Aluno do Embarcatech_37 no IFMA

Nome: Manoel Felipe Costa Furtado

Matrícula: 20251RSE.MTC0086

Residência Profissional em FPGA – Turma DELTA - 2025.2

Atividade – Referente ao capítulo 01 da unidade 04

Tema do Capítulo – Linguagem VHDL

Prazo dia 21/09/2025 as 23:59

Objetivo: Desenvolver a capacidade de projetar, modelar, simular e implementar circuitos digitais por meio da linguagem VHDL (VHSIC Hardware Description Language), promovendo a compreensão dos conceitos de lógica digital e sua aplicação prática no desenvolvimento de sistemas embarcados e dispositivos programáveis, como FPGAs.

Enunciado: Para que um alimento seja classificado como light, seu valor calórico deve corresponder, no máximo, a 50% das calorias presentes no produto original. Considera-se, neste contexto, a adição de ingredientes opcionais utilizados para realçar o sabor e a coloração do alimento, os quais contribuem com diferentes percentuais calóricos em relação ao produto normal, conforme descrito a seguir:

- A contém 40%
- B contém 30%
- C contém 20%
- D contém 10%

A partir dessas definições, monte a tabela-verdade, o mapa de Karnaugh e implemente em linguagem VHDL a solução do problema. Projete um circuito para acender uma lâmpada cada vez que a combinação dos produtos misturados em um tanque ultrapassar 50% das calorias de um produto normal.

Instruções:

Ação	Função
01	Analise o enunciado e identifique as combinações de ingredientes que resultam em mais de 50% das calorias do produto original.
02	Construa a tabela-verdade representando todas as possíveis combinações dos ingredientes A (40%), B (30%), C (20%) e D (10%).
03	Monte o mapa de Karnaugh e simplifique a lógica booleana.
04	Implemente a expressão lógica em VHDL, projetando um circuito que acione uma lâmpada sempre que o limite de 50% for ultrapassado.
05	Utilize o terminal do Visual Studio Code (VSCode), em conjunto com o simulador GHDL, para compilar e simular o código VHDL e o testbench.

Solução

- GitHub: Segunda_Fase_FPGA/Unidade_04/Cap_01
- Link do Vídeo: https://youtu.be/q3qJEAzaRho
- Link do Código Completo: Unid_04_Cap_01.zip

1) Analise o enunciado e identifique as combinações Explicação

O objetivo é acender uma lâmpada (L) sempre que a soma dos percentuais calóricos dos ingredientes adicionados ultrapassar 50%.

- Dado que: A contém 40%; B contém 30%; C contém 20%; D contém 10%
- Condição para a Lâmpada Acender **(Saída L = 1):** Soma dos percentuais > 50%

Vamos identificar as combinações que satisfazem essa condição:

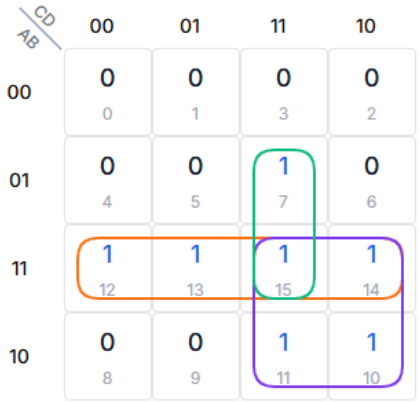
- A + B = 70% (> 50%) -> L=1
- A + C = 60% (> 50%) -> L=1
- A + D = 50% (não passa 50%) -> L=0
- B + C = 50% (não passa 50%) -> L=0
- B + D = 40% -> L=0
- C + D = 30% -> L=0
- A + B + C = 90% (> 50%) -> L=1
- A + B + D = 80% (> 50%) -> L=1
- A + C + D = 70% (> 50%) -> L=1
- B + C + D = 60% (> 50%) -> L=1
- A + B + C + D = 100% (> 50%) -> L=1

Qualquer outra combinação (um ingrediente sozinho ou combinações que somam 50% ou menos) resultará em L=0.

