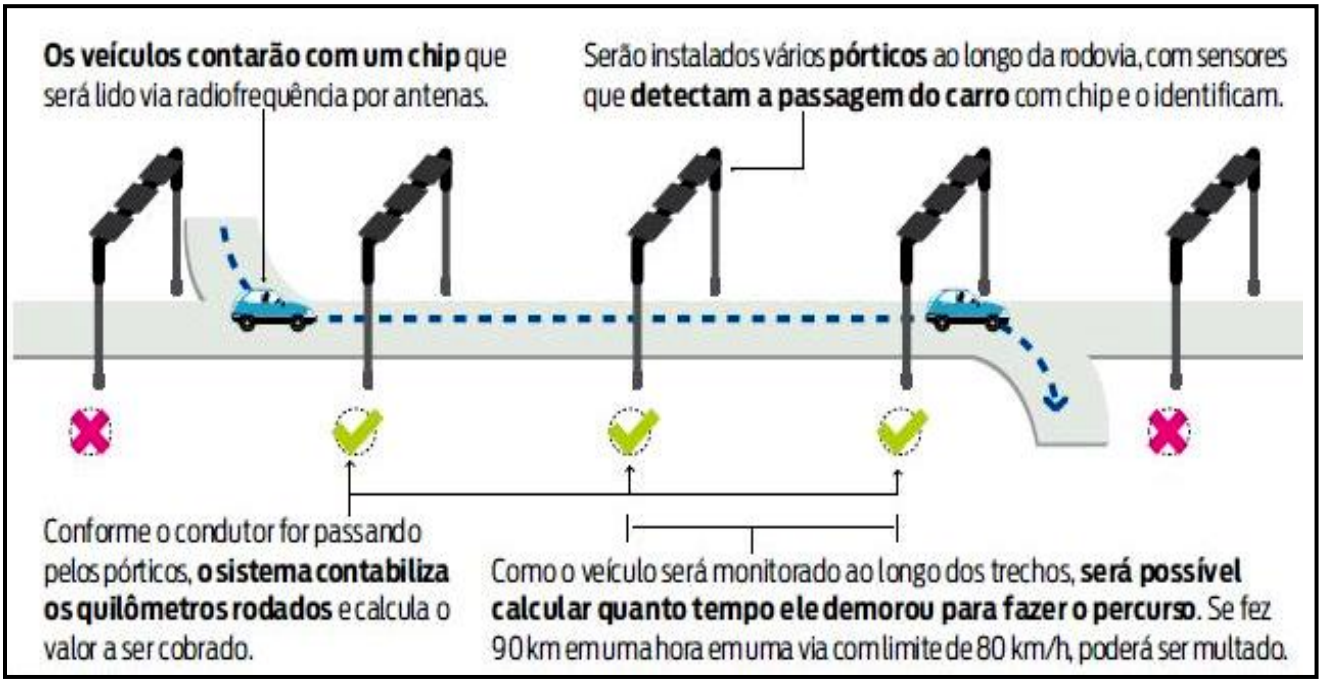
Figura 1: Localização de pórticos e tarifas na SP-75 para o Sistema Ponto a Ponto (Janeiro/2021).



Fonte: ARTESP, 2021.

Nos pórticos há antenas e leitores que detectam os dispositivos RFID afixados nos veículos pelas Operadoras de Serviço de Arrecadação autorizadas a operar no Estado de São Paulo. A cobrança de pedágio é eletrônica no sistema pré-pago. Ao passar por um pórtico, as antenas e leitores reconhecem o dispositivo instalado no veículo, recebem um código de identificação, e o valor da tarifa é automaticamente debitado dos créditos que aquele usuário possui com a sua operadora de pedágio eletrônico. Na **Figura 2** é apresentado um diagrama esquemático simplificado da operação de um Sistema de Pedágio Aberto (GAZETA, 2021).

Figura 2: Diagrama esquemático de operação de um Sistema de Pedágio Aberto

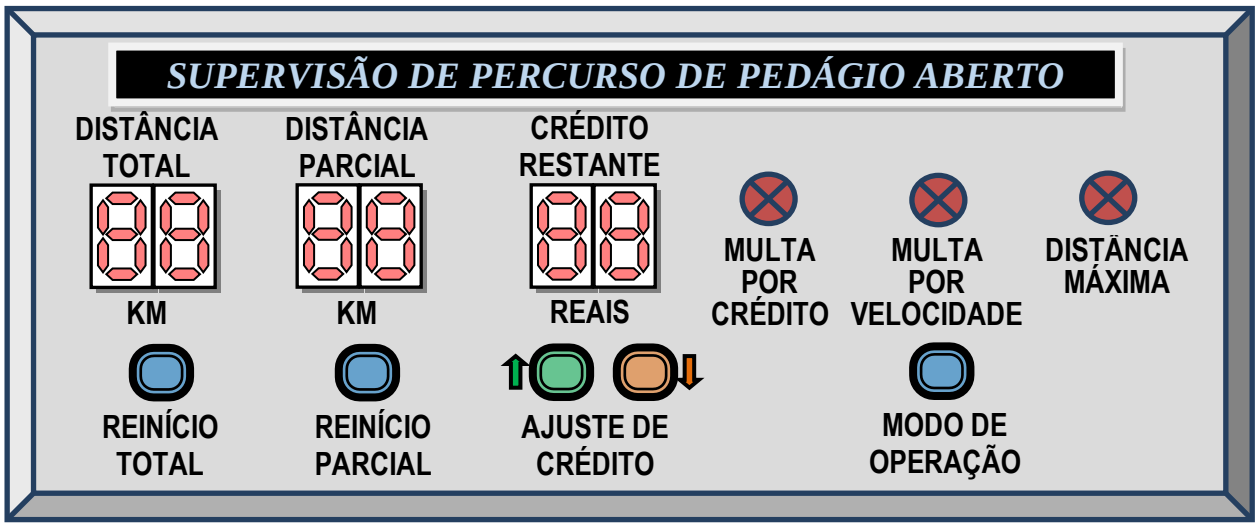


Fonte: GAZETA, 2021.

Para o projeto do laboratório consideramos que o sinal de identificação emitido pelo Tag RFID do veículo, ao receber o sinal das antenas, pode ser captado por um sistema de controle no interior do veículo, sendo identificado o sinal de passagem pelo pórtico (**PP**). O objetivo do projeto deste laboratório é implementar um sistema de Supervisão de Percurso de Pedágio Aberto para ser instalado no interior veículo, que identifica o sinal de passagem pelo pórtico (**PP**) e informa ao motorista: o total de créditos disponíveis, a distância parcial percorrida na rodovia (desde o último reinício parcial), a distância total percorrida, se ele atingiu um limite de distância máxima percorrida (para controle da quilometragem que o veículo pode percorrer) e se está sujeito a dois tipos de multa (por falta de créditos ou por excesso de velocidade).

O sistema de Supervisão de Percurso de Pedágio Aberto pode possuir um painel similar ao esquematizado na **Figura 3**. Esse painel apresenta: o crédito restante (**CR**), a contagem da distância total percorrida (**DT**), a distância parcial percorrida na rodovia (**DP**), um led de sinalização de multa por falta de crédito (**MFC**), um led de multa por excesso de velocidade (**MEV**) e um led para sinalizar um limite de distância máxima percorrida (**SDM**). Esse painel possui três chaves e dois botões: a chave de reinício total (**RT)** que recarrega o valor do crédito inicial, reinicia todos os acumuladores e apaga todos os leds; a chave de reinício parcial (**RP**) que reinicia apenas o contador de distância parcial (**DP**) e apaga apenas o led de sinalização de multa por excesso de velocidade (**MEV**); a chave de modo de operação (**MO**) que define se o sistema está no modo de carga de créditos ou em operação normal; o botão de o botão de incrementa crédito (**IC**) que aumenta o valor do crédito restante cada vez que é acionado; e o botão de decrementa crédito (**DC**) que reduz o crédito restante cada vez que é acionado.

Figura 3: Esquema do painel do sistema de Supervisão de Percurso de Pedágio Aberto



Fonte: Autor.

Quando o veículo passa por um pórtico o sistema identifica um pulso de passagem por pórtico (**PP**). Considere que os pórticos estão instalados a cada **1 Km** ao longo da rodovia. Se a passagem por dois pórticos demorar menos que certo intervalo de tempo deve ser sinalizada a possibilidade de multa por excesso de velocidade (por exemplo: se a distância entre dois pórticos for percorrida em menos que 30 segundos a velocidade é maior que 120 Km/h).

Quando o sistema de supervisão está no **modo de operação normal** (chave **MO** em nível lógico um) podem ser apresentados os valores das distâncias percorridas e do crédito restante assim que o veículo passar pelo pórtico. A multa por crédito deve ser sinalizada se ocorrer a passagem por um pórtico e o valor de crédito restante for inferior ao valor da tarifa. Neste caso o sistema deve sinalizar a multa por falta de crédito (**MFC**) e deve zerar o valor do crédito restante (não deve ser informado crédito negativo).

O sistema de supervisão possui um valor de crédito inicial (**CRI**) que é carregado quando é acionada a chave de reinício total (**RT**). Se o sistema estiver no **modo de carga de valores** (chave **MO** em nível lógico zero) esse valor de crédito inicial pode ser reajustado pelo usuário com os botões de incrementa (**IC**) e decrementa (**DC**). Nesse modo de operação, a cada acionamento do botão incrementa (ou decrementa) o crédito restante pode ser somado (ou subtraído) de um valor de ajuste de crédito (**AC**). O valor do crédito restante pode ser ajustado entre os limites **0** e um valor de crédito máximo (**CRM)**. Caso o incremento de valor exceda o valor **CRM** a lógica de controle deve limitá-lo a esse valor. Analogamente, o decremento do valor do crédito abaixo de zero deve ser impedido pela lógica de controle, limitando o valor do crédito restante ao valor mínimo zero. Observa-se que o ajuste de crédito restante só pode ser realizado depois que o sistema de supervisão for totalmente reiniciado e enquanto a chave **MO** estiver em nível lógico zero. Assim que o sistema entrar no **modo de operação normal** (chave **MO** em nível lógico um) não é mais possível o ajuste do crédito restante. Para realizar novo ajuste nos valores de crédito restante o sistema de supervisão deve ser totalmente reiniciado (**RT**) e a chave de modo de operação (**MO**) deve estar em nível lógico zero.

4.Parâmetros Individuais (por grupo de alunos)

O projeto deve ser desenvolvido em **grupo de no máximo dois alunos**. Cada grupo deve considerar a definição dos parâmetros associados ao dígito de controle dos dois alunos (**DA** e **DB**) indicados na **Tabela 1**. Onde **DA** é o dígito de controle do aluno que possuir o maior número de matrícula e **DB** é o dígito de controle do aluno com o menor número de matrícula.

Para esse projeto são definidos cinco parâmetros:

- Valor de crédito inicial (**CRI**): valor que deve carregado no reinício total do sistema;
- Valor de crédito máximo (**CRM**): valor do crédito máximo que o sistema aceita;
- Valor de ajuste de crédito (**AC**): valor a ser incrementado ou decrementado no crédito a cada acionamento dos botões **IC** e **DC**;
- Valor da tarifa de pedágio (**TRP**): valor a ser decrementado do crédito a cada passagem por um pórtico;
- Valor do limite máximo de distância (**LDM**): valor da distância percorrida para sinalização de limite.

A associação dos parâmetros do projeto com os dígitos de controle (**DA** e **DB**) é especificada na **Tabela 1.**

Tabela 1: Associação das variáveis de projeto com os dígitos de controle do grupo de alunos.

