UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

DEPARTAMENTO DE TECNOLOGIA

MI - PROJETO DE CIRCUITOS DIGITAIS

JEP: BRINQUEDO AUTÔNOMO

Enzo Cauã da S. Barbosa, Jamile Letícia C. da Silva, Pedro Lucas F. de Souza

Tutor: João Bosco Gertrudes e Wild Freitas da Silva Santos

RESUMO

O relatório detalha o desenvolvimento de um protótipo de brinquedo autônomo em Verilog, realizado no contexto de um projeto acadêmico. O brinquedo, simulado como um cachorrinho, executa uma sequência de ações com velocidades ajustáveis e responde a um sensor de proximidade. O projeto faz uso de circuitos digitais sequenciais e combinacionais, incluindo flip-flops, contadores, divisores de frequência e multiplexadores. Esses componentes foram implementados para controlar a sequência das ações, alternar entre a exibição de diferentes estados e velocidades em displays de sete segmentos e permitir que o brinquedo responda a interações externas, como mudanças de velocidade e a presença de objetos próximos. A solução foi desenvolvida em um dispositivo CPLD e utiliza um sistema de debounce para garantir a estabilidade das leituras de entrada. O relatório discute a metodologia de construção, as especificações técnicas e os resultados obtidos, indicando que o protótipo atende aos requisitos principais, mas sugere melhorias na resposta do sensor de proximidade.

1 INTRODUÇÃO

Ao longo da história da civilização, a sociedade sempre teve a necessidade de construir máquinas e estruturas capazes de automatizar tarefas e atividades lúdicas. Os avanços em engenharia permitiram o surgimento de máquinas e aparelhos capazes de automatizar suas atividades cotidianas. Na era contemporânea, a revolução industrial foi o salto responsável por otimizar e automatizar ainda mais a execução de tarefas, delegando-as às máquinas mecânicas surgidas em meados dos séculos XVIII e XIX (CASTRO, 2024). Assim sendo, no início do século XXI, a revolução industrial entra em sua quarta fase, marcada pelo advento do desenvolvimento digital e inovações como robôs, inteligência artificial e internet das coisas (JUNIOR, 2024).

Nesse sentido, houve grande desenvolvimento social e tecnológico de tal forma que houve mais democratização do acesso aos aparelhos digitais [1], permitindo a integração da esfera social e digital. Essa integração é tão forte que as máquinas digitais passaram a ser sistema interno de vários brinquedos e adentrado ao cotidiano de várias famílias, aumentando a demanda por esse tipo de atividade lúdica.

[1] Ainda que não seja totalmente democrático

Nesse sentido, os autores deste relatório foram convocados, por meio da disciplina de projetos de circuitos digitais, para projetar em circuito digital e linguagem de descrição de hardware *Verilog*, um protótipo de um brinquedo: um cachorrinho capaz de executar um ciclo de ações em tempo definido pelas configurações do usuário. O protótipo deve conter um controle de velocidade que vai permitir o usuário alterar o tempo de execução de cada ação. Ademais, o protótipo deve conter um sensor que faz com que o cachorrinho reinicie o ciclo independente em qual ação ele se encontre.

Para além disso, este relatório é estruturado em introdução, metodologia, que inclui referencial teórico e desenvolvimento, resultado e discussões, conclusão, referências e anexos. Tal estruturação objetiva explicar, de modo sucinto, os recursos e conceitos utilizados para o desenvolvimento do protótipo.

2 METODOLOGIA

2.1 REFERENCIAL TEÓRICO

Nesta seção, são abordados os diferentes tipos de circuitos utilizados no desenvolvimento, suas características, conceitos-chave de projeto e a linguagem de descrição de hardware aplicada. O protótipo foi desenvolvido em sistemas digitais e utiliza tanto circuitos combinacionais quanto sequenciais. Todas as informações desta seção foram obtidas do livro Sistemas Digitais: Princípios e Aplicações, de Ronald J. Tocci, Neal S. Widmer e Gregory L. Moss(TOCCI, 2011).

2.1.1 Circuitos Sequenciais

Circuitos sequenciais são elementos de circuitos lógicos digitais que, ao contrário dos combinacionais, possuem memória e, portanto, suas saídas dependem não só das entradas atuais, mas também do histórico de estados anteriores. Esses circuitos são acionados por um sinal de clock, que sincroniza as mudanças de estado em intervalos regulares, tornando-os ideais para aplicações que requerem armazenamento de dados ou operações controladas em sequência, como contadores, registradores e máquinas de estado finito. Assim, os combinacionais são rápidos e respondem diretamente às mudanças nas entradas, enquanto os sequenciais, acionados por sinais de clock, controlam a atualização de seus estados internos, permitindo operações mais complexas, como em contadores e máquinas de estado.

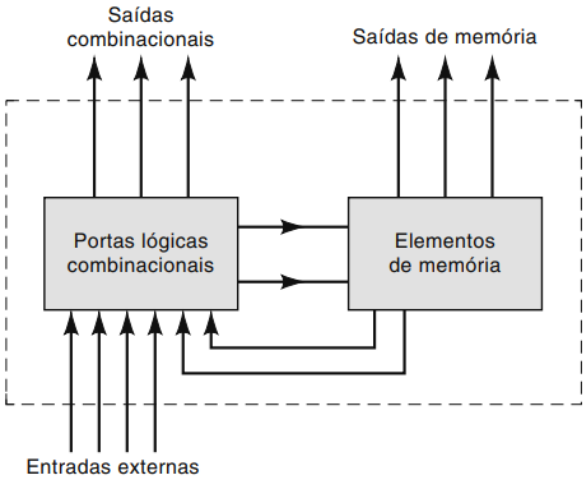

Figura 1 – Diagrama Geral de um sistema digital. Fonte: (TOCCI, 2011).

2.1.2 Latches e Flip Flops

Latches e flip-flops são componentes de memória digital que armazenam um bit de informação, mas diferem na forma como capturam e mantêm os dados. Ambos são essenciais para circuitos sequenciais, mas os latches são dispositivos transparentes, enquanto os flip-flops são acionados por borda de clock. Em sistemas digitais, esses elementos permitem o armazenamento e a manipulação de dados em sincronia com um sinal de clock, tornando-os úteis em contadores, registradores e outros circuitos sequenciais.

Os latches são dispositivos que mantêm a saída em um estado enquanto a entrada de controle estiver ativa, ou seja, enquanto o sinal de habilitação está presente. Existem diversos tipos, como o latch S-R, que usa duas entradas para setar ou resetar a saída, e o latch D, que é acionado por uma entrada de dados e mantém o último valor de entrada enquanto o controle é ativado.

Os flip-flops, por outro lado, são dispositivos que só alteram o estado da saída na borda do sinal de clock, seja na borda de subida ou de descida. Entre os tipos principais estão o flip-flop D, que armazena o valor da entrada D na transição do clock, e o flip-flop J-K, que permite operações mais complexas, como alternar entre estados. A diferenciação entre latches e flip-flops, além da sincronização por clock, está na capacidade dos flip-flops de evitar estados ambíguos em circuitos sequenciais síncronos.