2) Tabela Verdade e Reduzindo o circuito

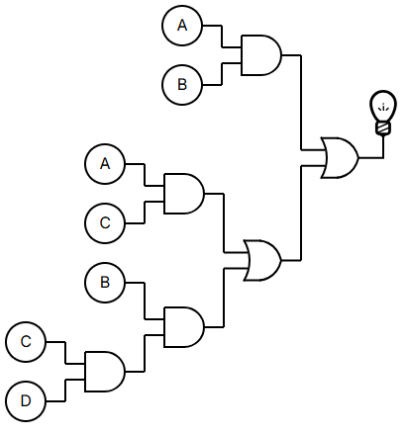
A (40%)	B (30%)	C (20%)	D (10%)	Soma (%)	L (Lâmpada)
0	0	0	0	0	0
0	0	0	1	10	0
0	0	1	0	20	0
0	0	1	1	30	0
0	1	0	0	30	0
0	1	0	1	40	0
0	1	1	0	50	0
0	1	1	1	60	1
1	0	0	0	40	0
1	0	0	1	50	0
1	0	1	0	60	1
1	0	1	1	70	1
1	1	0	0	70	1
1	1	0	1	80	1
1	1	1	0	90	1
1	1	1	1	100	1

3) Simplificando o Mapa de karnaugh



Expressão Booleana Simplificada

$AB + AC + BCD$



4) Implementação a expressão lógica em VHDL

```vhdl
-- Atividade: Unidade 03, Capítulo 01
-- Autor:     Manoel Felipe Costa Furtado
-- Data:      21/09/2025
-- Arquivo:   unid_03_cap_01.vhd
-- Versão:    1.0
-- Descrição: Implementa o circuito digital para o "Verificador de Calorias".
--            A saída é ativada quando a soma calórica dos ingredientes
--            ultrapassa 50% do produto original.

--------------------------------------------------------------------------------
-- BIBLIOTECAS
--------------------------------------------------------------------------------
LIBRARY ieee;
-- Usa o pacote std_logic_1164, que define os tipos STD_LOGIC e STD_LOGIC_VECTOR.
USE ieee.std_logic_1164.all;

--------------------------------------------------------------------------------
-- ENTIDADE (Interface Pública do Circuito)
--------------------------------------------------------------------------------
-- A entidade define as portas de entrada e saída do nosso circuito,
-- funcionando como uma "caixa preta" para o mundo exterior.
ENTITY unid_03_cap_01 IS
    PORT (
        -- Entradas que representam a presença de cada ingrediente.
        A, B, C, D : IN  STD_LOGIC;
        -- Saída que aciona a lâmpada (L).
        L          : OUT STD_LOGIC
    );
END ENTITY unid_03_cap_01;

--------------------------------------------------------------------------------
-- ARQUITETURA (Lógica Interna do Circuito)
--------------------------------------------------------------------------------
-- A arquitetura 'logica' descreve como as saídas são calculadas a partir
-- das entradas.
ARCHITECTURE logica OF unid_03_cap_01 IS
BEGIN
    -- Implementa a expressão booleana simplificada via Mapa de Karnaugh.
    -- A lâmpada (L) será '1' (acesa) se a combinação de ingredientes
    -- A(40%), B(30%), C(20%) e D(10%) ultrapassar 50% em valor calórico.
    L <= (A AND B) OR (A AND C) OR (B AND C AND D);

END ARCHITECTURE logica;
```