DA	CRI (R$)	CRM (R$)	DB	AC (R$)	TRP (R$/KM)	LDM (KM)
0	2	15	0	5	1	10
1	3	20	1	4	1	11
2	4	25	2	3	2	12
3	5	30	3	2	2	13
4	6	35	4	5	3	14
5	7	40	5	4	3	15
6	8	45	6	3	4	16
7	9	50	7	2	4	17
8	10	55	8	4	5	18
9	11	60	9	6	5	19

Fonte: Autor.

5.Entendimento do Projeto (Aula 10 no AVA)

Cada grupo de alunos deve desenvolver um **Diagrama da Máquina Estados de Alto Nível (*FSMD*)** representando todas as operações do sistema de Supervisão de Percurso de Pedágio Aberto. Os estados desse diagrama devem representar a sequência lógica com as operações que serão realizadas com os valores de crédito (ajuste de crédito com botões, decremento de crédito por pulso de pórtico), a verificação de multas (por velocidade e falta de crédito) e as operações de reinício total e parcial do sistema. Esse diagrama deve representar as funções executadas pelos sinais de entrada do sistema (**RT, RP, MO, IC**, **DC**, **PP**).

6.Requisitos de Desenvolvimento do Projeto (Aula 11 no AVA)

Cada grupo de alunos deve desenvolver um **Diagrama de Estados (*FSM*)** para a ***máquina de estados da Unidade de Controle (UC)*** e a **Tabela de Sinais de Controle do Fluxo de Dados**.

Os estados da ***FSM*** devem representar apenas as operações lógicas da máquina de estados. Para obter esse diagrama devem ser estudados os elementos do fluxo de dados e, a partir das operações previstas no ***FSMD***, devem ser identificados os estados lógicos necessários na Unidade de Controle.

A **Tabela de Sinais de Controle do Fluxo de Dados** representa, para cada estado da ***FSM***, a condição dos sinais são enviados aos elementos do Fluxo de Dados para realizar as operações previstas no sistema.

O **Diagrama de Estados da Unidade de Controle (*FSM*)** e a **Tabela de Sinais de Controle do Fluxo de Dados** devem ser **apresentados ao professor**.

7.Requisitos de Implementação do Projeto em VHDL (Aula 12 no AVA)

O sistema de Supervisão de Percurso de Pedágio Aberto deve ser totalmente implementado utilizando o **FPGA** da família **MAX 10** (modelo: **10M50DAF484C7G**) da placa de desenvolvimento **DE10-Lite.**

O projeto deve ser desenvolvido utilizando o método de projeto baseado na transferência de dados entre registradores, ou projeto ***RTL*** (*Register Transfer Level*), o qual consiste em representar o sistema digital através de blocos operacionais do fluxo de dados (***FD***) e uma unidade de controle (***UC***). A lógica da unidade de controle pode ser especificada através de uma máquina de estados finitos (***FSM*** – *Finite State Machine*).

Para maiores detalhes consulte a referência [2] da disciplina: Vahid, F. – Sistemas Digitais: projeto, otimização e HDLs, páginas 242 a 264 (VAHID, 2008).

Cada grupo de alunos deve desenvolver **individualmente**, uma unidade de controle (***UC***) e um conjunto de elementos que compõem o fluxo de dados (***FD***).

Esse sistema de Supervisão de Percurso de Pedágio Aberto deve atender aos seguintes requisitos:

- O sistema deve ser implementado utilizando a estrutura de fluxo de dados (***FD***) especificada e **utilizando os componentes do fluxo de dados** (***FD***) **fornecidos no Moodle**. São fornecidos os códigos ***VHDL*** de uma Unidade de Aritmética e Lógica para operações com oito bits (**ULA8.vhd**), de um multiplexador de oito entradas (**MUX8.vhd**), de um registrador de oito bits (**REG8.vhd**) e de um temporizador de um segundo (**TIMER_1S.vhd**). O **fluxo de dados (*FD*) deve ser estruturado como indicado na Figura 10**, sendo que a interligação dos sinais nas entradas dos multiplexadores pode ser realizada da forma que o aluno desejar (exceto nas primeiras três entradas do **MUX_B**). Os códigos ***VHDL*** desses componentes **não podem ser modificados**;
- A descrição de arquitetura em ***VHDL*** **deve ser realizada utilizando-se o formato de fluxo de dados** e **o formato comportamental**. A interligação dos elementos **deve ser realizada utilizando-se o formato estrutural** (representando os códigos ***VHDL*** **em componentes**). O aluno pode utilizar os componentes já desenvolvidos no laboratório (como seletores, decodificadores e Flip-Flops) além dos componentes já fornecidos no Moodle para o projeto do sistema;
- Cada grupo deve desenvolver uma **Máquina de Estados de Alto Nível (*FSMD*)** e um **Diagrama de Estados** para a ***UC*** **(*FSM*)**. Para a descrição ***VHDL*** do ***diagrama de estados da UC*** é **fornecido no Moodle** um código de referência (*template*) em **VHDL** (**UC_SPA_Exemplo.vhd**). O objetivo desse código é servir de referência para o desenvolvimento da máquina de estados dados da Unidade de Controle do projeto do aluno, conforme apresentado no **Anexo 2** deste roteiro;
- Também deve ser detalhada a **Tabela de Sinais de Controle do Fluxo de Dados** (conforme visto em teoria), de modo a definir as características dos sinais da interface entre a ***UC*** e o ***FD***, conforme representado no diagrama de blocos da **Figura 9**;
- O **Clock** do sistema deve ser o sinal de frequência 50 MHZ, já disponível na placa de desenvolvimento **DE10-Lite** (sinal **MAX10_CLK1_50** no pino **P11**). O sistema deve ser completamente síncrono com esse clock, ou seja, **todos os blocos síncronos** devem ser sincronizados com o sinal de clock de 50 MHz;

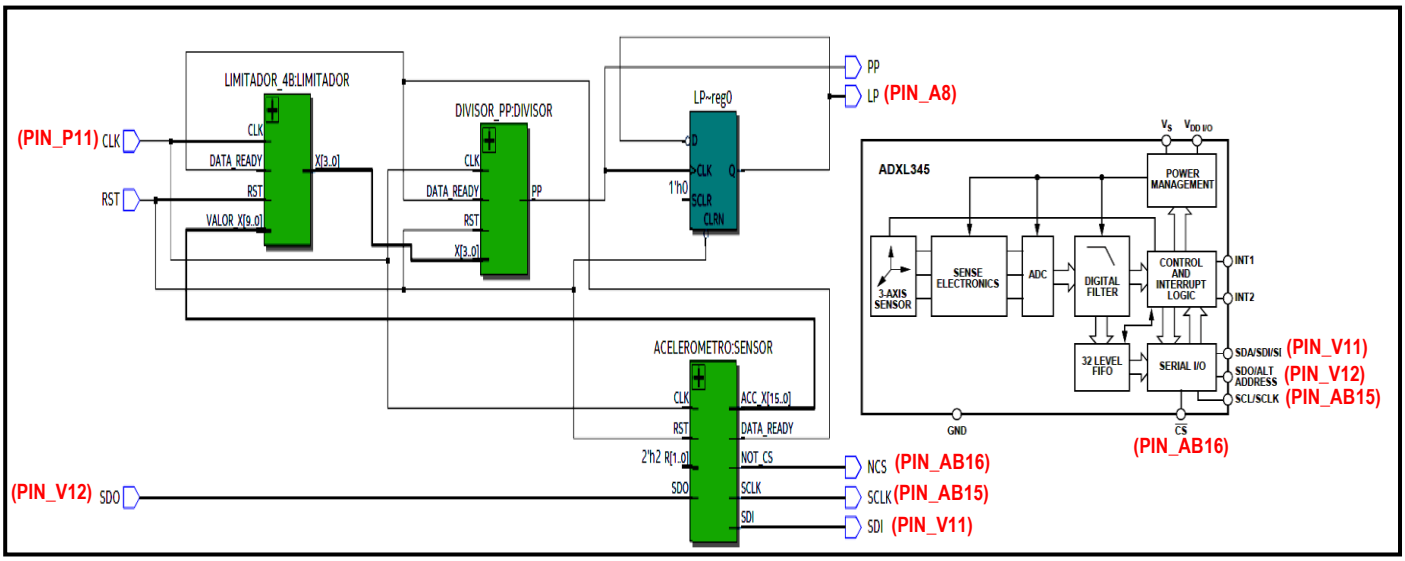
7.1. Interfaces com a placa de desenvolvimento DE10-Lite

No **Laboratório de Sistemas Digitais II** as interfaces com o usuário devem utilizar os recursos disponíveis na placa de desenvolvimento **DE10-Lite**. Essas interfaces têm as seguintes especificações:

- O sistema de Supervisão de Percurso de Pedágio Aberto deve ser configurado no **FPGA** da placa de desenvolvimento **DE10-Lite**, utilizando os botões, chaves, displays e o acelerômetro digital ADXL345, conforme representado nas **Figuras 9** e **10**;

- O sinal de passagem por pórtico (**PP**) deve ser controlado pelo acelerômetro digital ADXL345 presente na placa DE10-Lite. Para essa interface deve ser utilizado o componente **Gerador_PP** (código ***VHDL*** **GERADOR_PP.vhd** **fornecido no Moodle**). Esse componente deve ser conectado à interface de comunicação **SPI** (*Serial Peripheral Interface*) do acelerômetro digital ADXL345 com quatro sinais (**SDI**, **SDO**, **SCLK** e **NCS**), utilizando a atribuição de pinos representada na **Figura 4**. Como pode ser observado na visão RTL apresentada na **Figura 4**, o componente **Gerador_PP** é composto de dois outros componentes: Acelerômetro (que realiza a comunicação com o ADXL345) e Divisor_PP (que divide o clock de 50 MHz para a obtenção de pulsos simulados de passagem de pórtico (**PP**) em função da posição do acelerômetro).

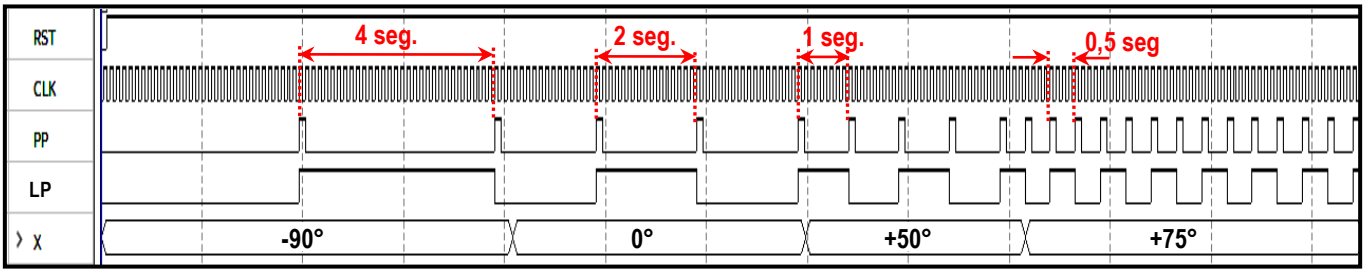
Figura 4: Sinais de interface entre o componente **GERADOR_PP** e o acelerômetro digital ADXL345.



Fonte: Autor (adaptado de ANALOG DEVICES, 2015).