O projeto foi desenvolvido utilizando Flip Flops do tipo D (ou Data Flip-Flop), que é um tipo de flip-flop simples, com apenas uma entrada de dados (D) e uma entrada de clock (CLK). Sua principal função é armazenar o valor da entrada D no instante em que ocorre uma borda ativa do sinal de clock, mantendo esse valor na saída Q até a próxima borda do clock. Isso faz com que o flip-flop D seja muito útil em aplicações de armazenamento e sincronização de dados, pois ele permite a retenção de um estado específico durante intervalos de tempo predeterminados.

Esse flip-flop foi escolhido, pois, ao contrário de outros tipos de flip-flops, como o S-R e o J-K, o flip-flop D não possui estados indeterminados, o que o torna mais confiável e fácil

de usar em sistemas digitais. Na borda de subida do clock, a saída Q assume o mesmo valor da entrada D. Esse comportamento permite que o flip-flop D funcione como uma memória de um bit que só muda de estado em resposta à borda do clock, evitando alterações indesejadas em outros momentos.

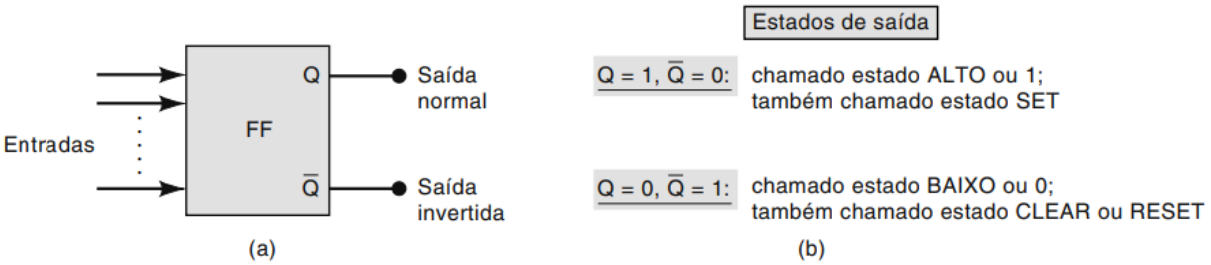

Figura 2 – Símbolo geral para um flip-flop e definição dos seus dois estados de saída possíveis. Fonte: (TOCCI, 2011).

2.1.3 Divisão de Frequência

A divisão de frequência é um processo em que a frequência de um sinal de entrada é reduzida ao ser dividida por um fator específico. Em circuitos digitais, especialmente em contadores binários, essa divisão ocorre quando a saída de um flip-flop alterna entre altos e baixos em resposta aos pulsos de clock, produzindo uma frequência que é uma fração da frequência de entrada. Cada flip-flop em uma cadeia de contagem divide a frequência de entrada pela metade, de modo que, ao final de uma série de N flip-flops, a frequência de saída é reduzida para $(1/2)^n$ da frequência inicial.

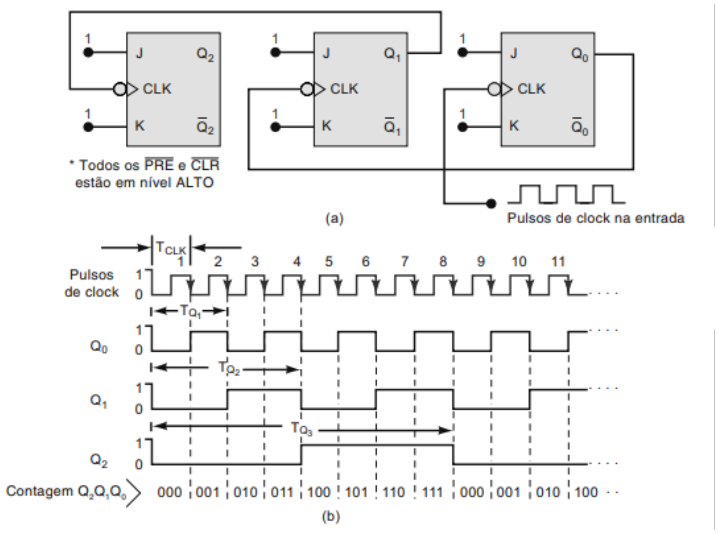

Figura 3 – Flip-flops J-K conectados para formar um contador binário de três bits. Fonte: (TOCCI, 2011).

2.1.4 Contagem

A contagem em sistemas digitais é realizada por meio de flip-flops dispostos em sequência para formar um contador, que avança sua contagem a cada pulso de clock. Os contadores podem ser configurados para contar em sistemas binários, decimais (BCD) ou qualquer outro sistema numérico. Um exemplo comum é o contador de módulo 8, que possui três flip-flops e conta de 0 a 7 em binário antes de retornar ao estado inicial. Cada flip-flop muda seu estado na borda do pulso de clock, e o último flip-flop da cadeia representa o bit mais significativo. Contadores são amplamente usados para controlar operações temporizadas e são fundamentais em temporizadores, sequenciadores e sistemas de medição de frequência.

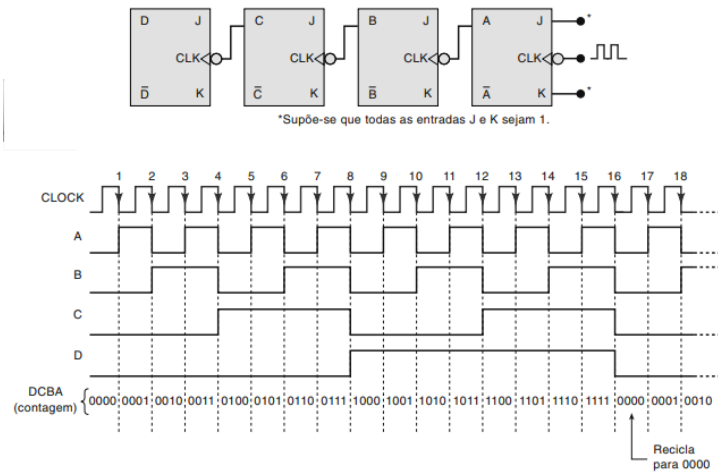

Figura 4 – Contador assíncrono (ondulante) de quatro bits. Fonte: (TOCCI, 2011).

2.1.5 Multiplexador

Multiplexadores, ou seletores de dados, são circuitos que desempenham uma função crucial em sistemas digitais ao permitir a seleção de um entre vários sinais de entrada para ser transmitido à saída, de acordo com as entradas de controle ou seleção. Sua estrutura básica inclui diversas entradas de dados, uma única saída e um conjunto de entradas de seleção, que especificam qual sinal de entrada será direcionado para a saída.

Os multiplexadores são amplamente utilizados em displays de 7 segmentos para controlar quais dígitos ou segmentos estarão ativos em determinado momento. Em sistemas que utilizam múltiplos displays, é comum empregar multiplexação para alternar rapidamente entre os dígitos, iluminando um de cada vez em sequência rápida. Esse processo, conhecido como “multiplexação de displays”, utiliza multiplexadores para enviar os dados corretos a cada display individual enquanto alterna a alimentação entre eles, dando a impressão de que todos os dígitos estão acesos simultaneamente.

