UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

CONVERSOR ADC RAMPA DUPLA

Entrega 4

SEL 0628 – Sistemas Digitais

Prof. Dr. Maximiliam Luppe

Daniel Dias Silva Filho - 13677114

Daniel Umeda Kuhn - 13676541

Francyélio - 13676537

João Marcelo Ferreira Battaglini - 13835472

Manoel Thomaz - 13676392

Lucas Sales Duarte - 11734490

São Carlos – SP

01/07/2024

Daniel Dias Silva Filho - 13677114
Daniel Umeda Kuhn - 13676541
Francyélio - 13676537
João Marcelo Ferreira Battaglini - 13835472
Manoel Thomaz - 13676392
Lucas Sales Duarte - 11734490

4° Entrega

4° parte do trabalho de recuperação da disciplina SEL 0628
Professor: Maximiliam Luppe

São Carlos – SP

01/07/2024

Sumário

1. Introdução

A conversão precisa de sinais analógicos para digitais é essencial em diversas aplicações modernas, como instrumentos de medição e sistemas de controle industrial. Um método conhecido por sua precisão e imunidade a ruídos é o Conversor Analógico para Digital (ADC) de rampa dupla. Este relatório aborda o desenvolvimento de uma máquina de estados em HDL (Hardware Description Language) para controlar um ADC de rampa dupla, utilizando contadores decimais para visualização do valor analógico convertido.

O ADC de rampa dupla opera em duas fases principais: integração e desintegração. Na fase de integração, a tensão analógica é aplicada a um integrador por um tempo fixo, acumulando carga de forma linear. Na fase de desintegração, uma tensão de referência de polaridade oposta é aplicada ao integrador e um contador é iniciado. O contador para quando a saída do integrador atinge zero, e o valor contado é proporcional à tensão analógica inicial.

Este projeto foca na implementação digital do controlador do ADC, utilizando uma máquina de estados finitos para gerenciar as fases de integração e desintegração, além de controlar os registradores que armazenam o valor convertido e os conversores BCD para displays de 7 segmentos. A operação do controlador começa com a ativação da chave de medição e continua através dos estados de contagem, transferência de dados e reset do sistema.

A implementação em HDL permite uma descrição precisa do comportamento do sistema, possibilitando simulações detalhadas antes da síntese em hardware físico. Este relatório apresenta a teoria de funcionamento do ADC de rampa dupla, o código HDL do controlador, o circuito RTL correspondente e os resultados das simulações realizadas para validar o projeto.

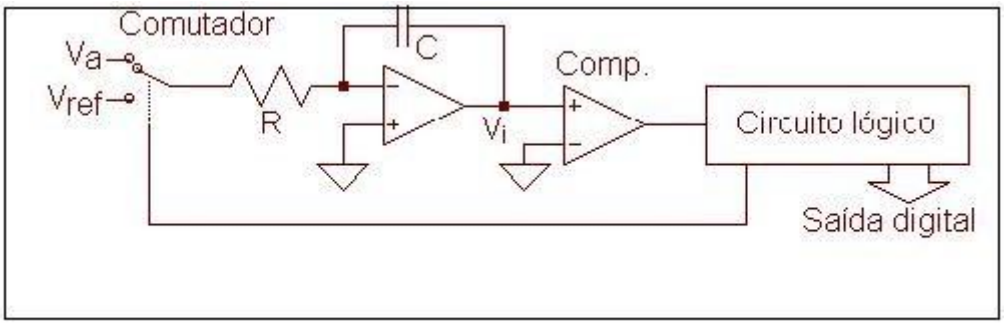
2. Desenvolvimento

Os conversores A/D de integração utilizam um integrador para converter o valor da tensão analógica em um intervalo de tempo, que pode ser medido com técnicas digitais simples. Existem diferentes tipos de circuitos que usam integradores, variando em complexidade e precisão. O conversor A/D de dupla rampa, por exemplo, é mais preciso do

que os mais simples, mas menos do que os de quádrupla rampa, que minimizam ainda mais os erros.

