

Aluno do Embarcotech_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 02

Tema do Capítulo – Fundamentos de FPGA

Prazo dia 24/08/2025 as 23:59

Objetivo: Desenvolver e simular, no ambiente DigitaUS, um circuito digital completo utilizando portas lógicas combinacionais, com entradas e saídas manipuláveis, baseado em código Verilog e com visualização em tempo real do funcionamento lógico. O circuito deverá representar um sistema de alarme digital simplificado.

Enunciado:

Sistema de Alarme Digital com Simulação Visual Você deverá criar um circuito lógico que funcione como um sistema de alarme de segurança, com as seguintes condições:

Três sensores de entrada:

- Sensor de porta (entrada A)
- Sensor de janela (entrada B)
- Sensor de presença (entrada C)

Regras de ativação do alarme (saída Y):

- O alarme deve ser ativado ($Y = 1$) se qualquer dois sensores forem acionados simultaneamente.
- O alarme deve permanecer desativado ($Y = 0$) se menos de dois sensores forem acionados.

Instruções:

Descrição do Sistema:

- A lógica deve ser implementada em Verilog, e sintetizada no DigitaUS.
- As entradas A, B e C deverão ser representadas por botões (I/O).
- A saída Y deverá ser representada por um LED.
- O circuito deverá funcionar corretamente para todas as 8 combinações possíveis de entrada (000 até 111).
- Utilize as portas lógicas disponíveis para construir a lógica combinacional.

Sugestão de Lógica:

```
1 module alarme (  
2     input A,  
3     input B,  
4     input C,  
5     output Y  
6 );  
7     wire AB, AC, BC;  
8  
9     assign AB = A & B;  
10    assign AC = A & C;  
11    assign BC = B & C;  
12  
13    assign Y = AB | AC | BC;  
14  
15 endmodule
```

Entrada (Sensores):

- Porta do cofre (C = 0 - porta fechada; C = 1 - porta aberta)
- Relógio eletrônico (R = 0 - fora do expediente; R = 1 - horário de expediente)
- Interruptor na mesa do gerente (I = 0 - alarme desativado; I = 1 - alarme ativado)

Etapas de Implementação:

- 1) Acesse o DigitalJS: <https://digitaljs.tilk.eu>
- 2) Crie um novo projeto e insira o código Verilog sugerido (ou uma versão própria);
- 3) Clique em “Run” para compilar e gerar a simulação visual;
- 4) Conecte os botões (entradas) e LED (saída) ao circuito sintetizado;
- 5) Teste as combinações de entrada e verifique o comportamento da saída;
- 6) Exporte seu circuito em .json (opcional) e salve o código Verilog.

Requisitos técnicos:

- Código Verilog funcional com estrutura clara e nomeação adequada de sinais.
- Circuito sintetizado corretamente e funcionando conforme lógica solicitada.
- Uso de pelo menos duas portas AND e uma OR no projeto.
- Entradas e saída conectadas corretamente via interface do DigitalJS.
- Simulação testada para todos os casos de entrada (de 000 até 111).

Solução

- GitHub: https://github.com/ManoelFelipe/Embarcotech_37/tree/main/Segunda_Fase_FPGA/Unidade_02/Cap_01
- Link do Vídeo: <https://youtu.be/dFA1vKq4fds>
- Link do Código Completo: [Atividade_Uni_02_Cap_01](#)

1) Expressão Booleana

Essa expressão Booleana foi relativamente fácil de se obter observando as regras de ativação do alarme (saída Y):

$$Y = (A \cdot B) + (A \cdot C) + (B \cdot C)$$

Contudo, construindo a tabela verdade podemos verificar. E depois verificaremos que ela já está na sua forma reduzida.

Isso atende aos requisitos (“usar pelo menos duas AND e uma OR”)

2) Tabela Verdade

Índices	A (Porta)	B (Janela)	C (Presença)	Y (Alarme)	Condição
0	0	0	0	0	Nenhum sensor ativo
1	0	0	1	0	Apenas um sensor ativo
2	0	1	0	0	Apenas um sensor ativo
3	0	1	1	1	Dois sensores ativos (B e C)
4	1	0	0	0	Apenas um sensor ativo
5	1	0	1	1	Dois sensores ativos (A e C)
6	1	1	0	1	Dois sensores ativos (A e B)
7	1	1	1	1	Três sensores ativos

3) Reduzindo o circuito

- Em Laranja: $A \cdot B$
- Em Roxo: $A \cdot C$
- Em Verde: $B \cdot C$

Logo, o circuito não tem como reduzir mais do que a interpretação natural: $Y = (A \cdot B) + (A \cdot C) + (B \cdot C)$