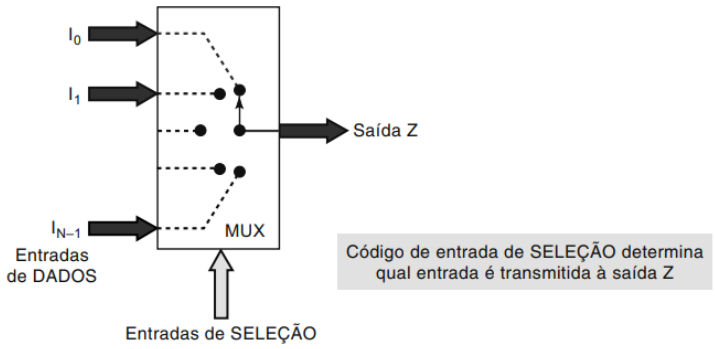

Figura 5 – Diagrama funcional de um multiplexador (MUX) digital. Fonte: (TOCCI, 2011).

2.1.6 Debounce

O conceito de debounce se refere à eliminação do efeito de "trepidação"(contact bounce) causado por contatos mecânicos ao serem acionados, como em uma chave. Em sistemas digitais, ao acionar uma chave mecânica, múltiplas transições de contato podem ocorrer em frações de segundo, gerando vários pulsos indesejados que podem ser interpretados como múltiplos acionamentos pelo sistema. Para evitar isso, técnicas de debounce utilizam circuitos eletrônicos, como flip-flops ou buffers, ou abordagens em software que garantem a interpretação de apenas uma transição limpa. Dessa forma, o debounce permite que sistemas digitais respondam de forma confiável a comandos de acionamento.

2.2 ESPECIFICAÇÕES DA LINGUAGEM VERILOG

O Verilog é uma linguagem de descrição de hardware (HDL) amplamente utilizada para o projeto de circuitos digitais. Ele permite que designers descrevam sistemas em um formato textual, que pode ser facilmente traduzido para implementações físicas. Entre as principais características do Verilog está sua portabilidade. A linguagem oferece um amplo suporte entre diferentes tecnologias de hardware, permitindo que projetos sejam implementados em vários tipos de chips e plataformas de ferramentas de automação (CAD) sem necessidade de alterações. Todas as informações foram desta seção obtidas do livro Fundamentals of Digital Logic with Verilog Design de Stephen Brown e Zvonko Vranesic (BROWN; VRANESIC, 2014).

Outra característica essencial do Verilog é a flexibilidade na representação de circuitos, que pode ser feita de duas maneiras: estrutural e comportamental. A descrição estrutural foca em como os circuitos estão interconectados, detalhando a lógica específica do circuito. Por outro lado, a descrição comportamental permite que o circuito seja especificado com base em seu comportamento desejado, utilizando um nível mais alto de abstração, sem detalhar o layout físico dos elementos.

A descrição comportamental em Verilog permite que o circuito seja especificado com base em seu comportamento lógico desejado, em vez de detalhes de interconexão física. Essa abordagem é particularmente útil para modelar o funcionamento de circuitos sequenciais, como flip-flops e registradores, que dependem de um sinal de clock para mudar de estado. Em Verilog, os blocos comportamentais geralmente são definidos usando o bloco always, que permite especificar

quando o estado de um componente deve ser atualizado com base em transições de sinais, como bordas do clock.

A manipulação do clock em descrições comportamentais é essencial para implementar circuitos sincronizados. Por exemplo, ao usar uma lista de sensibilidade com @(posedge Clock), o Verilog reage apenas na borda de subida do clock. Isso é comumente usado em flip-flops D, onde a mudança de estado ocorre exatamente no momento da borda positiva do clock. Além disso, o uso de instruções não-bloqueantes (<=) em blocos sincronizados ao clock permite que todos os valores de saída sejam atualizados simultaneamente ao final do ciclo, evitando dependências de ordem entre as atribuições.

2.2.1 Bloco Always

O bloco always em Verilog é um elemento central para a descrição de circuitos sequenciais e combinações. Ele permite ao designer definir a lógica de um circuito que precisa ser reavaliada ou atualizada em resposta a eventos específicos, como alterações em sinais de entrada ou transições de borda de um sinal de clock. Em um circuito combinacional, o bloco always é sensível a mudanças em qualquer sinal envolvido na expressão lógica.

Para circuitos sequenciais, o bloco always é fundamental para descrever a lógica sincronizada ao clock. O bloco é configurado para reagir em bordas específicas do sinal de clock, geralmente usando always @(posedge Clock) para bordas de subida ou always @(negedge Clock) para bordas de descida. Esse comportamento é especialmente útil para definir flip-flops e registradores, que mudam de estado apenas em momentos específicos do ciclo do clock.

2.2.2 Atribuições Bloqueantes e Não Bloqueantes

Em Verilog, as atribuições bloqueantes (=) e não bloqueantes (<=) são usadas para definir o comportamento de variáveis em diferentes contextos dentro do bloco always. Essas duas formas de atribuição afetam a execução das instruções e a atualização dos valores das variáveis. As atribuições bloqueantes (=) são executadas de forma sequencial e imediata dentro do bloco always. Isso significa que cada instrução deve ser completamente avaliada e concluída antes que a próxima seja executada. Esse comportamento é útil em lógica combinacional, onde se espera que as instruções sejam realizadas em uma ordem específica, e o resultado de uma instrução pode influenciar as próximas imediatamente.

As atribuições não bloqueantes (<=), por outro lado, são comumente usadas em lógica sequencial e sincronizada com o clock, como em flip-flops e registradores. Essas atribuições permitem que todas as instruções dentro do bloco always sejam "agendadas"para execução, de modo que os valores das variáveis sejam atualizados somente ao final do ciclo de clock. Esse comportamento evita a dependência direta de execução entre as linhas de atribuição e garante que as variáveis sejam atualizadas em paralelo.

2.2.3 Variáveis “REG”

Em Verilog, o tipo reg é usado para armazenar valores em variáveis que podem ser atribuídas em blocos procedurais, como o bloco always. As variáveis do tipo reg em Verilog

mantêm seus valores até que sejam explicitamente alteradas, permitindo que persistam entre ciclos do clock em circuitos sequenciais. Elas só podem ser atribuídas dentro de blocos procedurais, como always ou initial, o que proporciona um controle preciso sobre quando e como seus valores são atualizados.

Apesar do nome “reg”, que pode sugerir um “registrador” (register), esse tipo não se limita a circuitos de armazenamento. Ele é usado tanto para lógica combinacional quanto sequencial. Em blocos combinacionais (always @*), o reg armazena o resultado de operações lógicas ou aritméticas, reagindo a mudanças nas entradas, enquanto, em lógica sequencial, ele representa flip-flops controlados por um sinal de clock, retendo dados entre ciclos. Além disso, variáveis “reg” podem ter tamanhos específicos, definidos durante a declaração, permitindo armazenar diferentes números de bits e evitando truncamentos em operações aritméticas. Essa versatilidade faz do “reg” uma ferramenta fundamental na modelagem de diversos tipos de circuitos digitais.

2.3 DESENVOLVIMENTO

O protótipo foi desenvolvido em circuito integrado CPLD (Complex Programmable Logic Device) pertencente à família MAX II, modelo EPM240T100C5N, disponível no Laboratório de Eletrônica Digital e Sistemas (LEDS) da Universidade Estadual de Feira de Santana (DIAS, 2024). A placa é denominada CPLD-LEDS. O protótipo é projetado em linguagem de descrição de hardware *Verilog Estrutural* no software Quartus (Intel, 2024). O *Verilog Estrutural* é adotado mundialmente como linguagem de descrição de hardware e possui dependências para declarar portas lógicas e permitir associações entre entradas, fios e saídas (TOCCI, 2011).