5) Código em VHDL para a Simulação no GHDL

```vhdl
-- Atividade: Unidade 03, Capítulo 01
-- Autor:     Manoel Felipe Costa Furtado
-- Data:      21/09/2025
-- Arquivo:   unid_03_cap_01_tb.vhd
-- Versão:    1.0
-- Descrição: Testbench para o módulo "Verificador de Calorias".
--            Este código simula todas as 16 combinações de entrada possíveis
--            para verificar se a saída do circuito corresponde ao esperado.

--------------------------------------------------------------------------------
-- BIBLIOTECAS
--------------------------------------------------------------------------------
LIBRARY ieee;
-- Pacote padrão para os tipos STD_LOGIC.
USE ieee.std_logic_1164.all;
-- Pacote para funções de conversão entre tipos numéricos e STD_LOGIC_VECTOR.
USE ieee.numeric_std.all;

--------------------------------------------------------------------------------
-- ENTIDADE (Testbench)
--------------------------------------------------------------------------------
-- A entidade de um testbench é tradicionalmente vazia, pois ele não possui
-- portas de entrada ou saída para o mundo exterior.
ENTITY unid_03_cap_01_tb IS
END ENTITY unid_03_cap_01_tb;

--------------------------------------------------------------------------------
-- ARQUITETURA (Simulação)
--------------------------------------------------------------------------------
ARCHITECTURE teste OF unid_03_cap_01_tb IS
    -- Declaração do componente que será testado (DUT - Design Under Test).
    -- A estrutura deve ser idêntica à entidade do arquivo de design.
    COMPONENT unid_03_cap_01 IS
        PORT (
            A, B, C, D : IN  STD_LOGIC;
            L          : OUT STD_LOGIC
        );
    END COMPONENT unid_03_cap_01;

    -- Sinais internos do testbench que serão conectados às portas do DUT.
    -- 's_' é um prefixo comum para indicar sinais de simulação.
    SIGNAL s_A, s_B, s_C, s_D : STD_LOGIC := '0';
    SIGNAL s_L                : STD_LOGIC;

BEGIN
    -- Instanciação do Componente sob Teste (UUT).
```

```
    -- Conecta os sinais locais do testbench às portas do componente.
    uut : unid_03_cap_01
        PORT MAP(
            A => s_A, B => s_B, C => s_C, D => s_D, L => s_L
        );

    -- Processo de estímulo: responsável por gerar os sinais de entrada ao longo do tempo.
    stimulus_process : PROCESS
        -- Variável local usada para armazenar a representação binária do contador do laço.
        VARIABLE i_vec : STD_LOGIC_VECTOR(3 DOWNTO 0);
    BEGIN
        -- Este laço itera de 0 a 15, cobrindo todas as combinações de 4 bits.
        FOR i IN 0 TO 15 LOOP
            -- 1. Converte o inteiro 'i' para um vetor de 4 bits.
            i_vec := std_logic_vector(to_unsigned(i, 4));

            -- 2. Atribui cada bit do vetor ao sinal de entrada correspondente.
            s_A <= i_vec(3); -- Bit mais significativo
            s_B <= i_vec(2);
            s_C <= i_vec(1);
            s_D <= i_vec(0); -- Bit menos significativo

            -- 3. Aguarda um intervalo de tempo para que o circuito processe as novas
            --    entradas e a saída se estabilize antes da próxima mudança.
            WAIT FOR 10 ns;
        END LOOP;

        -- Após o laço, o processo é suspenso indefinidamente para finalizar a simulação.
        WAIT;
    END PROCESS stimulus_process;

END ARCHITECTURE teste;
```

6) Compilação

--- Comandos GHDL para Simulação VHDL ---

Analisa (compila) o arquivo VHDL do seu circuito principal.

- ghdl -a unid_03_cap_01.vhd

Analisa (compila) o arquivo VHDL do seu testbench.

- ghdl -a unid_03_cap_01_tb.vhd

Elabora (constrói) o modelo de simulação a partir dos arquivos compilados.

- ghdl -e unid_03_cap_01_tb

Roda (executa) a simulação e cria o arquivo de forma de onda.