- Com a placa DE10-Lite na posição horizontal a frequência dos pulsos do sinal **PP** é de 0,5 Hz (período de repetição de 2 segundos). A cada inclinação da placa de 12°, no eixo X, a periodicidade dos pulsos é variada em 250 ms (incrementa com variações angulares positivas e decrementa para inclinações negativas no eixo X). Na **Figura 5** estão representadas algumas das periodicidades do sinal **PP** que podem ser obtidas com inclinações no eixo X entre -90° e +75°. Além do sinal **PP**, que tem a largura de pulso de um período de clock (20 ns) o componente **Gerador_PP** também fornece um sinal **LP** que altera seu estado a cada borda de subida do sinal **PP**. Esse sinal permite que seja observado visualmente (através do **LED0**) a frequência de ativação do sinal **PP**, que pode ter periodicidade de 0,25 a 4 segundos (frequências de 4 Hz a 0,25 Hz). Maiores detalhes da configuração e aplicações do acelerômetro digital ADXL345 podem ser obtidas no Relatório de Iniciação Didática: Módulo Acelerômetro Digital na Placa DE10-Lite (PRATES, 2021).

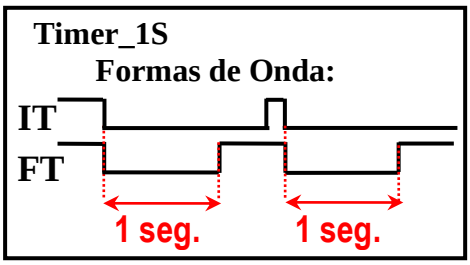
Figura 5: Diagrama de tempos dos sinais de saída do componente **GERADOR_PP**.



Fonte: Autor.

- O fluxo de dados do sistema deve conter **três registradores** utilizados para armazenamento dos valores de crédito restante (**CR**), distância parcial percorrida (**DP**) e distância total percorrida (**DT**), os quais devem ser conectados a decodificadores para display de sete segmentos da seguinte forma:
  - **REG_A** para registro do crédito restante (**CR**), apresentado nos displays **HEX1** e **HEX0**;
  - **REG_B** para registro da distância parcial percorrida (**DP**), apresentado nos displays **HEX3** e **HEX2**;
  - **REG_C** para registro da distância total percorrida (**DT**), apresentado nos displays **HEX5** e **HEX4**;
- Para apresentação dos valores nos displays é necessária a conversão dos valores binários de oito bits (B[7..0]) em códigos BCD de modo que cada display apresente a Centena, Dezena e Unidade do número decimal correspondente. O código ***VHDL*** de um decodificador binário para display BCD de dois dígitos (arquivo **DECODIFICADOR_BCD.vhd**) é **fornecido no Moodle**. Esse código utiliza o algoritmo "Desloca e Soma 3" (*double dabble algorithm*). Esse algoritmo realiza diversas iterações. Em cada iteração, qualquer dígito BCD (ou nibble que representa unidade, dezena, centena) que seja pelo maior ou igual a 5 é incrementado de 3 e, em seguida, todos os bits são deslocados de uma posição à esquerda. O incremento garante que um valor de um dígito BCD igual a 5, incrementado de 3 e deslocado para a esquerda, se torne 16, gerando assim um "vai um" para o próximo dígito BCD. Logo, se algum dígito tiver valor cinco ou mais, o valor três é adicionado para garantir que esse dígito gere o "vai um" para o próximo dígito na base dez (uma vez que o valor cinco dobrado deve gerar o "vai um" para o próximo dígito decimal). Deve ser observado que o código **DECODIFICADOR_BCD.vhd** decodifica um número binário para dois dígitos decimais, logo o máximo valor representável em decimal é 99.
- A temporização necessária para a base de tempo para o sistema de Supervisão de Percurso de Pedágio Aberto deve ser implementada com um temporizador de **1 segundo** cujo código **VHDL** é **fornecido no Moodle** (arquivo **TIMER_1S.vhd**). Esse temporizador conta os pulsos de clock (50 MHz) e mantém desativado o sinal de Fim de Temporização (**FT=0**) até que seja decorrido o tempo de um segundo após ter sido desativado o sinal Inicia Temporizador (**IT=0**). Após um segundo da iniciação do temporizador o sinal **FT** é ativado (**FT=1**). A **Figura 6** representa o diagrama de temporização do **TIMER_1S**;

Figura 6: Diagrama de tempos do temporizador de 1 segundo (**TIMER_1S)**.

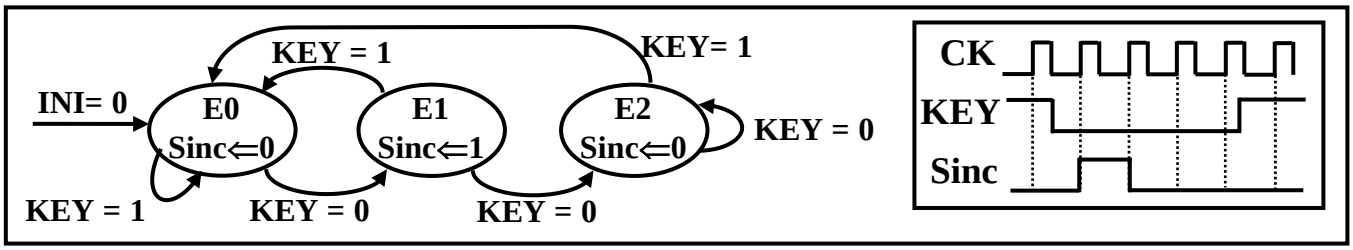


Fonte: Autor.

- Para verificação da multa por excesso de velocidade (**MEV**) deve ser considerado que o valor de 1 segundo é o tempo mínimo para o veículo percorrer a distância entre dois pórticos sem exceder a velocidade máxima. Logo, a multa por excesso de velocidade deve ocorrer se o sistema de supervisão detectar dois pulsos PP dentro do intervalo de tempo em que **FT** está desativado;
- Os sinais **IC** e **DC** devem ser representados por botões de contato momentâneo (**KEY[0]** e **KEY[1]**) existentes na placa de desenvolvimento **DE10-Lite**. Esses botões já possuem um circuito de *debouncing* associado no próprio hardware (porta com *Schmitt Trigger*). Esses botões possuem **lógica de ativação negada**, ou seja, **quando o botão é pressionado** o sinal na entrada do FPGA **vai a nível lógico zero**;
- Para assegurar que os sinais gerados pelos botões **IC** e **DC** estejam ativos apenas **durante um período de clock** devem ser utilizados **circuitos de sincronização** (código **Sincron.vhd**, **fornecido no Moodle**),

que opera conforme o diagrama de estados representado na **Figura 7** (**v**ide o exemplo 3.9 de Vahid, 2008). Note que o botão (**KEY**) tem ativação **em nível lógico zero**, mas a saída **Sinc** é ativa **em nível lógico um**, ou seja, a lógica do circuito de sincronização executa a inversão do estado lógico do sinal de entrada;

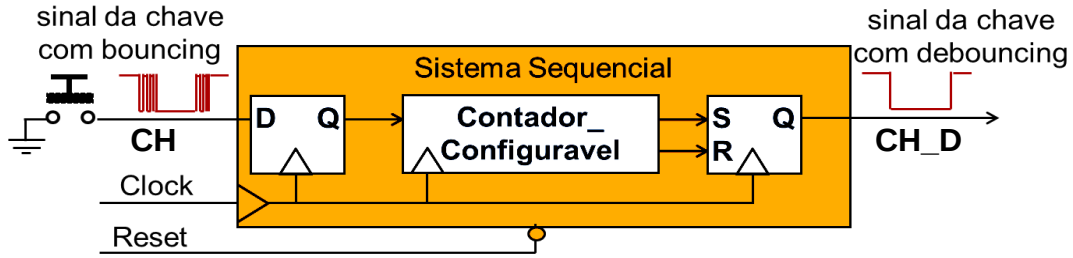
Figura 7: Diagrama de estados do sincronizador de botão (KEY) com saída (Sinc).



Fonte: Autor.

- Os sinais **RP** e **MO** devem ser representados pelas chaves deslizantes (*slide switch*) **SW0** e **SW** da placa de desenvolvimento **DE10-Lite**. Essas chaves possuem vibração de contatos (*bouncing*), logo, para esses sinais devem passar por um circuito de eliminação de trepidação de contatos (***Debouncing***). Para essa função pode ser utilizado o circuito descrito no exemplo 7 da aula 4 do laboratório (código **Debouncing.vhd fornecido no Moodle**) e representado na **Figura 8**.

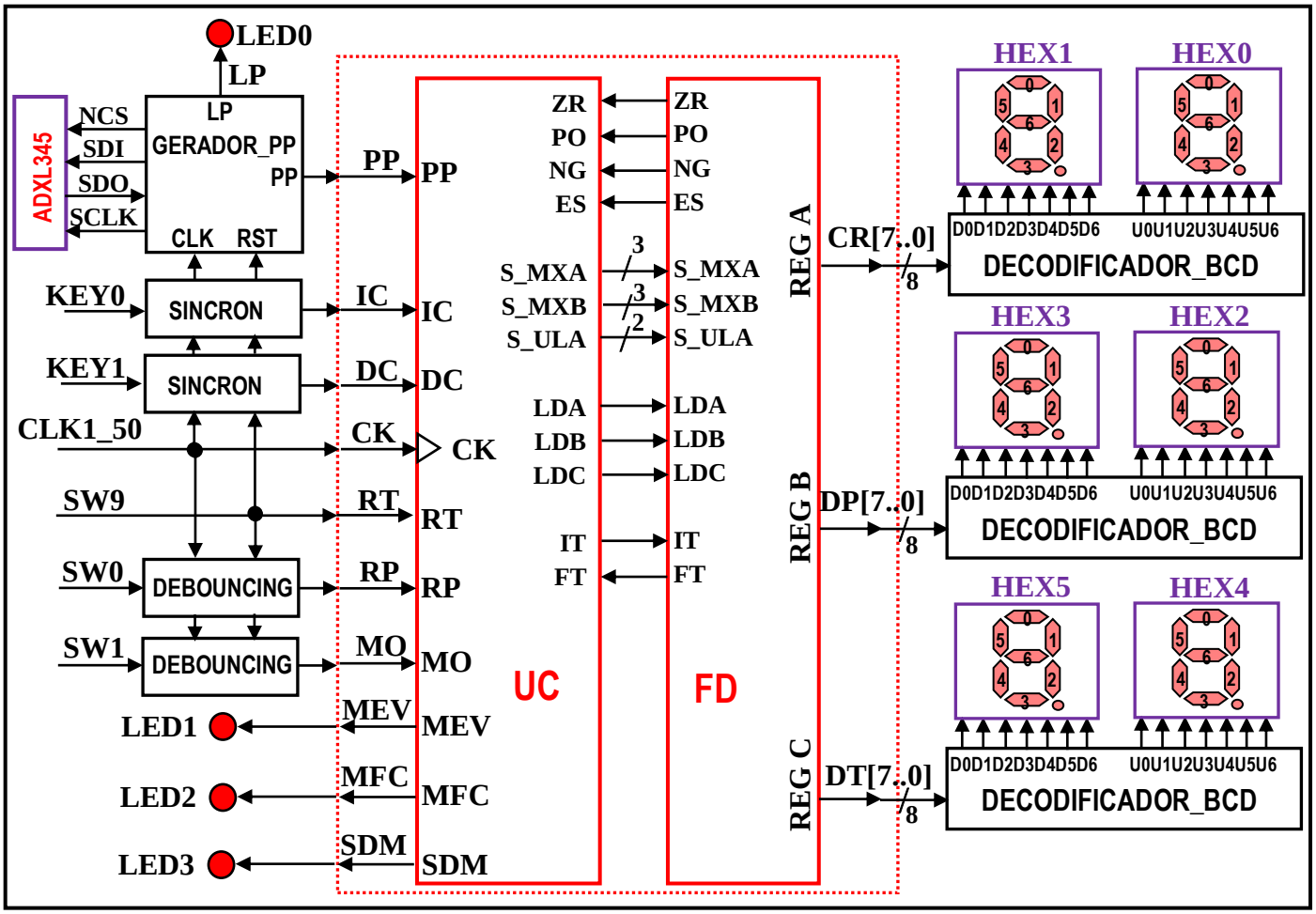
Figura 8: Diagrama de blocos do circuito eliminador de trepidação de chaves.