		C	
		0	1
AB	00	0 0	0 1
	01	0 2	1 3
11	10	1 6	1 7
	11	0 4	1 5

4) Código em Verilog

```
/*
 * @file Atividade_Uni_02_Cap_01.sv
 * @version 1.0
 * @date 17/08/2025
 * @author Manoel Felipe Costa Furtado
 * @copyright 2025 Manoel Furtado (MIT License) (veja LICENSE.md)
 * @brief Psistema de alarme simplificado
 *
 *
 * @details
 * Módulo: alarme
 * Descrição: Implementa um sistema de alarme simplificado que é ativado
 * se pelo menos dois de três sensores estiverem ativos.
 * Entradas: A, B, C (sinais dos sensores)
 * Saída: Y (sinal de ativação do alarme)
 */

module alarme (
    input wire A, // Entrada do sensor de porta
    input wire B, // Entrada do sensor de janela
    input wire C, // Entrada do sensor de presença
    output wire Y // Saída para o LED do alarme
);

    // Fios (wires) intermediários para armazenar os resultados das
    // verificações de pares de sensores.
    wire AB, AC, BC;

    // Lógica Combinacional usando atribuições contínuas (assign)

    // A primeira porta AND verifica se os sensores A e B estão ativos
    assign AB = A & B;

    // A segunda porta AND verifica se os sensores A e C estão ativos
    assign AC = A & C;

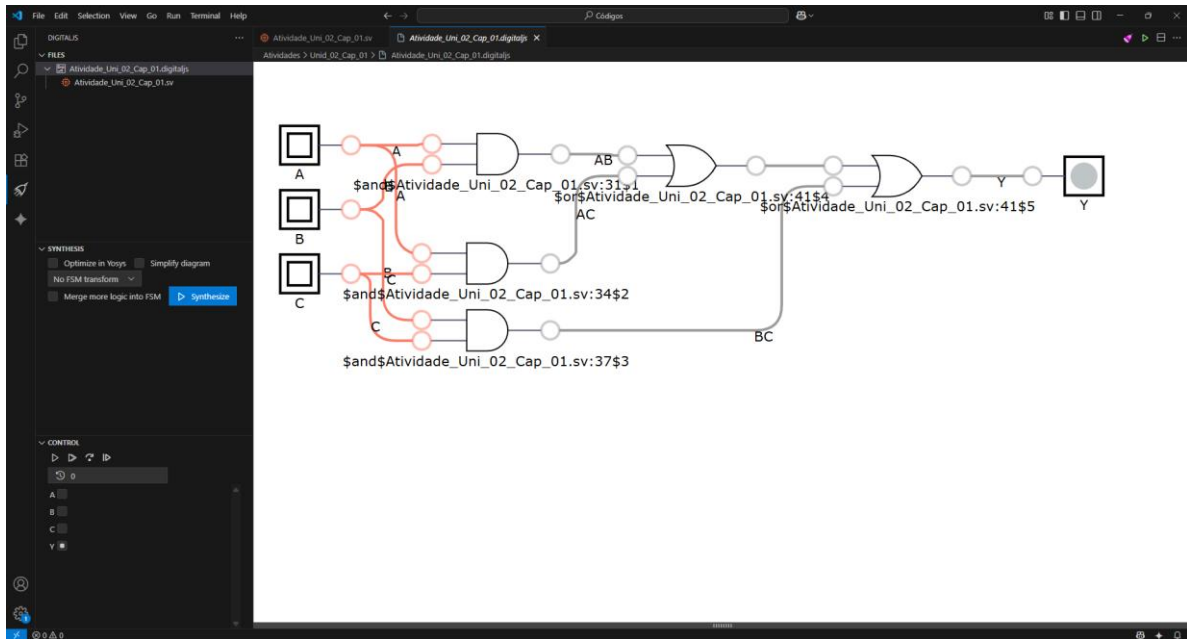
    // A terceira porta AND verifica se os sensores B e C estão ativos
    assign BC = B & C;

    // A porta OR final combina os resultados. O alarme (Y) será ativado
    // se QUALQUER uma das condições anteriores (AB, AC ou BC) for verdadeira.
    assign Y = AB | AC | BC;

endmodule
```