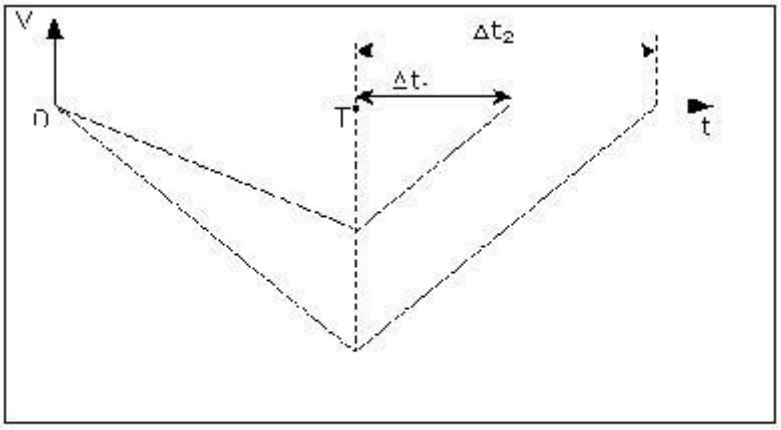
Na Figura 1, temos um esquema simplificado de um conversor A/D de dupla rampa.

Figura 1: *Esquema simplificado do conversor A/D Dupla Rampa*



Um switch eletrônico à esquerda alterna entre a tensão de entrada Va e uma tensão de referência Vref, sob o controle de um circuito lógico à direita, que inclui um contador de n bits. No início do processo de conversão, o contador está zerado e o capacitor C está descarregado, resultando em uma tensão de saída do integrador Vi igual a zero. O switch conecta a tensão analógica de entrada Va, considerada constante e positiva. A partir desse ponto, a tensão Vi começa a diminuir linearmente com um declive constante enquanto o contador incrementa os pulsos do relógio do sistema.

Figura 2: *Tensão do Conversor Rampa Dupla*



Quando o contador atinge seu valor máximo após um tempo $T = 2^n \cdot T_{REL}$ (onde T_{REL} é o período do relógio), o switch muda para a tensão de referência Vref, de sinal oposto à tensão de entrada. A tensão de saída do integrador passa a aumentar linearmente. Quando a tensão Vi atinge zero, o comparador muda seu estado, sinalizando ao circuito lógico para

parar o contador. O valor do contador é então uma medida do tempo de integração da segunda fase $\Delta T_1 = N \cdot T_{REL}$, onde Né o valor do contador.

Se a tensão de entrada fosse maior, a inclinação de Vi durante a primeira fase de integração seria maior e o tempo de desintegração seria diferente, representado por ΔT_2. Considerando Va constante, a corrente de carga do capacitor C é Va/R, e a carga acumulada ao final do tempo T é Va * T/R. Esta carga será removida na segunda fase, em que a corrente é -Vref/R e a carga é removida durante ΔT_1. A igualdade das cargas pode ser expressa como:

$$\frac{Va*T}{R} = \frac{\Delta T_1 * Vref}{R}$$

Assim, o valor de Va é dado por:

$$Va = Vref * \frac{\Delta T_1}{T}$$

O intervalo de tempo ΔT_1 é medido pelo contador, que tem uma resolução de n bits. O valor do contador N, quando a segunda fase de integração termina, é:

$$N = int[\frac{\Delta T_1}{Tref}]$$

Aqui, o operador Int extrai o maior inteiro da quantidade entre os parênteses. Quando o contador é parado, seu valor é o maior número de ciclos de relógio completos dentro do intervalo ΔT_1. Reescrevendo a equação para Va e considerando$T = 2^n \cdot T_{REL}$, temos:

$$N * T_{REL} \leq \Delta T_1 < (N + 1) * T_{REL}$$

Substituindo na equação paraVa:

$$\frac{Va}{Q} = N$$

Onde Q é o quantum definido como Vref/2^n. Assim, o resultado N da conversão é:

$$N = int[\frac{Va}{Q}]$$

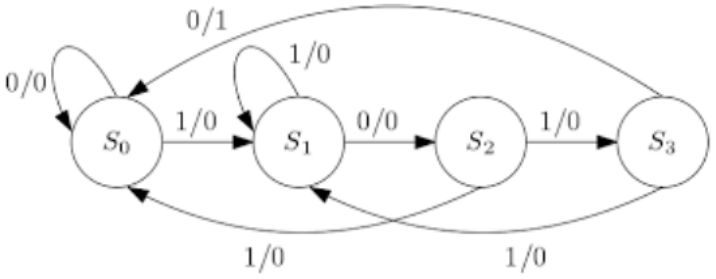
Isso indica que o valor N é a maior quantidade inteira de quanta que cabe na tensão Va. A característica teórica ideal teria a primeira transição em Va = Q/2 e as seguintes em múltiplos ímpares de Q/2, ou seja:

$$N = int[\frac{Va}{Q} + 0.5]$$

Para alcançar essa característica teórica, a conversão deve começar com um deslocamento de meio período de relógio em relação à contagem.

Por fim, concluímos que quanto mais bits o contador possuir, mais longo será o processo de conversão, mas a resolução será maior. A máquina de estados que gera os sinais pode ser definida de maneira arbitrária, seguindo os princípios das máquinas de estado finito. Uma máquina de estado finito, ou autômato finito, é um modelo matemático que descreve o comportamento de um sistema por meio da definição de um conjunto finito de estados e das transições entre esses estados em resposta a entradas específicas. Esse modelo é definido por um número S de estados, incluindo um estado inicial. Uma função de transição especifica as mudanças de estado, baseadas no estado atual e na entrada recebida pela máquina, conforme ilustrado na imagem abaixo.

Figura 3: *Máquina de Estados*



3. Códigos e RTL

```
module ADC_Controller (
    input wire clk,      // Sinal de clock
    input wire ch_vm,    // Chave para iniciar a conversão
    input wire ch_ref,   // Chave para iniciar a contagem do tempo "tm"
    input wire ch_zr,    // Chave para zerar a saída do integrador
    input wire Vint_z,   // Sinal indicando que a saída do integrador atingiu zero
    input wire enb_3,    // Sinal indicando o final da contagem "tx"
    input wire [9:0] count, // Contagem do contador de 000 a 999
    output reg [3:0] seg,   // Saída para os displays de 7 segmentos
    output reg dp           // Ponto decimal do display de 7 segmentos
);

    // Definição dos estados usando `parameter`
    parameter [2:0]
        STATE_IDLE = 3'b000,       // Estado inicial
        STATE_COUNT_TX = 3'b001,   // Estado de contagem "tx"
        STATE_COUNT_TM = 3'b010,   // Estado de contagem "tm"
        STATE_UPDATE = 3'b011,     // Estado de atualização dos registradores
        STATE_ZERO = 3'b100;       // Estado de zerar a saída do integrador

    // Registrador para armazenar o estado atual e o próximo estado
    reg [2:0] state, next_state;

    // Sinais de controle
    reg start_conversion;  // Sinal de controle para iniciar a conversão
    reg start_count_tm;    // Sinal de controle para iniciar a contagem "tm"
    reg transfer_data;     // Sinal de controle para transferir dados para os registradores
    reg zero_output;       // Sinal de controle para zerar a saída do integrador

    // Máquina de estados
    always @(posedge clk or posedge ch_zr) begin
        if (ch_zr) begin
            state <= STATE_IDLE; // Reset para o estado inicial
        end else begin
            state <= next_state;
```

```
    end
end

// Lógica de transição de estados
always @(*) begin
    // Valores padrão para evitar latches
    next_state = state; // Permanece no estado atual por padrão

    case (state)
        STATE_IDLE: begin
            if (ch_vm)
                next_state = STATE_COUNT_TX;
        end
        STATE_COUNT_TX: begin
            if (enb_3)
                next_state = STATE_COUNT_TM;
        end
        STATE_COUNT_TM: begin
            if (ch_ref)
                next_state = STATE_UPDATE;
        end
        STATE_UPDATE: begin
            if (Vint_z)
                next_state = STATE_ZERO;
        end
        STATE_ZERO: begin
            if (ch_zr)
                next_state = STATE_IDLE;
        end
        default: next_state = STATE_IDLE;
    endcase
end

// Lógica de controle dos sinais
always @(*) begin
    // Valores padrão para evitar latches
    start_conversion = 1'b0;
    start_count_tm = 1'b0;
    transfer_data = 1'b0;
    zero_output = 1'b0;

    case (state)
        STATE_IDLE: begin
            // Nada a fazer, sinais mantêm seus valores padrão
        end
        STATE_COUNT_TX: begin
            start_conversion = 1'b1;
        end
```

```
            STATE_COUNT_TM: begin
                start_count_tm = 1'b1;
            end
            STATE_UPDATE: begin
                transfer_data = 1'b1;
            end
            STATE_ZERO: begin
                zero_output = 1'b1;
            end
            default: begin
                // Define todos os sinais para valores padrão para evitar latches
                start_conversion = 1'b0;
                start_count_tm = 1'b0;
                transfer_data = 1'b0;
                zero_output = 1'b0;
            end
        endcase
    end

    // Controle dos displays de 7 segmentos
    always @(posedge clk) begin
        if (transfer_data)
            seg <= count[9:6]; // Atualiza o display com os bits mais significativos
        else
            seg <= 4'b1111; // Desliga o display
        dp <= 1'b0; // Configura o ponto decimal conforme necessário
    end

endmodule
```