Fonte: Autor.

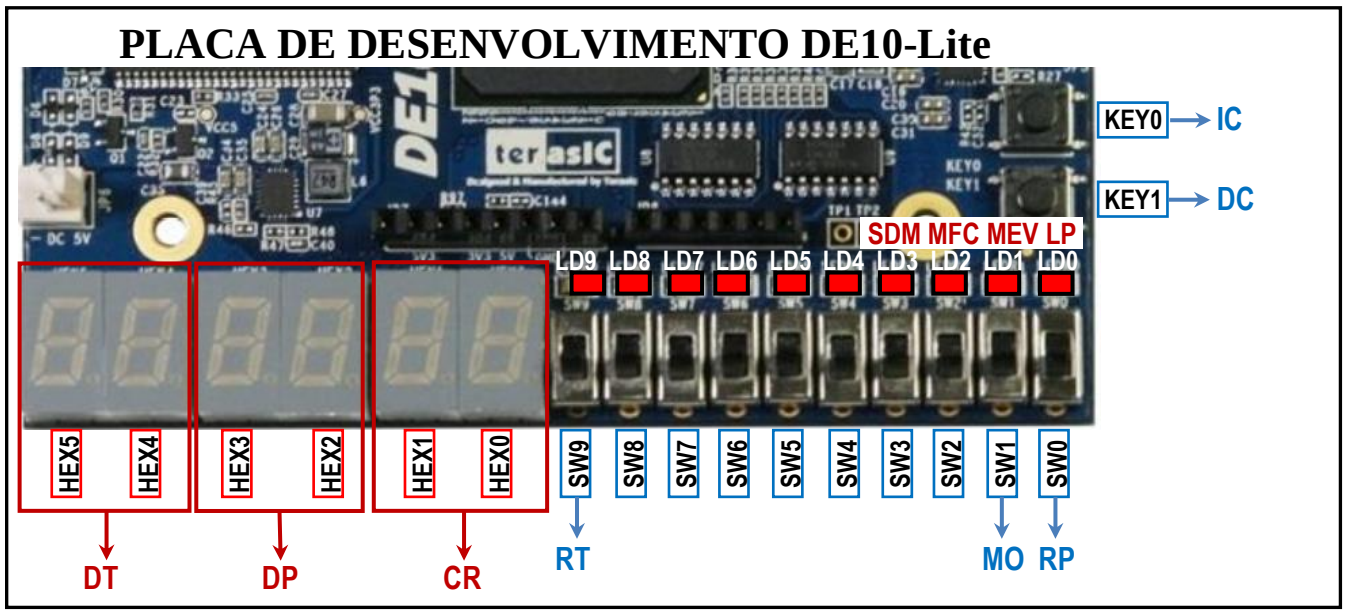
- O botão de reinício parcial (**RP**) deve reiniciar apenas o contador de distância parcial (**DP**) e o led de sinalização de multa por excesso de velocidade (**MEV**). Se um pulso de passagem pelo pórtico (**PP**) ocorrer simultaneamente com ao reinício parcial (**RP**) o **pulso da passagem pelo pórtico deve ter prioridade**, ou seja, o decremento do crédito restante e incremento das distâncias total e parcial tem prioridade sobre o reinício parcial;
- O sinal de reinício total (**RT**) deve ser simulado com uma chave deslizante (**SW9**). Esse sinal deve operar como um **reset assíncrono** do sistema, reiniciando em zero todos os registadores. Como é um sinal de reinício assíncrono não há necessidade de circuito de eliminação de trepidação para essa chave;
- O sinal de multa por excesso de velocidade (**MEV**) deve ser apresentado no led vermelho **LED1**.
- O sinal de multa por falta de crédito (**MFC**) deve ser apresentado no led vermelho **LED2**;
- A sinalização de limite de distância máxima percorrida (**SDM**) deve ser apresentado no led vermelho **LED3**;

Figura 9: Diagrama blocos do sistema de Supervisão de Percurso de Pedágio Aberto.



Fonte: Autor.

Figura 10: Interfaces do sistema de Supervisão de Percurso de Pedágio Aberto na placa **DE10-lite**.



Fonte: TERASIC, 2020 (adaptado).

8.Apresentação da Simulação do Projeto (Aula 13 no AVA)

Para a apresentação das formas de onda da simulação (**UC** + **FD)** do sistema de Supervisão de Percurso de Pedágio Aberto no laboratório deve ser utilizado o simulador do Quartus Prime. Nessa simulação **não** devem ser incluídos os circuitos do gerador_PP, nem os sincronizadores, nem circuitos de *debouncing* e nem os decodificadores. Deve ser simulada a **Máquina de Estados** da unidade de controle **juntamente com os elementos do fluxo de dados** (ULA, registradores, multiplexadores e timer de 1 segundo), indicados no **retângulo pontilhado vermelho** da **Figura 9**. A simulação deve ser realizada **no modo funcional**, utilizando o editor de vetor de formas de onda.

As formas de onda simuladas no modo funcional devem apresentar os seguintes sinais de interface:

- **Sinais de entrada**: **CK**, **RT**, **RP**, **MO**, **IC**, **DC** e **PP**;
- **Sinais de saída**: **SDM**, **MFC**, **MEV**, **CR[7..0]**, **DP[7..0]** e **DT[7..0];**

Observação: **Para a simulação** do sistema a variável de contagem do código do **TIMER_1S.vhd deve ser modificada** de 50.000.000 **para 4** (**count <= "00000000000000000000000100"**) de modo que o período de temporização passe a ser de quatro períodos de clock. Para a realização da **programação do FPGA** o valor dessa variável de contagem **deve ser retornado ao original** (**count <= "10111110101111000010000000"**).

Essas formas de onda devem demonstrar as seguintes funções do sistema (pode ser mais que uma tela com as situações especificadas):

- **Reinício total do sistema** (**RT**), carregando o valor zero nos registradores de distância e o valor inicial do crédito;
- **Ajuste do valor de crédito** (**modo de operação de carga de valores**) com os botões de incrementa (**IC**) e decrementa (**DC**) devem ser somados/subtraídos os incrementos de ajuste de crédito (**AC**), a partir do valor inicial de crédito (**CRI**), com a verificação dos limites mínimo e máximo de crédito (**0** a **CRM**);
- **Registro do decremento do crédito disponível** (**CR**), **registro da distância parcial** (**DP**) e **registro da distância total** (**DT**) para cada pulso de pórtico (**PP**);
- **Geração da sinalização de multa por falta de crédito** (**MFC**);
- **Geração da sinalização de multa por excesso de velocidade** (**MEV**);
- **Geração da sinalização de limite máximo de percurso** (**SDM**);
- **Reinício parcial** (**RP**), zerando o registro de distância parcial e sinalização de multa por velocidade;
- **Acionamento do reinício parcial** (**RP**) **simultâneo ao de um pulso de pórtico** (**PP**).

A figura apresentada no **Anexo 3** deste roteiro mostra um exemplo possível de simulação das formas de onda exemplificando algumas operações que devem ser demonstradas na aula de projeto.

9.Conclusão do Projeto (Aula 14 no CGI)

Para a conclusão do projeto do sistema de Supervisão de Percurso de Pedágio Aberto devem ser incluídos todos os componentes lógicos: **gerador de pulso** (controlado pelo acelerômetro digital ADXL345), **decodificadores_BCD**, saídas para **LED**'s, saídas para displays (**HEX0** a **HEX5**), o **TIMER_1S** **deve ser reconfigurado para a contagem de 50.000.000 pulsos**, assim como devem ser enumerados os pinos de entrada/saída do **FPGA** da família **MAX 10** (modelo: **10M50DAF484C7G**) utilizado. Consulte a **Tabela de Associação de Pinos** ou o **DE10-Lite User Manual** e o **Tutorial de Quartus Prime para Projeto (baseado em VHDL**. Disponíveis nas referências de laboratório no Moodle (TERASIC, 2020), (PRATES, 2019).

A **Tabela 3** apresentada no **Anexo 5** deste roteiro mostra a numeração de pinos básica aplicável ao projeto do sistema de Supervisão de Percurso de Pedágio Aberto.

Atenção: Não deixe conectores de saída (OUTPUT) **sem atribuição de pino** (utilizando o ***Pin Planner***). O Quartus Prime associa automaticamente todos os conectores de saída a algum pino de I/O, **podendo ocorrer conflito** de um pino escolhido pela ferramenta com um sinal já utilizado na placa **DE10-Lite**.

Para a verificação da correta operação do sistema integrado o aluno pode utilizar a **Sala de Projetos (sala D3-07)** onde estão disponíveis até cinco máquinas com o ambiente do **Quartus Prime**, bem como podem ser solicitadas no almoxarifado do **CLE** as placas **DE10-Lite** para teste completo do sistema.

10. Critérios para Elaboração do Relatório e Avaliação do Projeto no Laboratório

- O projeto deve ser desenvolvido **em grupo de no máximo dois alunos, devendo ser entregue um único relatório para o grupo**. Projetos copiados entre grupos, **total ou parcialmente**, não serão considerados, **sendo atribuído zero para todos os projetos iguais**;
- As avaliações das **apresentações e os relatórios serão individuais**, sendo que **cada aluno do grupo deve demonstrar o perfeito conhecimento de todas** as etapas do projeto e teste;
- O relatório deve ser entregue no Moodle **(arquivo no formato PDF**) e deve conter a documentação detalhada do projeto, permitindo este seja entendido e executado por outro projetista. O relatório deve apresentar, **no mínimo**, os seguintes itens:

1. **Capa** com o nome, número dos alunos e turma;
2. **Parte 1: Descrição do Problema:**

A introdução deve apresentar um resumo dos objetivos do projeto. Deseja-se uma descrição do modo de operação da unidade de controle principal (***UC***) com a definição dos sinais de interface (tais como botões, chaves, Leds, *displays*). Deve ser apresentado o **Diagrama da Máquina de Estados de Alto Nível** (***FSMD***) contendo a sequência lógica dos estados e a especificação das características particulares da implementação realizada pelo grupo (valores de crédito inicial, incremento de ajuste de crédito, limite máximo de crédito e tarifa). Podem ser utilizadas as figuras apresentadas neste roteiro (modificadas para atender às especificações particulares de cada projeto), entretanto não serão aceitas simples cópias do roteiro experimental;

3. **Parte 2: Descrição da Realização do Projeto:**

Devem ser descritos os vários componentes que compõem o projeto. Deve ser explicado como cada parte do projeto funciona individualmente e como ela opera no conjunto do sistema. Devem ser apresentados **em detalhe no mínimo** os seguintes elementos:

- **Diagrama Transição de Estados (*FSM*) da Unidade de Controle:** com **todos os detalhes** da definição das suas entradas, saídas, transições de estado e o que representa cada estado;
- **Tabela de Sinais de Controle do Fluxo de Dados:** com **todos os sinais de controle do FD para cada estado da UC**;
- **Diagrama de Interconexão do Fluxo de Dados:** com **os detalhes de interconexão** dos elementos do **FD** adotado no projeto (**incluir a visão RTL do Fluxo de Dados**);
- **Códigos VHDL desenvolvidos pelo aluno**: para todos os **componentes desenvolvidos ou modificados pelo aluno** (como Máquina de Estados, Multiplexadores) devem ser apresentados **os respectivos códigos VHDL**;
- É **obrigatório o Diagrama RTL do projeto completo** (similar ao **Anexo 4**);
- É **obrigatório o Diagrama de Estados gerado pelo Quartus para a máquina de estados apresentada em sala de aula** (Tools > Netlist Viewers> State Machine Viewer);
- É **obrigatória a inclusão das formas de onda da simulação funcional apresentadas na aula de simulação de projeto** (geradas no laboratório pelo simulador do Quartus Prime – arquivo**.vwf**).