O desenvolvimento do protótipo baseou-se na combinação módulos lógicos, contendo a combinação das portas logicas a partir das entradas requisitadas no problema. Tais entradas simulam as chaves de seleção e o sensor do brinquedo, que definem sua velocidade de movimento, o botão de ligar/desligar e o reset da movimentação. Dessa maneira, o protótipo foi projetado para receber 3 entradas de variáveis, combinadas logicamente, juntamente com o clock disponível na placa para compor as possíveis saídas discutidas nas sessões tutoriais.

As saídas foram pensadas para simular e informar ao usuário a execução de cada uma das 6 ações do brinquedo, sendo elas: andar, agachar, latir, saltar, balançar o rabinho e dar meia volta. As saídas também sinalizam qual a velocidade de execução selecionada, sendo elas: 0s (parado), 2s, 4s ou 8s.

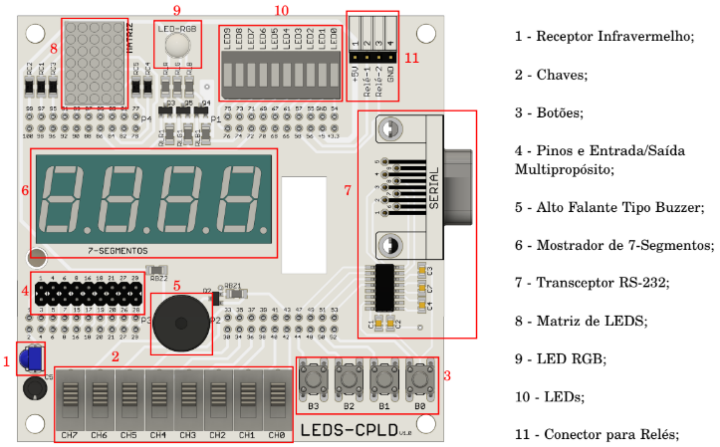

Figura 6 – Visão geral da placa CPLD-LEDS. Fonte: (DIAS, 2024)

Nesse sentido, os principais componentes eletrônicos dispostos na placa CPLD-LEDS, mostrado na figura 6, foram os chaves (2), botões (3), mostrador de 7 segmentos (6) e o LED RGB (9).

Chaves	Entrada
CH0	Chave de ligar/desligar
B0	Sensor de Proximidade
B3	Seletor de Velocidade

Tabela 1 – Tabela de correspondência entre canais e entrada.

As chaves são responsáveis por simular as entradas do brinquedo, em que as possíveis combinações dos estados das chaves (0 ou 1) determinam saídas diferentes. As chaves são pinadas conforme os dados da tabela 1. Pela quantidade de entradas, considera-se 8 possibilidades de saída. Dessas, 4 possibilidades representam o estado em que o chave de ligar/desligar está em estado 0 e em lógica conjuntiva com os demais componentes lógicos, isto é, o protótipo não realiza simulações de movimentação. Além disso, o sensor de proximidade apresenta lógica semelhante, em lógica conjuntiva com os demais componentes do circuito. Isso significa que o sensor e o chave de ligar/desligar são hierarquicamente superiores em relação ao restante do circuito. Nesse sentido, o brinquedo inicia seu movimento até que haja três condições: botão de ligar/desligar em 0; sensor em estado 1, indicando objeto próximo, e velocidade igual a 0.

Para as questões de movimento, o protótipo sempre executa as ações em sequência, de acordo com a velocidade, na seguinte ordem: andar, agachar, latir, saltar, balançar o rabinho e dar meia volta. O brinquedo só altera essa sequência caso a velocidade seja definida em zero, assim, ele para a execução da ação, ou, então, caso o sensor seja acionado, ele dá meia volta, independente da ação em que estiver, e retorna para a primeira ação. Todas as ações do brinquedo são exibidas no display de 7 segmentos (Figura 7), sendo as velocidades exibidas no digito mais a esquerda e as ações no digito mais a direita. Além disso, é utilizado o LED RGB (Figura 8) para

indicar o funcionamento do brinquedo, com a cor azul indicando que está ligado e a cor roxa indicando que o sensor de proximidade foi acionado.

Figura 7 – Display 7 segmentos. Fonte: (DIAS, 2024)

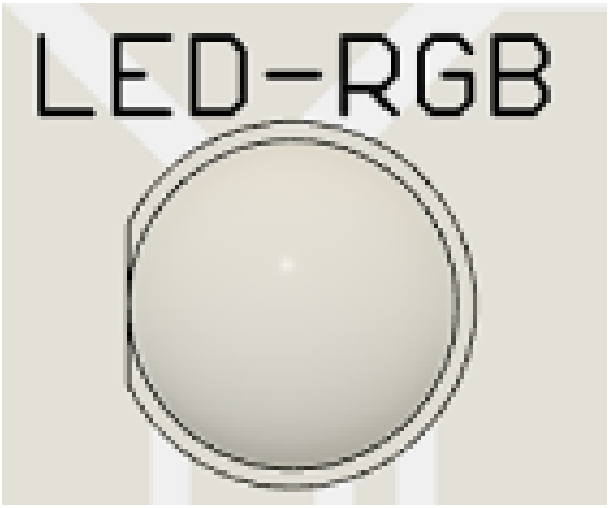

Figura 8 – LED RGB. Fonte: (DIAS, 2024)

3 RESULTADOS E DISCUSSÕES

3.1 ASPECTO TÉCNICO

A nível de desenvolvimento técnico, o código em linguagem de descrição de hardware foi construído em 6 módulos de código principais, sendo eles: um divisor de frequência, um Contador seletor de velocidade, dois contadores assíncronos, um multiplexador de velocidade e um modulo do display. Cada módulo é responsável por executar uma determinada função a partir das suas entradas e saídas, relacionando-as por meio de declarações de portas lógicas e de outros módulos secundários.

3.1.1 Divisor de Frequência

O divisor de frequência foi um modulo projetado para dividir a frequência de um clock de entrada de 50 MHz utilizando contadores assíncronos e flip-flops D. A divisão é realizada por múltiplos de 5 e 2, sendo útil para aplicações em que diferentes partes de um circuito necessitam

operar em frequências mais baixas. Tal abordagem foi escolhida, pois era necessário alcançar um clock de exatamente 1hz, sendo inviável de ser alcançado utilizando divisores de frequência contendo apenas flip-flops encadeados. Esse modulo utiliza instancias de outros dois módulos.

O módulo *dividir_5* realiza uma divisão do clock de entrada por 5, utilizando três flip-flops D organizados para contar até 5. Em cada ciclo de clock, o primeiro flip-flop (b0) muda seu estado a cada pulso de clock, enquanto o segundo flip-flop (b1) é acionado pelo clock invertido do primeiro flip-flop, alternando em uma frequência mais baixa. O terceiro flip-flop (b2) é acionado pelo clock invertido do segundo, permitindo que o contador complete a sequência até 5. O reset é ativado quando q[0] e q[2] estão em 1, indicando que o contador atingiu o valor 5, momento em que o módulo reinicia a contagem. A saída clk_out corresponde ao bit mais significativo do contador (q[2]), que representa um clock com frequência 1/5 da frequência de entrada.