Figura 4: *RTL gerado no Quartus*

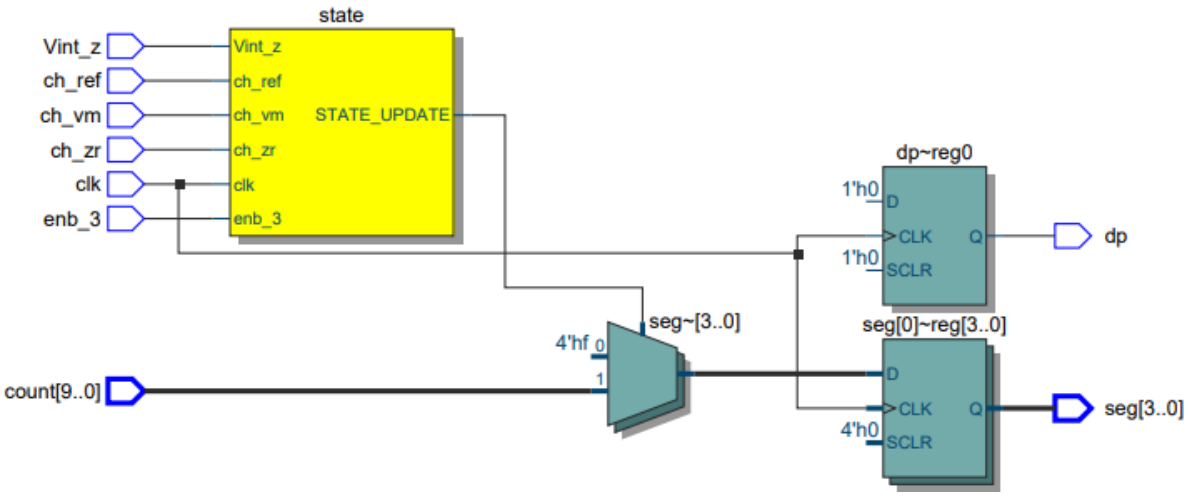


Figura 5: *Post-mapping gerado no Quartus*

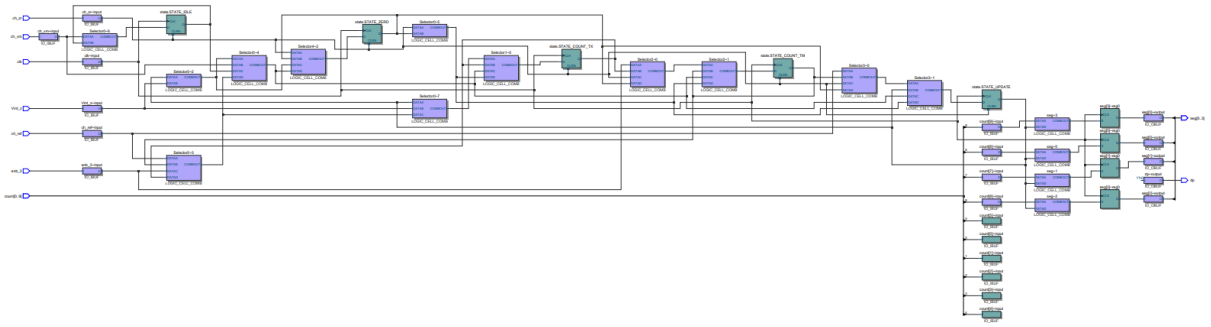
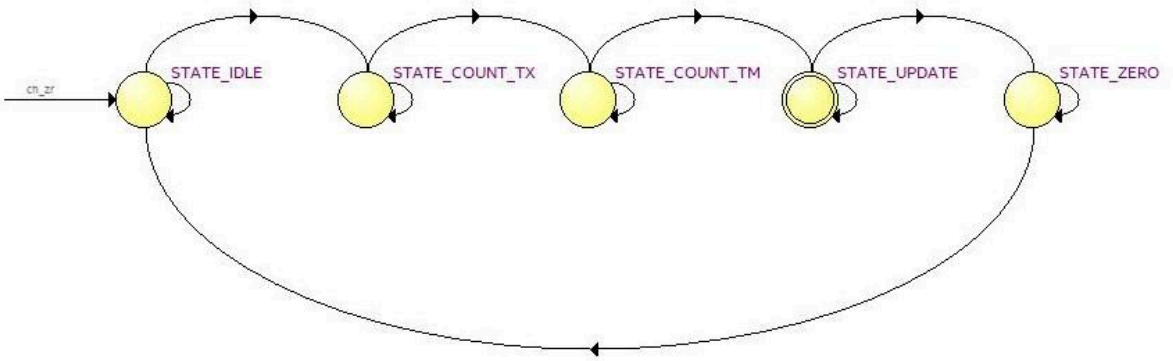


Figura 6: *Máquina de estados gerada no simulador*



4. Conclusão

Em conclusão, a implementação de um conversor analógico-digital (ADC) foi uma experiência valiosa no campo dos sistemas digitais. Utilizando Verilog, fomos capazes de descrever e simular o comportamento do conversor de maneira eficiente, assegurando a correta conversão de sinais analógicos para digitais. Esse processo nos proporcionou uma compreensão mais aprofundada das características e limitações dos dispositivos de conversão ADC, permitindo a exploração de diferentes estratégias de projeto para otimizar o desempenho do sistema. A prática com essa linguagem de descrição de hardware também aprimorou nossa capacidade de lidar com desafios práticos em projetos de sistemas digitais.

5. Referências

Conversor Analógico Digital - Rampa Dupla. 2010. Disponível em:<https://wiki.sj.ifsc.edu.br/index.php/Conversor_Anal%C3%B3gico_Digital_-_Rampa_Dupla> Acesso em: 01 jul. 2018

Thomas L. Floyd. (2016). "Sistemas Digitais: Fundamentos e Aplicações". Editora Bookman.

Donald D. Givone. (2016). "Introdução à Lógica Digital". Editora Cengage Learning.

Mano, M. M., & Kime, C. R. (2008). "Lógica Digital e Projeto de Computadores". Editora Pearson.