4. Conclusão:

Na conclusão do relatório devem ser comentados os objetivos do projeto 2 do Laboratório de Sistemas Digitais II e de que forma os alunos acreditam que os tenham atendido. Comentar o que foi aprendido com esse projeto e/ou quais os conceitos importantes que foram praticados. Outro aspecto que pode ser comentado é o que os alunos fariam de forma diferente se tivessem que executar esse projeto novamente.

5. Arquivo Anexo:

Ao postar o relatório principal, os alunos **devem incluir, obrigatoriamente**, na página de *upload* do **Moodle** o relatório de compilação resumido da etapa de Fitter (output_files > arquivo**.fit.rpt**).

ATENÇÃO: O *upload* dos **dois arquivos** no Moodle é de **inteira responsabilidade do aluno**, que **deve verificar** se os arquivos foram efetivamente postados (verificando sua visualização após o *upload*).
Não serão aceitos arquivos postados em **turmas incorretas**, nem **fora do horário** especificado para upload da atividade.

11.Referências Bibliográficas

ANALOG DEVICES – Datasheet: ADXL345 Digital Accelerometer – Analog Devices, Inc., 2015. Disponível em: (*https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf*). Acesso em: 10/01/2021.

ARTESP – Sistema Ponto a Ponto - – Agência de Transportes do Estado de São Paulo. Disponível em: (*http://www.artesp.sp.gov.br/Style%20Library/extranet/rodovias/sistema-ponto-a-ponto.aspx*). Acesso em: 11/01/2021.

CNT – Transporte em Movimento – 06/2020 - Novas tecnologias de pagamento de pedágio – Confederação Nacional de Transporte - CNT. Disponível em: (*https://cdn.cnt.org.br/diretorioVirtualPrd/bf8665da-3e39-45cf-9fbd-2dbbafc9ddd4.pdf*). Acesso em: 13/01/2021.

GAZETA – Novo pedágio permitirá multar – Gazeta do Povo - 22/03/2012. Disponível em: (*https://www.gazetadopovo.com.br/vida-e-cidadania/novo-pedagio-permitira-multar-8d9f7fcuqucoqjqnu0caez3ny/*). Acesso em: 11/01/2021.

QUARTUS – Intel Quartus Prime – Lite, Version 16.1. Intel-FPGA (©Intel Corporation). Disponível em: (*https://fpgasoftware.intel.com/?edition=lite*). Acesso em: 07/07/2020.

PRATES, R. R. – Tutorial de Quartus Prime para Projeto de CPLD/FPGA (baseado em VHDL). Programa de Iniciação Didática – Disponível nas referências do Laboratório de Sistemas Digitais – Centro Universitário FEI, 2019.

PRATES, R. R. – Módulo Acelerômetro Digital na Placa DE10-Lite – Revisão 1. Programa de Iniciação Didática – Disponível nas referências do Laboratório de Sistemas Digitais – Centro Universitário FEI, 2021.

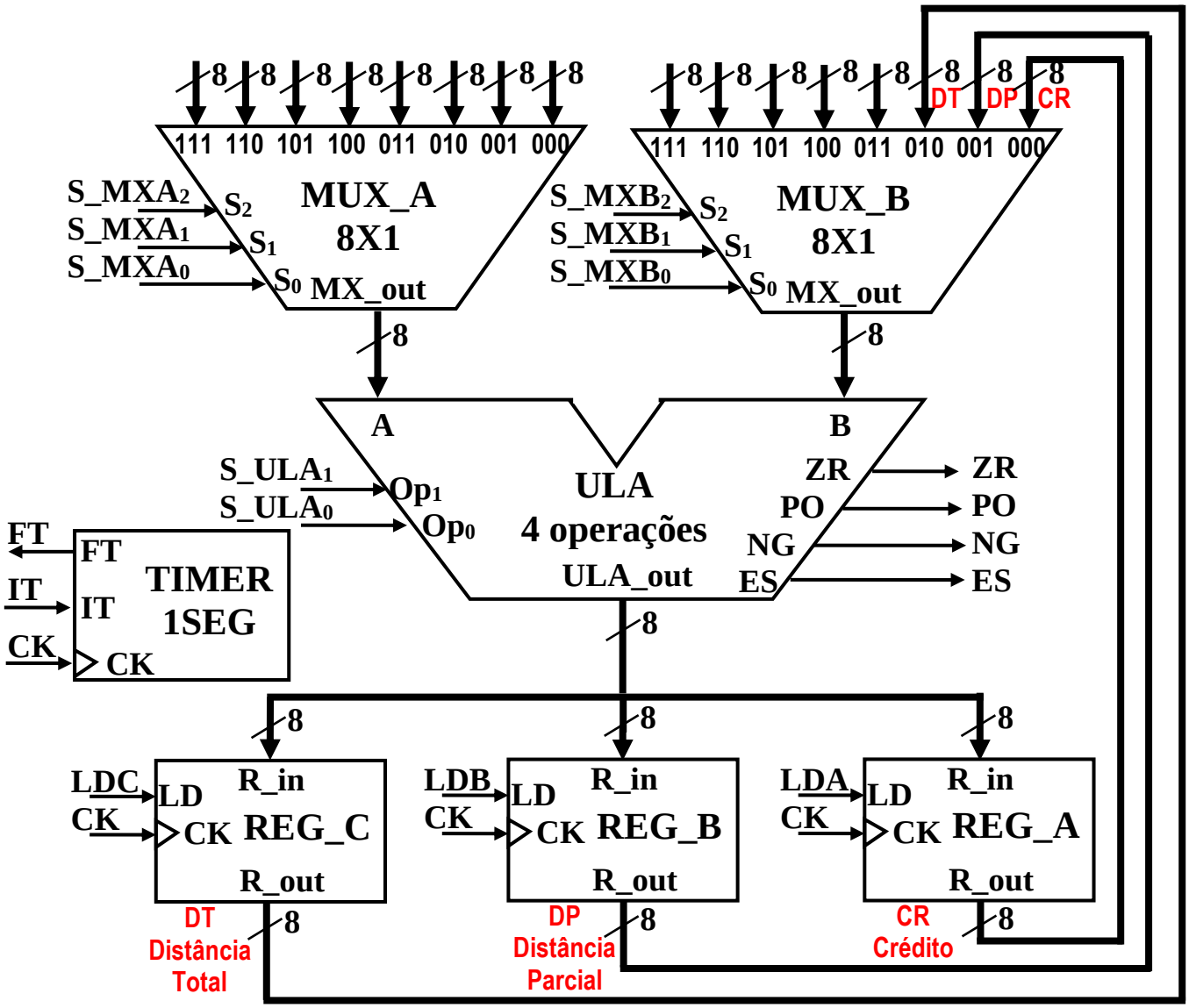
TERASIC - DE10-LITE Board - User Manual. (©Terasic Inc). Disponível em: (*https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=234&No=1021&PartNo=4*). Acesso em: 06/07/2020.

TOCCI, Ronald J; WIDMER, Neal S.; MOSS, Gregory L. **Sistemas Digitais: Princípios e Aplicações**. Revisão técnica: Renato Giacomini. Tradução: Jorge Ritter. 11. Ed. São Paulo: Pearson Prentice Hall, 2011.

VAHID, F. - **Sistemas Digitais – projeto, otimização e HDLs**, 1ª Ed., Artmed – Bookman, 2008.

12.Definição dos Blocos Operacionais do Fluxo de Dados

Figura 11: Diagrama dos blocos operacionais do fluxo de dados do sistema.

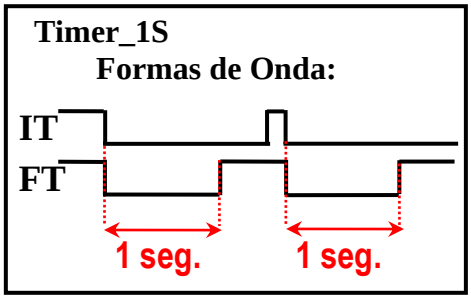


Fonte: Autor.

Tabela 2: Tabela de funções e sinalização de status da ULA e temporização do Timer_1S

Código (Op_1 Op_0)	Operação da ULA
0 0	Passa B
0 1	NOT(B)
1 0	B + A
1 1	B – A

Sinal	Sinalização da ULA
ZR = 1	Resultado = zero
PO = 1	Resultado > zero
NG = 1	Resultado < zero
ES = 1	Resultado > 255

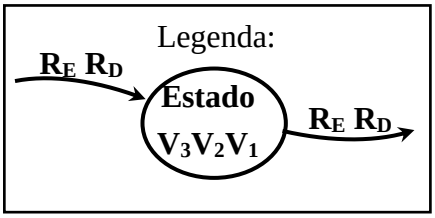


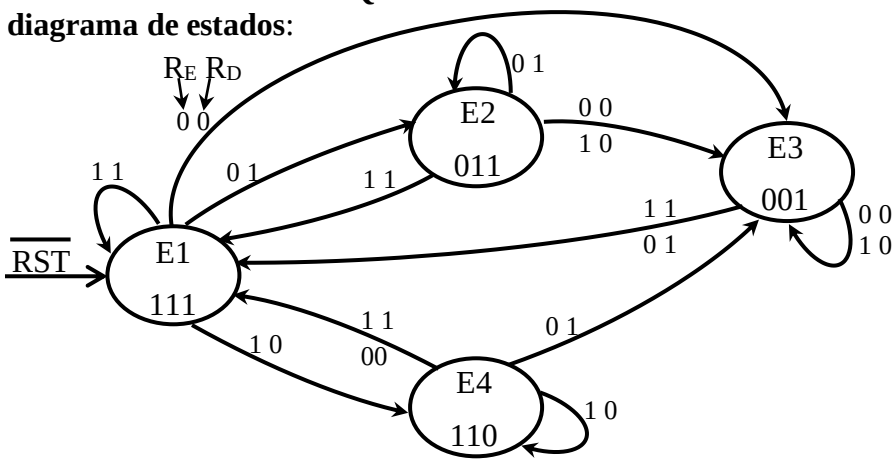
Fonte: Autor.

ANEXO 1 – TEMPLATE VHDL PARA MÁQUINA DE ESTADOS

a) Considere o seguinte exemplo de diagrama de estados:

Entradas: CK, RST, RE e RD
Saídas: V3, V2, V1
Estados: E1, E2, E3, E4





Para esse diagrama de estados as **Equações dos Estados** e **Equações das Saídas** são:

EQUAÇÕES DOS ESTADOS	$E1_+ = RE.RD.(E1 + E2 + E3 + E4) + /RE.RD.E3 + /RE./RD.E4 = RE.RD + RD.E3 + /RE./RD.E4$
	$E2_+ = /RE.RD.E1 + /RE.RD.E2 = /RE.RD.(E1 + E2)$
	$E3_+ = /RE./RD.E1 + /RD.(E2 + E3) + /RE.RD.E4$
	$E4_+ = RE./RD.(E1 + E4)$

EQUAÇÕES DAS SAÍDAS	$V3 = E1 + E4$
	$V2 = E1 + E2 + E4$
	$V1 = E1 + E2 + E3$

b) Descrição em VHDL:

A descrição da máquina de estados em código **VHDL** pode ser realizada **diretamente a partir das equações de estado** como mostrado na **Figura A1.1**.