O módulo principal, *divisor_frequencia*, utiliza diversas instâncias do *dividir_5* junto com uma série de flip-flops D para realizar divisões adicionais na frequência do clock. O clock de entrada passa inicialmente pela primeira instância de *dividir_5*, que divide a frequência por 5 e envia essa saída para a próxima instância, onde o processo se repete em cascata ao longo de oito instâncias de *dividir_5*. Cada uma delas recebe a saída da anterior e continua a dividir a frequência.

Depois que o sinal de clock é reduzido por estas oito instâncias de *dividir_5*, ele passa por uma série de flip-flops D adicionais, que dividem a frequência em fatores adicionais de 2. A divisão completa resulta em três clocks de saída: *clk_out*, que fornece um clock com frequência reduzida para ser utilizado em outras partes do sistema, além de *clk_aux* e *clk_botao*, clocks auxiliares com frequências intermediárias que são utilizadas diferente partes do circuito como no *debounce* do botão, na exibição dos displays de 7 segmentos e no tempo de ação.

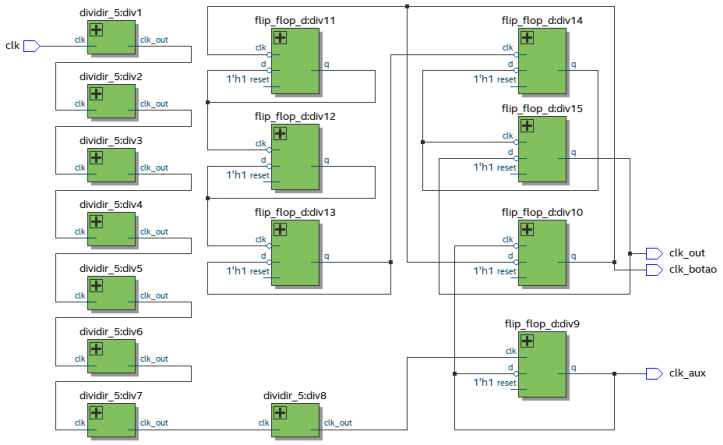

Figura 9 – Modulo Divisor de Frequência. Fonte: Autoria Própria.

3.1.2 Contador Seletor de Velocidade

O modulo de contador seletor de velocidade pode ser considerado um tipo de contador, embora seja mais especificamente projetado para realizar o debouncing do botão e controlar a velocidade das ações baseadas no estado desse botão. Ele utiliza flip-flops D para armazenar

estados e realizar uma ação de debounce no botão de entrada (botao), reduzindo a interferência causada por ruído quando o botão é pressionado.

O modulo *botao_velociade* funciona da seguinte forma: a primeira parte do código, composta pelos flip-flops q[0] e q[1] juntamente com a lógica and And0(sinal, q[0], !q[1]), atua como um debounce para o botão. Isso significa que ele filtra mudanças rápidas ou ruídos no sinal de entrada, garantindo que o sistema detecte apenas uma transição estável. Embora esse processo não seja exatamente uma contagem, ele gera um pulso limpo sempre que o botão é pressionado.

Na segunda parte do código, os flip-flops q[2] e q[3] atuam de maneira semelhante a um contador binário de dois bits. Esses flip-flops, atribuídos às saídas chave_0 e chave_1, alternam em resposta ao sinal de debounce (sinal). Assim, cada vez que o botão é pressionado, q[2] e q[3] mudam de estado em uma sequência previsível: chave_0 (q[2]) inverte a cada pulso de sinal, enquanto chave_1 (q[3]) muda de estado com uma frequência ainda mais lenta, sendo acionada pelo estado de q[2].

Esse comportamento é semelhante ao de um contador binário de 2 bits, onde chave_0 e chave_1 formam uma sequência de contagem binária (00, 01, 10, 11). No entanto, ao contrário de um contador convencional acionado por um clock contínuo, este contador "passo-a-passo"avança na contagem em resposta a cada acionamento do botão. Portanto, embora o código tenha como objetivo principal o controle de velocidade das ações e o debouncing, ele apresenta características de um contador binário que se incrementa a cada pressão do botão.

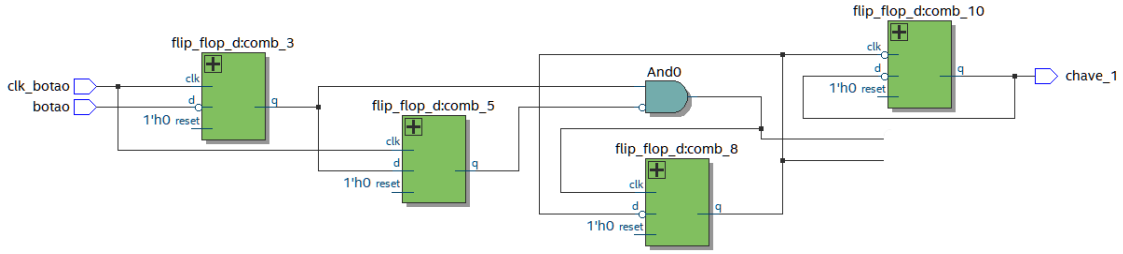

Figura 10 – Modulo Seletor de Velocidade. Fonte: Autoria Própria.

3.1.3 Contador Assíncrono

O modulo de contador assincrono foi projetado para fazer uma contagem binária conforme os ciclos de clock, e ele é resetável por um sinal externo. O contador assíncrono foi implementado usando flip-flops D encadeados.

O módulo *cadeamento* recebe como entradas um sinal de clock (‘clk‘) e um sinal de reset (‘reset‘), que reinicia a contagem. As saídas são os bits individuais de contagem (‘b3‘, ‘b2‘, ‘b1‘ e ‘b0‘), representando o estado atual do contador em formato binário. Dentro do módulo, os flip-flops D (*flip_flop_d*) são conectados de forma encadeada, com o objetivo de formar o contador. O primeiro flip-flop (‘bit0‘) inverte seu estado a cada pulso de clock. Os flip-flops subsequentes (‘bit1‘, ‘bit2‘, e ‘bit3‘) são acionados pelos clocks invertidos dos flip-flops anteriores, o que cria uma sequência de contagem com frequência cada vez menor.

Cada flip-flop armazena seu estado em um fio (‘T0‘ a ‘T3‘), e as saídas do módulo (‘b0‘, ‘b1‘, ‘b2‘, ‘b3‘) são atribuídas a esses fios, representando o valor atual do contador binário. O sinal de reset, quando ativado, define os estados dos flip-flops para zero, reiniciando a contagem. O módulo *cadeamento* é instanciado duas vezes no circuito principal, uma para contar o tempo de execução cada ação (0s, 2s, 4s, 8s) e a outra para contar as ações (0, 1, 2, 3, 4, 5).

Este contador assíncrono é útil para dividir a frequência de um sinal de clock, pois cada bit de saída alterna em frequências sucessivamente menores. Isso o torna ideal para circuitos digitais onde é necessário contar pulsos ou gerar frequências divididas para sincronização ou controle.