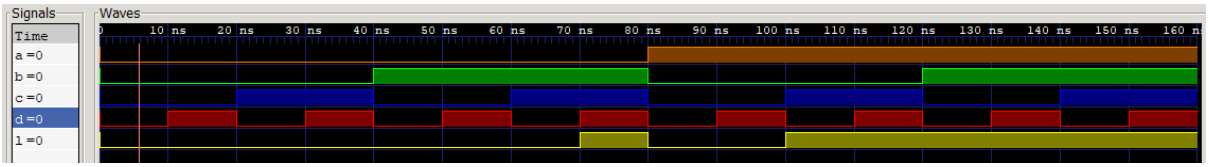
- ghdl -r unid_03_cap_01_tb --vcd=waveform.vcd

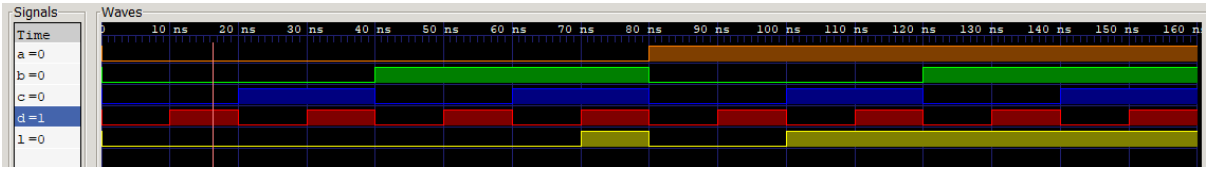
Visualiza o arquivo de forma de onda gerado.

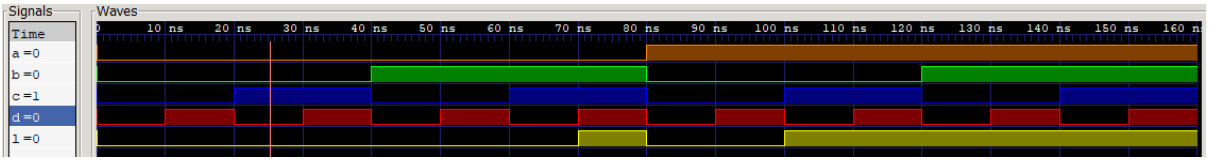
- gtkwave waveform.vcd

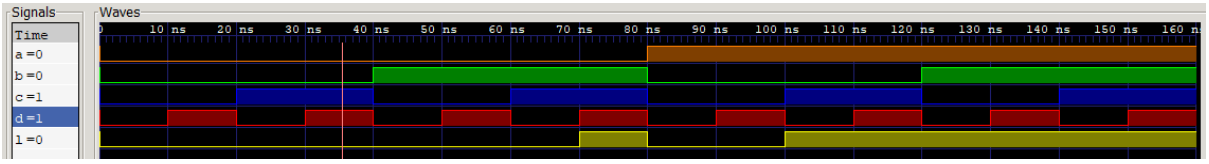
7) Simulação

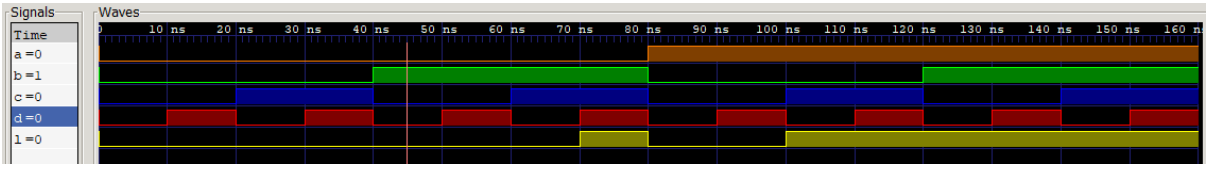
Fica claro pelas imagens abaixo que a forma de onda está de acordo com a tabela.