Figura A1.1: Exemplo de codificação de máquina de estados diretamente por equações de estados.

```
ENTITY EXEMPLO_EE IS
    PORT (CLK, CLK_EN, RST, RE, RD: IN STD_LOGIC;-- declaração dos sinais de entrada
          V: OUT STD_LOGIC_VECTOR(3 DOWNTO 1) );  -- declaração dos sinais de saída
END EXEMPLO_EE;

ARCHITECTURE BEHAVIOR OF EXEMPLO_EE IS
    SIGNAL E1,E2,E3,E4: STD_LOGIC;                -- declaração das variáveis de estado
BEGIN
    PROCESS (CLK, CLK_EN, RST)       -- processo para definição das equações de estado
    BEGIN
        IF (RST='0') THEN E1 <='1'; E2 <='0'; E3 <='0'; E4 <='0';-- estado inicial
        ELSIF (CLK'event and CLK='1' and CLK_EN='1') THEN         -- detecção de CLK
         E1 <= (RE AND RD) OR (RD AND E3) OR ((NOT(RE) AND NOT(RD)) AND E4);
         E2 <= (NOT(RE) AND RD) AND(E1 OR E2);
         E3 <= ((NOT(RE) AND NOT(RD)) AND E1) OR (NOT(RD) AND (E2 OR E3)) OR
               ((NOT(RE) AND RD) AND E4);
         E4 <= ((RE AND NOT(RD)) AND (E1 OR E4));                 Equações dos Estados
        END IF;
    END PROCESS;

    -- Definição das saídas
    V(3) <= (E1 OR E4);        -- equação da saída V3
    V(2) <= (E1 OR E2 OR E4); -- equação da saída V2
    V(1) <= (E1 OR E2 OR E3); -- equação da saída V1          Equações das Saídas
END BEHAVIOR;
```

Fonte: Autor.

A descrição em código **VHDL** de máquina de estados mais comum em ferramentas automáticas utiliza a técnica de um bit por estado, mas as equações dos estados são representadas por estruturas lógicas sequenciais (**CASE** e **IF-THEN-ELSE**) para descrever as transições das expressões das equações lógicas dos estados e das saídas. Essa deve ser a técnica utilizada na realização dos projetos do laboratório.
Neste caso utilizam-se dois tipos de processos: um **processo síncrono** para definir as **transições de estados** (uma condição Case-When para cada estado) e um **processo assíncrono** para definir as condições lógicas das saídas (uma condição Case-When para cada estado), como mostrado na **Figura A1.2.**

Figura A1.2: Exemplo de codificação de máquina de estados por lógica sequencial.

```
ENTITY FSM_EXEMPLO IS
    PORT (  CLK, CLK_EN, RST, RE, RD: IN STD_LOGIC;-- declaração dos sinais de entrada
          V: OUT STD_LOGIC_VECTOR(3 DOWNTO 1) );    -- declaração dos sinais de saída
END FSM_EXEMPLO;

ARCHITECTURE BEHAVIOR OF FSM_EXEMPLO IS
    TYPE type_state IS (E1,E2,E3,E4);          -- criação de tipos enumerados
    SIGNAL Estado: type_state;                 -- declaração de variáveis de estado
    SIGNAL Entradas: STD_LOGIC_VECTOR(2 DOWNTO 1);    -- declaração de vetor auxiliar

BEGIN
    Entradas<= RE & RD;     -- concatenação dos sinais de entrada RE e RD como um vetor

    PROCESS (CLK, CLK_EN, RST)   -- processo para definição das transições dos estados
    BEGIN
        IF (RST='0') THEN   Estado <= E1;                -- estado de reset do sistema
        ELSIF (CLK'event and CLK='1' and CLK_EN='1') THEN -- detecção de borda de CLK
            CASE Estado IS                                -- sincronização com CLK_EN
                WHEN E1 =>
                    IF Entradas= "01" THEN Estado <= E2;     -- transição E1->E2
                    ELSIF Entradas= "10" THEN Estado <= E4; -- transição E1->E4
                    ELSIF Entradas= "00" THEN Estado <= E3; -- transição E1->E3
                    ELSE Estado <= E1;        -- essa condição não é obrigatória
                    END IF;
                WHEN E2 =>
                    IF Entradas= "11" THEN Estado <= E1;
                    ELSIF Entradas= "00" OR Entradas= "10" THEN Estado <= E3;
                    ELSE Estado <= E2;        -- essa condição não é obrigatória
                    END IF;
                WHEN E3 =>
                    IF Entradas= "01" OR Entradas= "11" THEN Estado <= E1;
                    ELSE Estado <= E3;        -- essa condição não é obrigatória
                    END IF;
                WHEN E4 =>
                    IF Entradas= "00" OR Entradas= "11" THEN Estado <= E1;
                    ELSIF Entradas= "01" THEN Estado <= E3;
                    ELSE Estado <= E4;        -- essa condição não  é obrigatória
                    END IF;
      END CASE;
        END IF;
    END PROCESS;

    PROCESS (Estado)        -- processo para definição das variáveis de saída
    BEGIN
            CASE Estado IS
                WHEN E1 =>
                    V <= "111";      -- atribuição das saídas para o Estado E1
                WHEN E2 =>
                    V <="011";       -- atribuição das saídas para o Estado E2
                WHEN E3 =>
                    V <= "001";      -- atribuição das saídas para o Estado E3
                WHEN E4 =>
                    V <="110";       -- atribuição das saídas para o Estado E4
          END CASE;
      END PROCESS;

END BEHAVIOR;
```

Uma condição "Case - When" para cada variável de "Estado"

Processo síncrono para definição das transições dos Estados

Este processo não é síncrono, os valores são atribuídos às variáveis de saída sempre que houver mudança do sinal "Estado".

Uma condição "Case - When" para cada variável de "Estado"

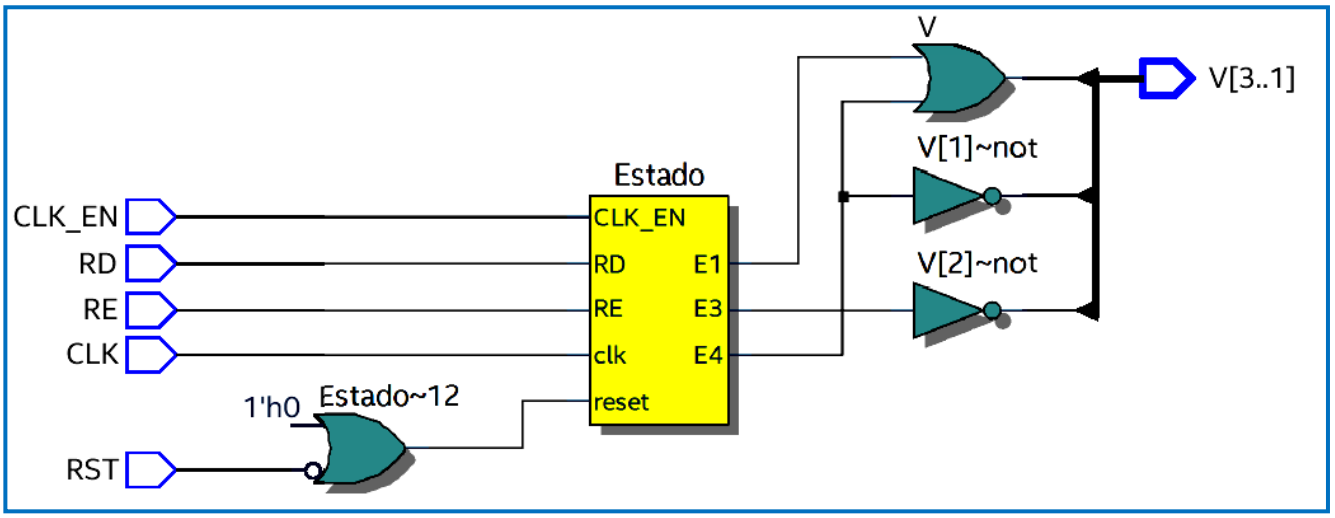
Processo assíncrono para definição das Saídas

Fonte: Autor.

c) Representação RTL:

A representação **RTL** da descrição da máquina de estados por um bit por estado do exemplo anterior, utilizando estruturas lógicas sequenciais resulta na **Figura A1.3**.

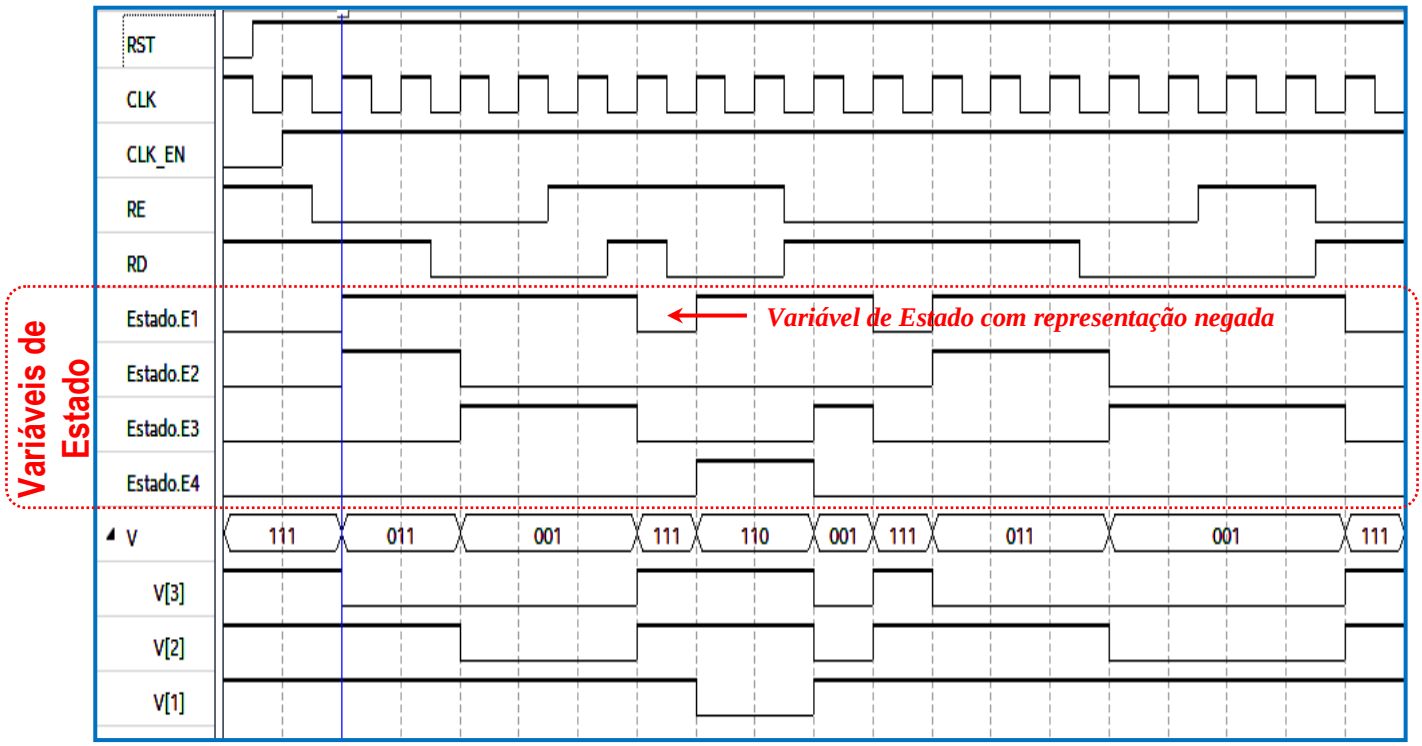
Figura A1.3: Exemplo de codificação de máquina de estados por lógica sequencial.