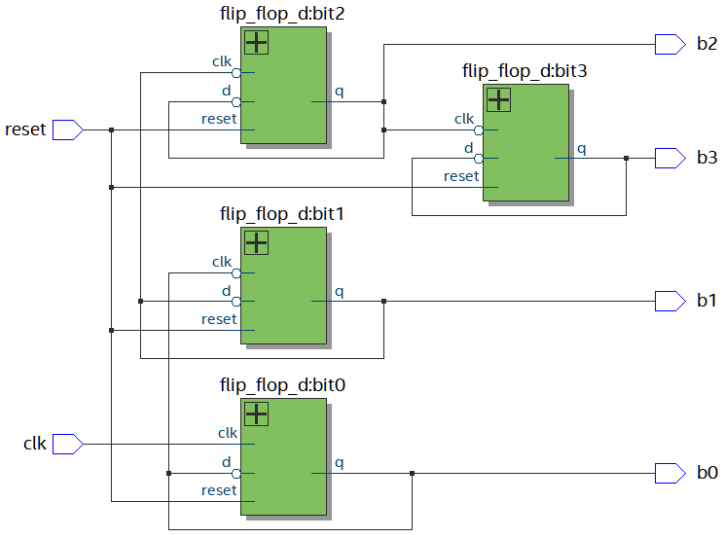

Figura 11 – Modulo Contador Assíncrono. Fonte: Autoria Própria.

3.1.4 Multiplexador de Velocidades

O módulo multiplexador de velocidades tem como finalidade realizar uma contagem e controlar a ação do sistema, por meio de resets acionados quando certas condições de contagem são atendidas. O funcionamento do módulo começa com suas entradas, que incluem chave_0 e chave_1, sinais de controle, além de b3, b2, b1 e b0, que representam o estado do contador. A saída, chamada reset_mux, é acionada para resetar a contagem e alterar a ação do sistema.

Dentro do módulo *mux*, três contadores são utilizados: contar_2, contar_4 e contar_8. Cada um deles é responsável por detectar se a contagem atual é igual a 2, 4 ou 8, respectivamente. O sinal reset_2 é acionado quando a contagem atinge 2, reset_4 quando atinge 4 e reset_8 quando atinge 8. Esses contadores monitoram as variáveis b1, b2 e b3, que são os bits mais significativos do contador. As saídas dos três contadores (reset_2, reset_4, reset_8) são então conectadas a uma porta lógica OR, que gera o sinal reset_mux. Dessa forma, sempre que qualquer um dos contadores detectar a contagem desejada, que pode ser 2, 4 ou 8, o reset_mux será ativado.

O propósito principal do módulo *mux* é integrar os sinais de reset gerados por contagens específicas, permitindo que o sistema reaja de maneira apropriada, como resetar a contagem e

aumentar a ação do brinquedo em uma unidade. Assim, o módulo serve como um componente chave para o controle velocidade de excução de cada uma das ações.

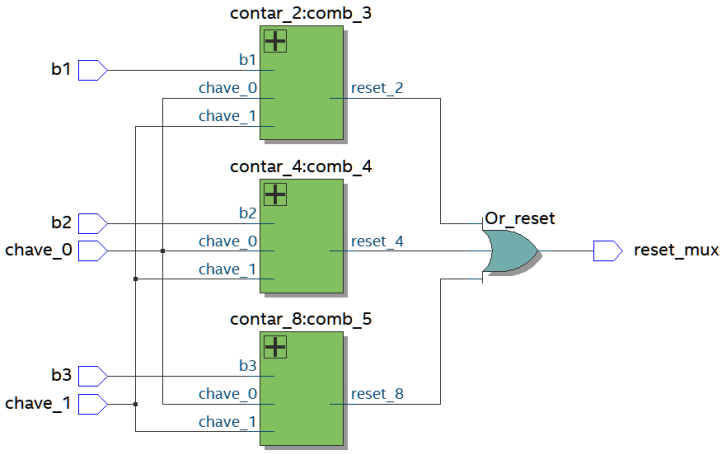

Figura 12 – Modulo Multiplexador de Velocidades. Fonte: Autoria Própria.

3.1.5 Controle do Display

O modulo do display é projetado para controlar a exibição alternada de duas informações em um display de sete segmentos: a contagem de ação e a velocidade de um brinquedo. A estrutura é composta por três módulos principais que trabalham juntos para multiplexar e exibir esses valores de maneira eficaz. O primeiro módulo, *display_contagem_acao*, gera os sinais para exibir a contagem de ação com base em quatro entradas binárias (b3, b2, b1 e b0). Essas entradas são combinadas com operações lógicas para definir quais segmentos (Tseg_a a Tseg_g) do display devem ser ativados para representar diferentes números de acordo com o valor da contagem. O segundo módulo, *display_velocidade*, é responsável por gerar os sinais correspondentes à velocidade em que o brinquedo se encontra, com base em duas entradas (chave_0 e chave_1). Através de uma lógica combinacional usando portas *and, or* e *not*, este módulo determina quais segmentos (Vseg_a a Vseg_g) do display serão ativados para formar números específicos que representam a velocidade.

O módulo principal, *display*, integra as informações dos módulos de contagem de ação e velocidade e utiliza um conjunto de multiplexadores para alternar a exibição entre os dois valores no display de sete segmentos. Para isso, o módulo *display* instancia *display_contagem_acao* e *display_velocidade*, cujas saídas são conectadas a um conjunto de sete módulos *mux_display*. Cada *mux_display* controla um dos segmentos do display e utiliza um sinal de clock auxiliar (clk_aux) com frequência de 64 Hz como chave de seleção, que alterna a exibição entre os sinais de contagem (Tseg_) e velocidade (Vseg_). Esse clock rápido cria um efeito visual em que o display parece mostrar ambos os valores simultaneamente devido à persistência da visão, já que a alternância entre os valores ocorre em uma velocidade que o olho humano percebe como contínua.

O sistema completo permite que as informações de contagem e velocidade sejam exibidas alternadamente no mesmo display, criando uma exibição multiplexada e eficiente. Dessa forma, os módulos trabalham juntos para alternar entre os valores de forma a exibir as informações de

maneira clara e sem a necessidade de múltiplos displays, proporcionando uma solução compacta e eficaz para a exibição de informações dinâmicas em um brinquedo.

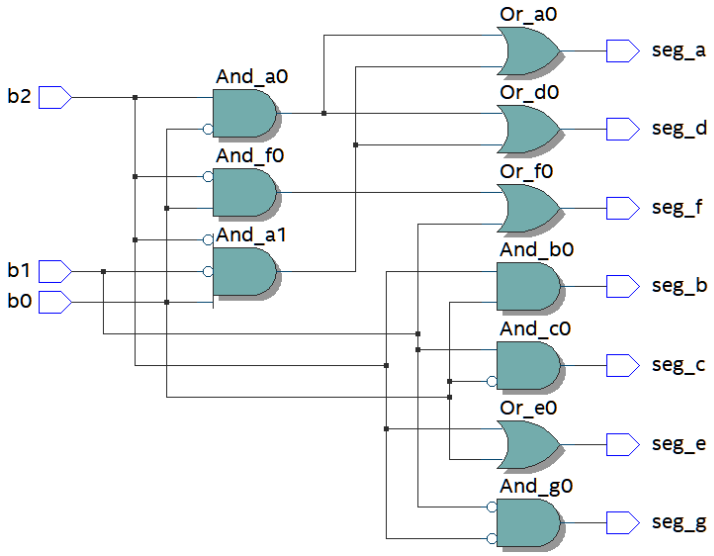

Figura 13 – Modulo Seletor de Contagem das Ações. Fonte: Autoria Própria.