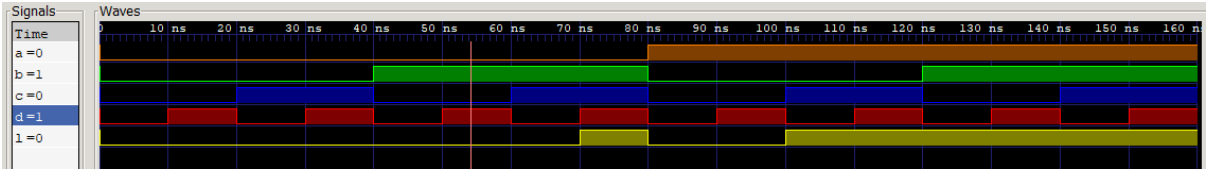


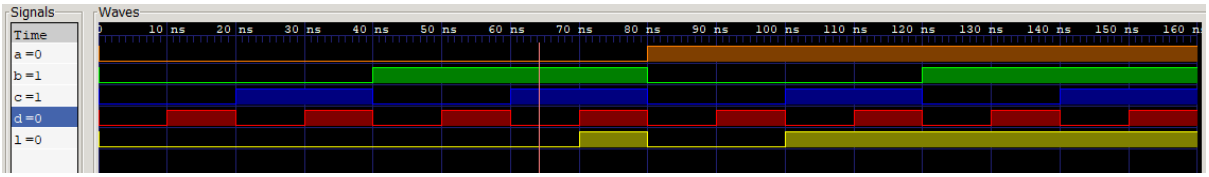


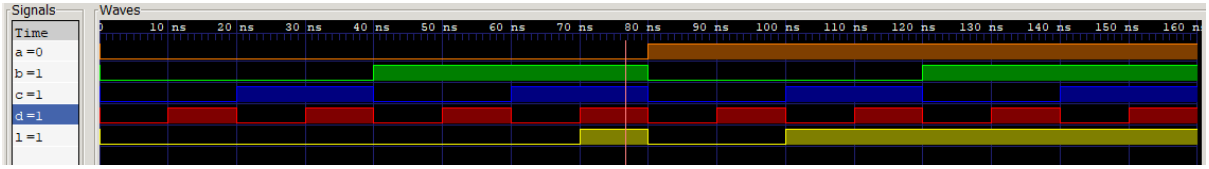


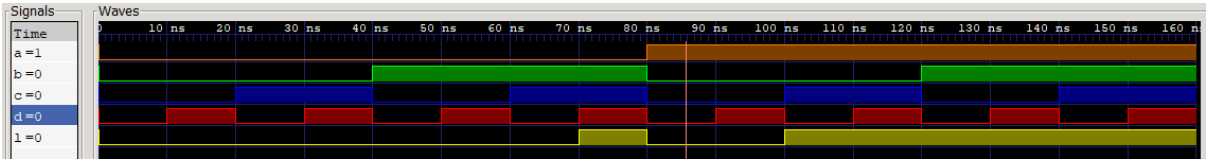


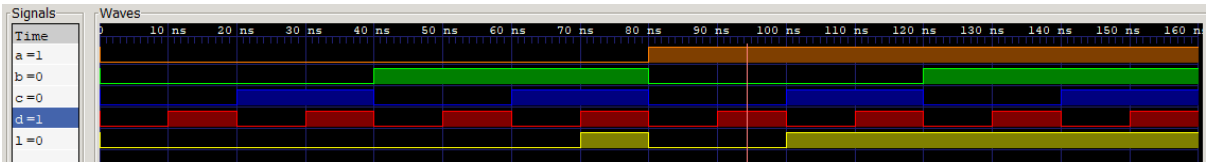


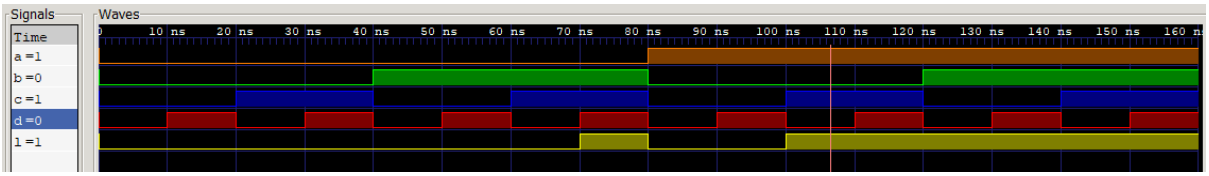


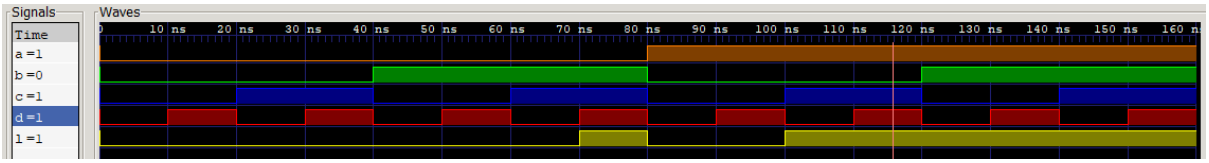


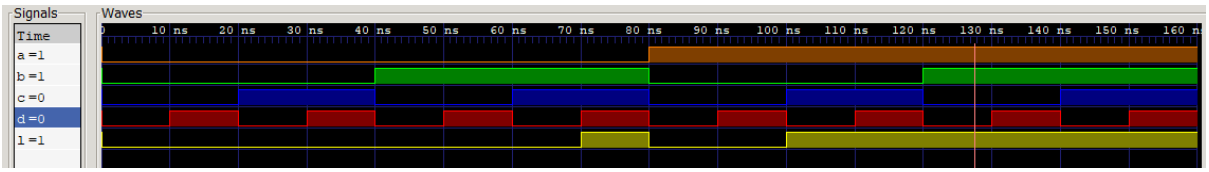


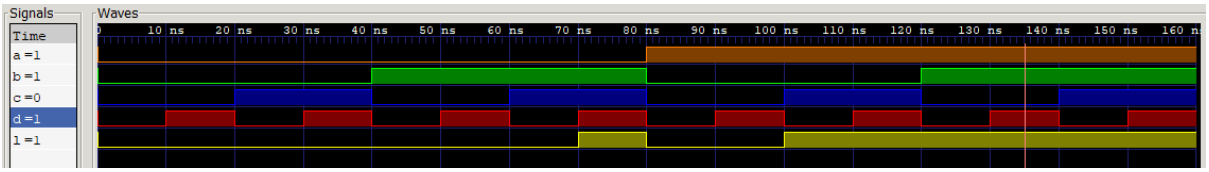