Fonte: QUARTUS, 2020 (adaptado).

d) Exemplo de formas de onda resultantes (incluindo os sinais de Estado):

Figura A1.4: Formas de onda simuladas para máquina de estados do exemplo.



Fonte: QUARTUS, 2020 (adaptado).

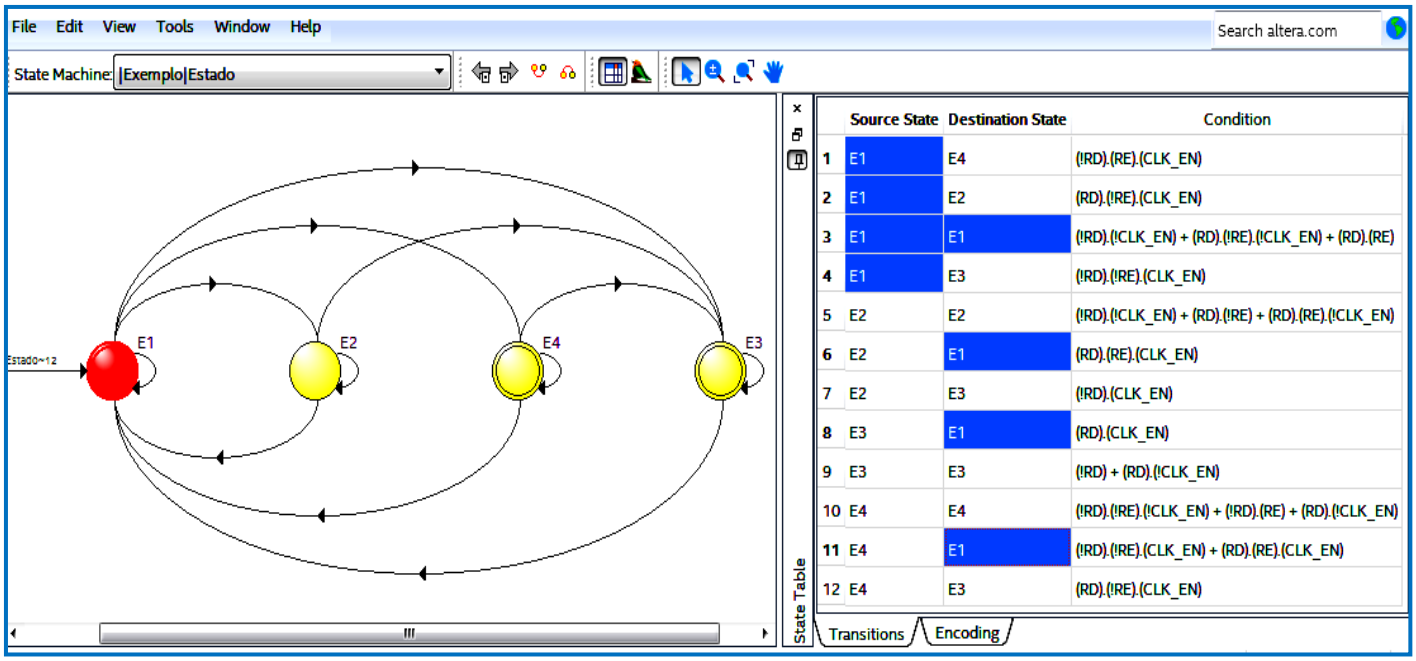
Para a inclusão das variáveis de estado (**Estado.XX**) nas formas de onda do simulador do Quartus Prime deve ser selecionada a opção ***Filter: Design Entry (all names)*** na janela ***Node Finder*** e ativar a opção ***List*** (serão apresentados todos os sinais internos da arquitetura), podendo ser selecionados os sinais que

representam o "**Estado**" do sistema. Observar que o estado inicial do sistema (neste exemplo o estado **E1**) possui sua representação de ativação negada (está ativo em nível lógico zero).

e) Geração automática do diagrama de estados:

O Quartus Prime também pode gerar um grafo com os estados associados à máquina de estados do projeto. Para isso selecione a opção ***Tools*> *Netlist Viewers* > *StateMachine Viewer***. Se a descrição **VHDL** estiver no formato de um bit por estado com estruturas lógicas sequenciais o exemplo anterior resulta na **Figura A1.5**.

Figura A1.5: Grafo gerado pelo Quartus Prime com o diagrama de estados do ME_Exemplo.



	Source State	Destination State	Condition
1	E1	E4	(!RD).(RE).(CLK_EN)
2	E1	E2	(RD).(!RE).(CLK_EN)
3	E1	E1	(!RD).(!CLK_EN) + (RD).(!RE).(!CLK_EN) + (RD).(RE)
4	E1	E3	(!RD).(!RE).(CLK_EN)
5	E2	E2	(!RD).(!CLK_EN) + (RD).(!RE) + (RD).(RE).(!CLK_EN)
6	E2	E1	(RD).(RE).(CLK_EN)
7	E2	E3	(!RD).(CLK_EN)
8	E3	E1	(RD).(CLK_EN)
9	E3	E3	(!RD) + (RD).(!CLK_EN)
10	E4	E4	(!RD).(!RE).(!CLK_EN) + (!RD).(RE) + (RD).(!CLK_EN)
11	E4	E1	(!RD).(!RE).(CLK_EN) + (RD).(RE).(CLK_EN)
12	E4	E3	(RD).(!RE).(CLK_EN)

Fonte: QUARTUS, 2020 (adaptado).

Observe que as equações de estado são representadas pelas equações de transição de estados da tabela associada ao grafo (na **Figura A1.5** são destacados em azul as transições do estado **E1**).

f) Recomendações para a implementação da máquina de estados do projeto:

- Utilize o *template* deste **Anexo** (arquivo **UC_SPA_Exemplo.vhd** fornecido no **Moodle**) como referência para a descrição **VHDL** da máquina de estados da Unidade de Controle do sistema de Supervisão de Percurso de Pedágio Aberto;
- Você pode criar um projeto **VHDL** exclusivo para essa máquina de estados e gerar as formas de onda correspondentes;
- Para completar o projeto você pode criar um novo projeto, utilizando a máquina de estados como um "componente" do projeto principal e incluir os elementos do fluxo de dados;
- A interligação dos componentes pode ser realizada por símbolos criados a partir dos códigos VHDL ou diretamente através do código **VHDL** (arquitetura do tipo estrutural). (PRATES, 2019).

ANEXO 2 – EXEMPLO DE ARQUIVO VHDL PARA A UC DO SISTEMA

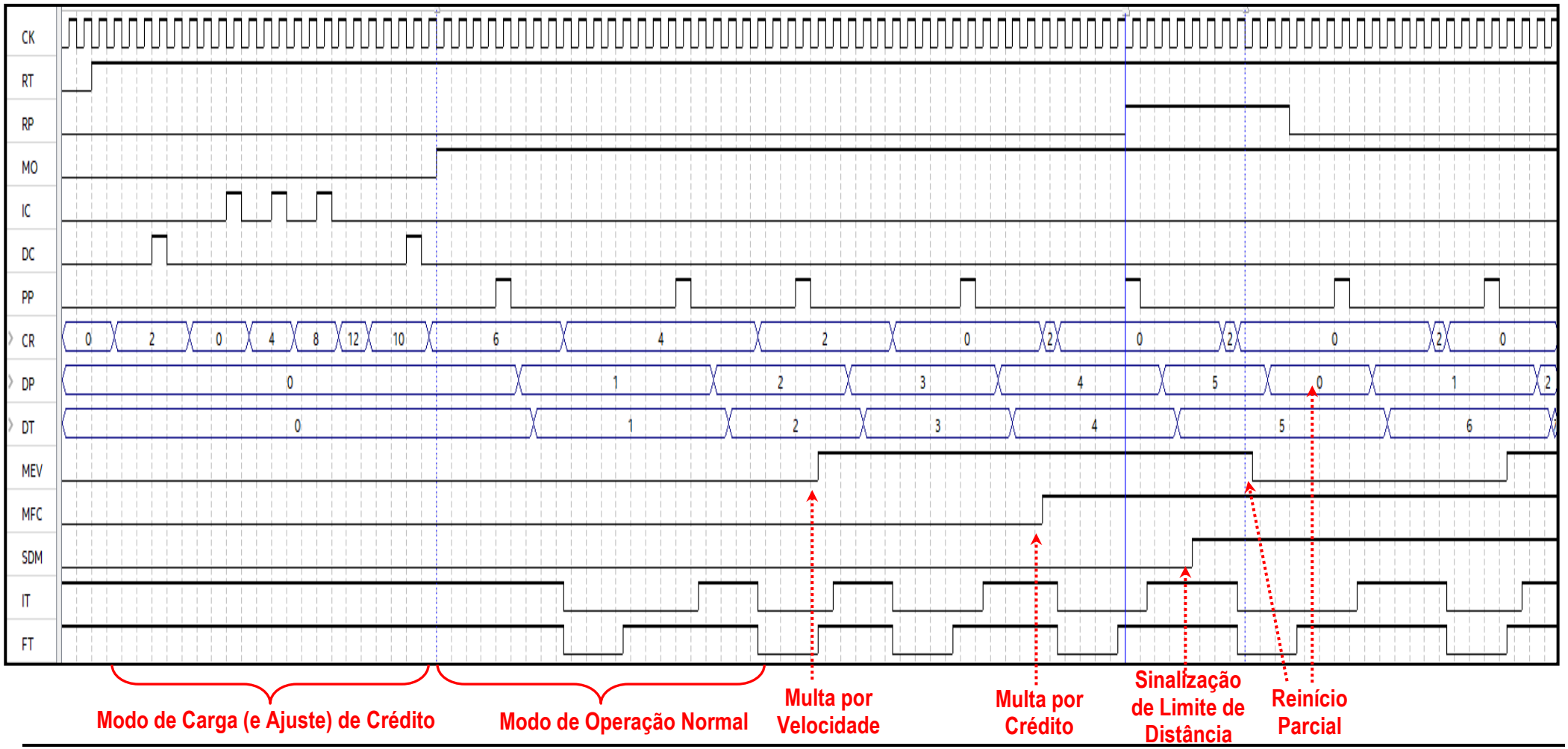
```
14  library ieee;
15  use ieee.std_logic_1164.all;
16  use ieee.std_logic_arith.all;
17  use ieee.std_logic_unsigned.all;
18
19  entity UC_SPA_Exemplo is
20  port( CK : in std_logic;              -- clock de 50MHz
21        RT : in std_logic;              -- reinicio total -> ativo em zero
22  -- Entradas Externas:
23        IC : in std_logic;          -- incrementa credito -> ativo em um
24        DC : in std_logic;          -- decrementa credito -> ativo em um
25        RP : in std_logic;          -- reinicio parcial -> ativo em um
26        PP : in std_logic;          -- passagem pelo portico -> ativo em um
27        MO : in std_logic;          -- modo de operação -> 1= Normal, 0= Ajusta Valor
28  -- Sinais de Estado da ULA:
29        ZR : in std_logic;          -- resultado zero na operação da ULA
30        PO : in std_logic;          -- resultado positivo na operação da ULA
31        NG : in std_logic;          -- resultado negativo na operação da ULA
32        ES : in std_logic;          -- resultado da operação da ULA maior que 255
33  -- Sinais de Estado do TIMER:
34        FT : in std_logic;          -- fim da temporização de 1 segundo
35  -- Sinais de Saida para MUX:
36        Sel_mxa : out std_logic_vector(2 downto 0);  -- seleciona entrada de MUX_A
37        Sel_mxb : out std_logic_vector(2 downto 0);  -- seleciona entrada de MUX_B
38  -- Sinais de Saida para ULA:
39        Sel_ula : out std_logic_vector(1 downto 0);  -- seleciona operação da ULA
40  -- Sinais de Saida para Registradores:
41        Lda   : out std_logic;      -- carrega RA
42        Ldb   : out std_logic;      -- carrega RB
43        Ldc   : out std_logic;      -- carrega RC
44  -- Sinais de Saida para TIMER:
45        IT    : out std_logic;      -- inicia temporização
46  -- Sinais de Saida Externos:
47        MEV   :  out std_logic;     -- sinaliza multa por excesso de velocidade
48        MFC   :  out std_logic;     -- sinaliza multa por falta de crédito
49        SDM   :  out std_logic      -- sinaliza limite de distância total percorrida
50     );
51  end UC_SPA_exemplo;
52  ---------------------------------------------------------------------------------
53  architecture FSM of UC_SPA_Exemplo is
54  type ESTADOS_ME is (ZER_RG, CAR_CR, AJU_CR, ...
55
56              -- COMPLETAR COM OS NOMES DOS ESTADOS DO SEU PROJETO
57
58                    );
59  signal E: ESTADOS_ME;
60  begin
61  process(CK, RT)
62  begin
63  if RT='0' then E <= ZER_RG;                     -- zera registros
64              MFC <= '0'; MEV <= '0'; SDM<='0';   -- zera multas e sinalização
65  elsif (CK'event and CK='1') then
66   case E is
67     when ZER_RG =>
68                 E <= CAR_CR;          -- carrega CR com credito inicial
69     when CAR_CR =>
70                 E <= AJU_CR;          --- espera IC e DC com MO=0 para ajustar CR
71     when AJU_CR =>
72              if MO = '1' and PP='1' then
73                 E <= INC_DP;          -- incrementa distancia parcial
74              elsif ....
75
76              -- COMPLETAR COM AS CONDIÇOES LÓGICAS DOS ESTADOS DA SUA FSM
77
78     when others => Null;
79   end case;
80  end if;
81  end process;
```