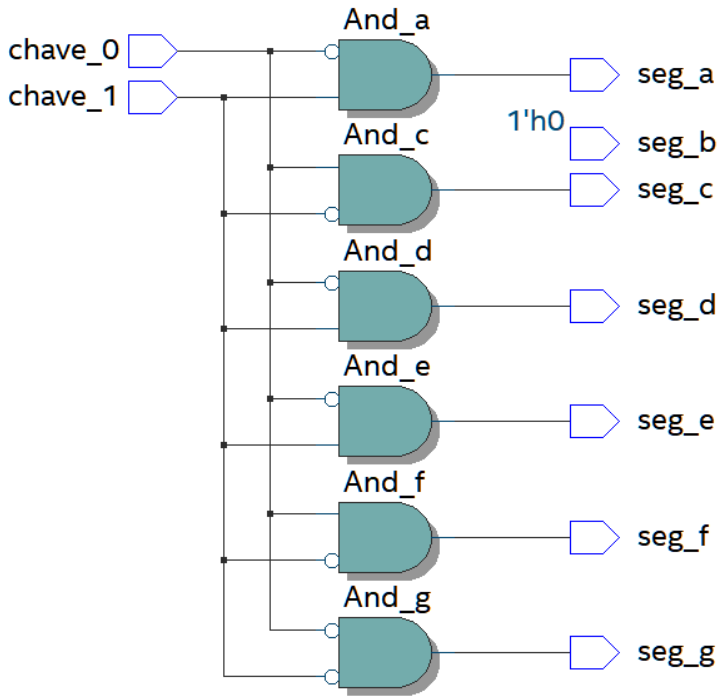

Figura 14 – Modulo Seletor de Contagem das Velocidades. Fonte: Autoria Própria.

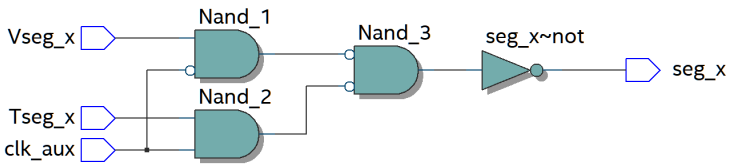

Figura 15 – Modulo Multiplexador dos Segmentos do Display. Fonte: Autoria Própria.

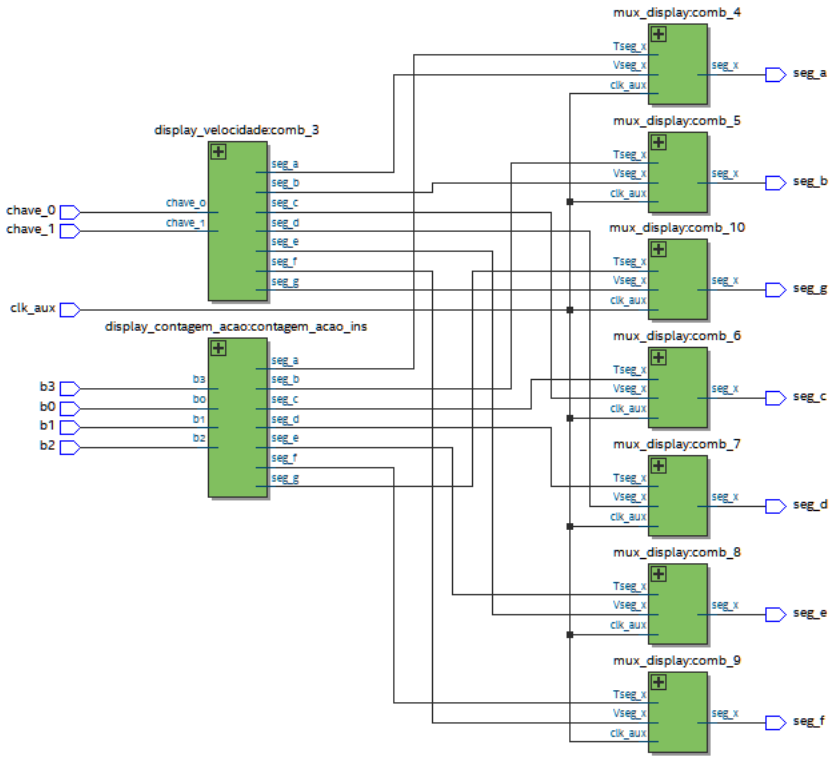

Figura 16 – Modulo do Display de 7 segmentos. Fonte: Autoria Própria.

4 CONCLUSÃO

Diante do exposto e considerando os testes realizados, nota-se que o protótipo proposto, o qual simula um brinquedo autônomo, foi desenvolvido de maneira satisfatória, uma vez que os requisitos principais foram atendidos. Sendo esses requisitos: selecionar a velocidade por ação do brinquedo manualmente; a chave liga/desliga; o botão que simula o sensor de proximidade; e a exibição da ação que está sendo executada e da velocidade selecionada nos displays de 7 segmentos.

Dessa forma, um sistema digital constituído por circuitos sequenciais e combinacionais foi feito, a fim de solucionar o problema. Para isso, recursos como flip-flops, contadores, divisores de frequência e multiplexadores foram utilizados. Ademais, o protótipo foi implementado no kit de desenvolvimento LEDS-CPLD, fazendo uso de uma chave, dois botões, dois displays de sete seguimentos e o LED RGB.

Nessa perspectiva, pode-se considerar o produto eficaz, porém, é válido destacar que ele pode ser melhorado. Pois, apesar do sensor de proximidade ter sido implementado, ele não funciona de forma completamente eficiente. Assim, torna-se necessária uma reformulação da implementação desse sensor, visando adicionar na simulação, quando ele é ativado, a exibição da ação número cinco do brinquedo (dar meia volta), uma vez que, durante o tempo em que essa ação está sendo executada, é exibido o número zero, que corresponde à ação andar. Com isso, será obtido um melhor desempenho do protótipo.

REFERÊNCIAS

BROWN, S.; VRANESIC, Z. *Fundamentals of Digital Logic with Verilog Design*. 3rd. ed. [S.l.]: McGraw-Hill, 2014. Citado na página 6.

CASTRO, L. L. de. *Revolução Industrial*. 2024. Toda Matéria. Disponível em: <https://www.todamateria.com.br/revolucao-industrial/>. Acesso em: 13 de Setembro de 2024. Citado na página 1.

DIAS, A. M. *Kit de desenvolvimento LEDS-CPLD*. [S.l.], 2024. Disponível em: <https://drive.google.com/file/d/1mqm-I2mHjwPsXzaypq_S3-uXIpvc42o5/view>. Acesso em: 13 de Setembro de 2024. Citado 3 vezes nas páginas 8, 9 e 10.