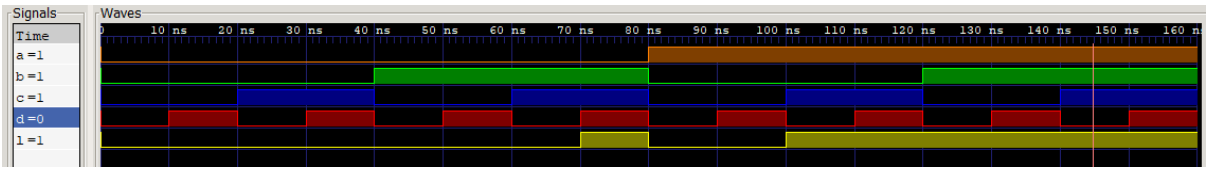

Signals
Waves
Time
a =1
b =0
c =0
d =1
l =0

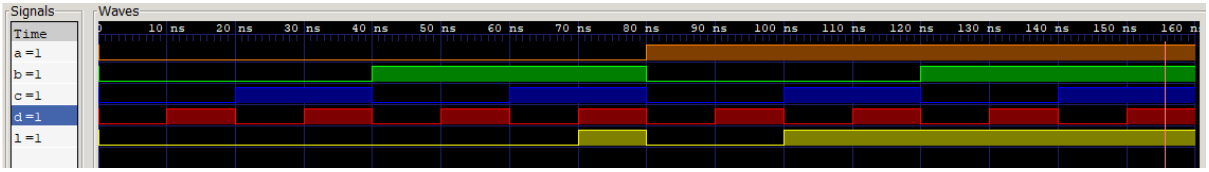

Signals
Waves
Time
a =1
b =0
c =1
d =0
l =1


Signals
Waves
Time
a =1
b =0
c =1
d =1
l =1


Signals
Waves
Time
a =1
b =1
c =0
d =0
l =1


Signals
Waves
Time
a =1
b =1
c =0
d =1
l =1


Signals
Waves
Time
a =1
b =1
c =1
d =0
l =1


Signals
Waves
Time
a =1
b =1
c =1
d =1
l =1