ANEXO 2 – EXEMPLO DE ARQUIVO VHDL PARA A UC DO SISTEMA (continuação)

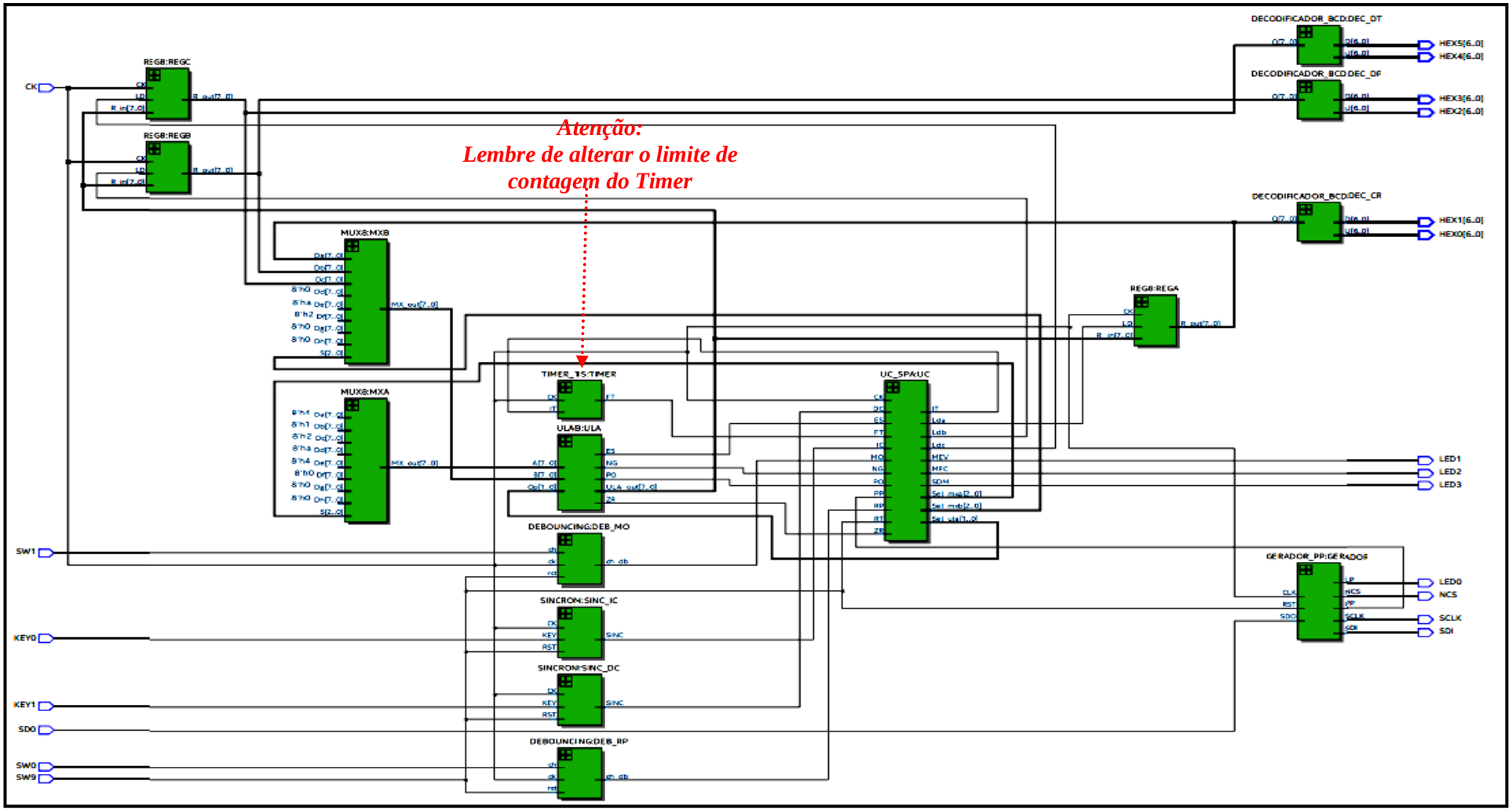
```
 82
 83    -- Atualização das Saídas para Fluxo de Dados (Multiplexadores, Registradores, ULA)
 84    process(E)
 85    begin
 86     case E is
 87       when ZER_RG =>                                -- zera registros
 88                Sel_mxa <= "XXX"; Sel_mxb <= "011"; Sel_ula <= "00";
 89                Ldc <= '1'; Ldb <= '1'; Lda <='1';  IT <= '1';
 90       when CAR_CR =>                               -- carrega CR com Credito Inicial
 91                Sel_mxa <= "XXX"; Sel_mxb <= "101"; Sel_ula  <= "00";
 92                Ldc <= '0'; Ldb <= '0'; Lda <='1'; IT <= '1';
 93       when AJU_CR =>                               -- espera IC e DC para ajustar CR
 94                Sel_mxa <= "XXX"; Sel_mxb <= "XXX"; Sel_ula  <= "XX";
 95                Ldc <= '0'; Ldb <= '0'; Lda <='0';  IT <= '1';
 96
 97                -- COMPLETAR COM OS SINAIS DE CONTROLE DO FLUXO DE DADOS DO SEU PROJETO
 98
 99
100       when others => Null;
101     end case;
102    end process;
103    end FSM;
```

Fonte: Autor.

ANEXO 3 – EXEMPLO DE SIMULAÇÃO PARCIAL DO SISTEMA DE SUPERVISÃO DE PEDÁGIO ABERTO (UC+FD)
TESTE DE REGISTRO DE DISTÂNCIAS E MULTA POR VELOCIDADE (PARÂMETROS: CRI=2, CRM=10, AC=4, TRP=2, LDM=4)



ANEXO 4 – EXEMPLO DE DIAGRAMA RTL DO PROJETO COMPLETO DO SISTEMA DE SUPERVISÃO DE PEDÁGIO ABERTO



ANEXO 5 – NUMERAÇÃO DOS PINOS PARA OS SINAIS DE ENTRADA E SAÍDA DO FPGA: MAX 10 (MODELO: 10M50DAF484C7G) COM OS PERIFÉRICOS DA PLACA DE10-LITE PARA O SISTEMA DE SUPERVISÃO DE PEDÁGIO ABERTO

Tabela 3: Numeração dos pinos para os sinais de entrada e saída do **FPGA** da família **MAX 10** (modelo: **10M50DAF484C7G**)

SINAL	PINO	DISPLAY HEX0	PINO	DISPLAY HEX1	PINO	DISPLAY HEX2	PINO	DISPLAY HEX3	PINO	DISPLAY HEX4	PINO	DISPLAY HEX5	PINO	LEDS	PINO
KEY0 (IC)	PIN_B8	HEX0 [6]	PIN_C17	HEX1 [6]	PIN_B17	HEX2 [6]	PIN_B22	HEX3 [6]	PIN_E17	HEX4 [6]	PIN_F20	HEX5 [6]	PIN_N20	LED0 (LP)	PIN_A8
KEY1 (DC)	PIN_A7	HEX0 [5]	PIN_D17	HEX1 [5]	PIN_A18	HEX2 [5]	PIN_C22	HEX3 [5]	PIN_D19	HEX4 [5]	PIN_F19	HEX5 [5]	PIN_N19	LED1 (MEV)	PIN_A9
SW0 (RP)	PIN_C10	HEX0 [4]	PIN_E16	HEX1 [4]	PIN_A17	HEX2 [4]	PIN_B21	HEX3 [4]	PIN_C20	HEX4 [4]	PIN_H19	HEX5 [4]	PIN_M20	LED2 (MFC)	PIN_A10
SW1 (MO)	PIN_C11	HEX0 [3]	PIN_C16	HEX1 [3]	PIN_B16	HEX2 [3]	PIN_A21	HEX3 [3]	PIN_C19	HEX4 [3]	PIN_J18	HEX5 [3]	PIN_N18	LED3 (SDM)	PIN_B10
SW9 (RT)	PIN_F15	HEX0 [2]	PIN_C15	HEX1 [2]	PIN_E18	HEX2 [2]	PIN_B19	HEX3 [2]	PIN_E21	HEX4 [2]	PIN_E19	HEX5 [2]	PIN_L18	NCS	PIN_AB16
CLK1_50 (CK)	PIN_P11	HEX0 [1]	PIN_E15	HEX1 [1]	PIN_D18	HEX2 [1]	PIN_A20	HEX3 [1]	PIN_E22	HEX4 [1]	PIN_E20	HEX5 [1]	PIN_K20	SCLK	PIN_AB15
SD0	PIN_V12	HEX0 [0]	PIN_C14	HEX1 [0]	PIN_C18	HEX2 [0]	PIN_B20	HEX3 [0]	PIN_F21	HEX4 [0]	PIN_F18	HEX5 [0]	PIN_J20	SDI	PIN_V11

Fonte: TERASIC, 2020 (adaptado).

- **Atenção**: se no seu projeto existirem sinais de saída associados a conectores de saída (OUTPUT) **não listados na Tabela 3** você deve retirar esses conectores (como o QUARTUS PRIME associa automaticamente esses conectores a algum pino de I/O, **pode ocorrer conflito** com outros sinais já utilizados da placa **DE10-Lite**).