Intel. *Software de projeto Quartus Prime*. [S.l.], 2024. Disponível em: <https://www.intel.com.br/content/www/br/pt/products/details/fpga/development-tools/quartus-prime.html>. Acesso em: 13 de Setembro de 2024. Citado na página 8.

JUNIOR, J. M. F. *Quarta Revolução Industrial*. 2024. Brasil Escola. Disponível em: <https://brasilescola.uol.com.br/historiag/quarta-revolucao-industrial.htm>. Acesso em: 13 de Setembro de 2024. Citado na página 1.

TOCCI, R. J. *Sistemas Digitais - Princípios e Aplicações. Ed 11ª*. [S.l.]: Pearson Prentice Hall, 2011. Citado 6 vezes nas páginas 2, 3, 4, 5, 6 e 8.

Anexos

A TABELAS VERDADE E EXPRESSÕES DOS MÓDULOS

A.1 DIVISOR DE FREQUÊNCIA

Ciclo de Clock	clk_out	clk_aux	clk_botao
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0
9	0	0	1
10	0	1	0
11	0	1	1
12	1	0	0
13	1	0	1
14	1	1	0
15	1	1	1

Tabela 2 – Módulo Divisor de Frequência

Entradas

- *clk*

Saídas

- *clk_out*, *clk_aux*, *clk_botao*

Expressão Booleana

$$
\begin{aligned}
clk\_out &= q[5] \cdot q[4] \cdot q[3] \cdot q[2] \cdot q[1] \cdot q[0] \\
clk\_aux &= q[0] \\
clk\_botao &= q[1]
\end{aligned}
$$

A.2 SELETOR DE VELOCIDADE

clk_botao	botao	q[1]	q[0]	sinal	q[2] (chave_0)	q[3] (chave_1)	reset_contagem_botao
0	0	0	0	0	0	0	0
1	1	0	1	1	1	0	1
2	1	1	1	0	1	1	0
3	0	1	0	1	0	1	1
4	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Tabela 3 – Seletor de Velocidade

Entradas

- *botao*, *clk_botao*

Saídas

- *reset_contagem_botao*, *chave_*0, *chave_*1

Expressão Booleana

$$\begin{aligned} sinal &= q[0] \cdot \overline{q[1]} \\ reset\_contagem\_botao &= sinal \\ chave\_0 &= q[2] \\ chave\_1 &= q[3] \end{aligned}$$

A.3 CONTADOR ASSÍNCRONO

clk	reset	b3	b2	b1	b0
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1

Tabela 4 – Contador Assíncrono

Entradas

- clk, $reset$

Saídas

- $b3$, $b2$, $b1$, $b0$

Expressão Booleana

$$b0 = T0$$
$$b1 = T1$$
$$b2 = T2$$
$$b3 = T3$$

A.4 MULTIPLEXADOR DE VELOCIDADES

chave_0	chave_1	b3	b2	b1	b0	reset_2	reset_4	reset_8	reset_mux
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	1
0	0	0	1	0	0	0	1	0	1
0	0	1	0	0	0	0	0	1	1
0	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Tabela 5 – Modulo Multiplexador de Velocidades

Entradas

- $chave\_0$, $chave\_1$, $b3$, $b2$, $b1$, $b0$

Saídas

- $reset\_mux$

Expressão Booleana

$$reset\_2 = chave\_0 \cdot chave\_1 \cdot b1$$
$$reset\_4 = chave\_0 \cdot chave\_1 \cdot b2$$
$$reset\_8 = chave\_0 \cdot chave\_1 \cdot b3$$
$$reset\_mux = reset\_2 + reset\_4 + reset\_8$$

A.5 SELETOR DE CONTAGEM DAS AÇÕES

b3	**b2**	**b1**	**b0**	**seg_a**	**seg_b**	**seg_c**	**seg_d**	**seg_e**	**seg_f**	**seg_g**
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	1	0	1	1	1	0
1	1	0	0	1	0	0	1	1	1	1
1	1	0	1	1	0	1	1	1	0	1
1	1	1	0	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1	0	0	1

Tabela 6 – Modulo Seletor de Contagem das Ações

Entradas

- $b3$, $b2$, $b1$, $b0$

Saídas

- seg_a, seg_b, seg_c, seg_d, seg_e, seg_f, seg_g

Expressões Booleanas

$$
\begin{aligned}
seg\_a &= b2 \cdot \overline{b0} + \overline{b2} \cdot \overline{b1} \cdot b0 \\
seg\_b &= b2 \cdot b0 \\
seg\_c &= b1 \cdot \overline{b0} \\
seg\_d &= seg\_a \quad \text{(ou seja, mesma expressão de seg\_a)} \\
seg\_e &= b2 \cdot b0 \\
seg\_f &= \overline{b2} \cdot b0 + b1 \\
seg\_g &= \overline{b2} \cdot \overline{b1}
\end{aligned}
$$

A.6 SELETOR DE CONTAGEM DAS VELOCIDADES

chave_1	**chave_0**	**seg_a**	**seg_b**	**seg_c**	**seg_d**	**seg_e**	**seg_f**	**seg_g**
0	0	0	0	0	0	0	0	1
0	1	0	0	1	0	0	1	1
1	0	1	0	0	1	1	0	1
1	1	1	0	1	1	1	1	1

Tabela 7 – Modulo Seletor de Contagem das Velocidades

Entradas

- *botao*, *clk_botao*

Saídas

- *reset_contagem_botao*, *chave_0*, *chave_1*

Expressão Booleana

$$sinal = q[0] \cdot \overline{q[1]}$$
$$reset\_contagem\_botao = sinal$$
$$chave\_0 = q[2]$$
$$chave\_1 = q[3]$$

A.7 MULTIPLEXADOR DOS SEGMENTOS DO DISPLAY

Tseg_x	**Vseg_x**	**clk_aux**	**seg_x**
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Tabela 8 – Multiplexador dos Segmentos do Display

Entradas

- $Tseg\_x$, $Vseg\_x$, clk_aux

Saída

- seg_x

Expressão Booleana

$$seg\_x = \overline{clk\_aux} \cdot Tseg\_x + clk\_aux \cdot Vseg\_x$$

A.8 CONTROLE DO DISPLAY

Entradas

- $b3$, $b2$, $b1$, $b0$, clk_aux, $chave\_0$, $chave\_1$

Saídas

- seg_a, seg_b, seg_c, seg_d, seg_e, seg_f, seg_g

Expressões Booleanas

$$
\begin{aligned}
seg\_a &= \overline{clk\_aux} \cdot Tseg\_a + clk\_aux \cdot Vseg\_a \\
seg\_b &= \overline{clk\_aux} \cdot Tseg\_b + clk\_aux \cdot Vseg\_b \\
seg\_c &= \overline{clk\_aux} \cdot Tseg\_c + clk\_aux \cdot Vseg\_c \\
seg\_d &= \overline{clk\_aux} \cdot Tseg\_d + clk\_aux \cdot Vseg\_d \\
seg\_e &= \overline{clk\_aux} \cdot Tseg\_e + clk\_aux \cdot Vseg\_e \\
seg\_f &= \overline{clk\_aux} \cdot Tseg\_f + clk\_aux \cdot Vseg\_f \\
seg\_g &= \overline{clk\_aux} \cdot Tseg\_g + clk\_aux \cdot Vseg\_g
\end{aligned}
$$