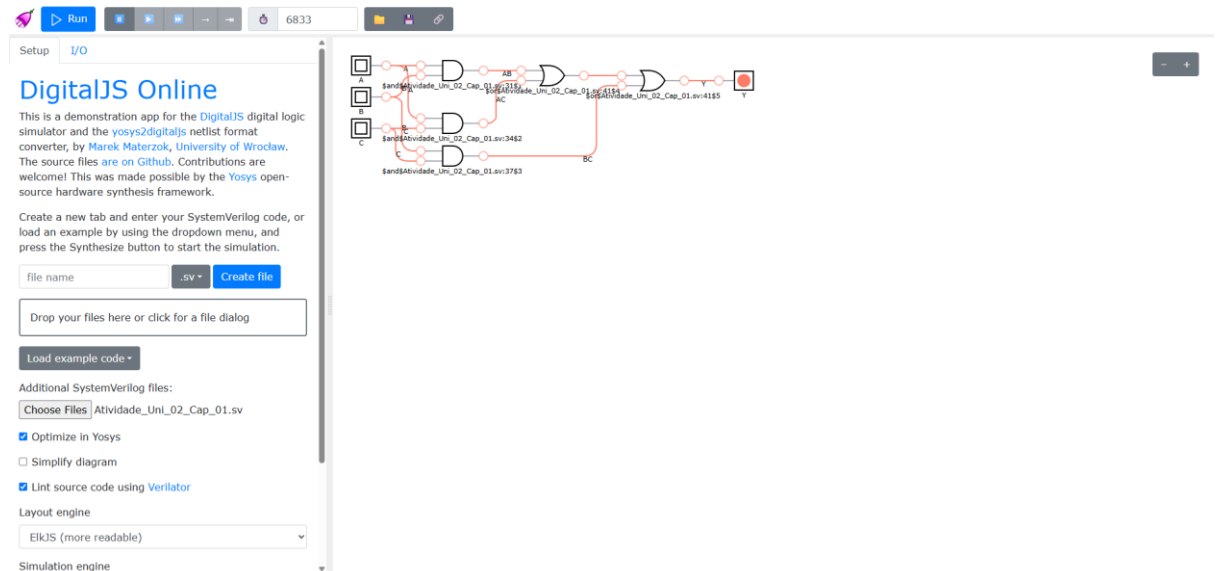
5) Desenho do Circuito

Gerado no VsCode

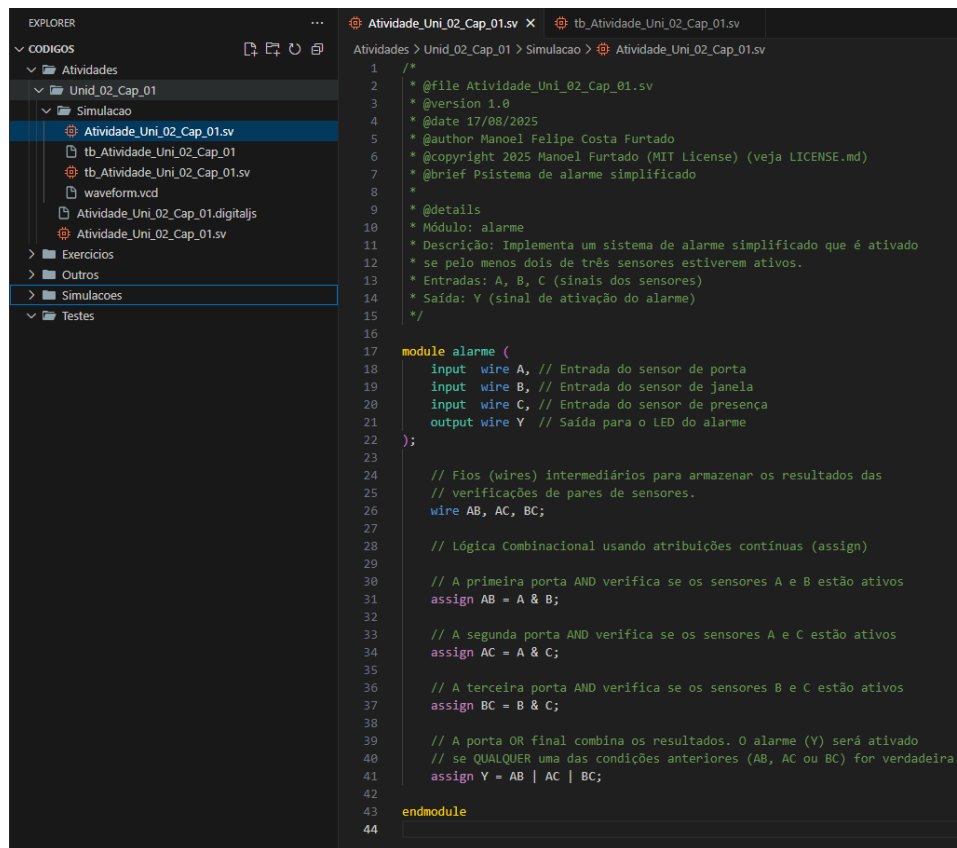


Gerado no <https://digitaljs.tilk.eu/>

Link: <https://digitaljs.tilk.eu/#783f8e0c2538fb51b0467aa86040135c27e1d1530f37326139150f12327e61ad>

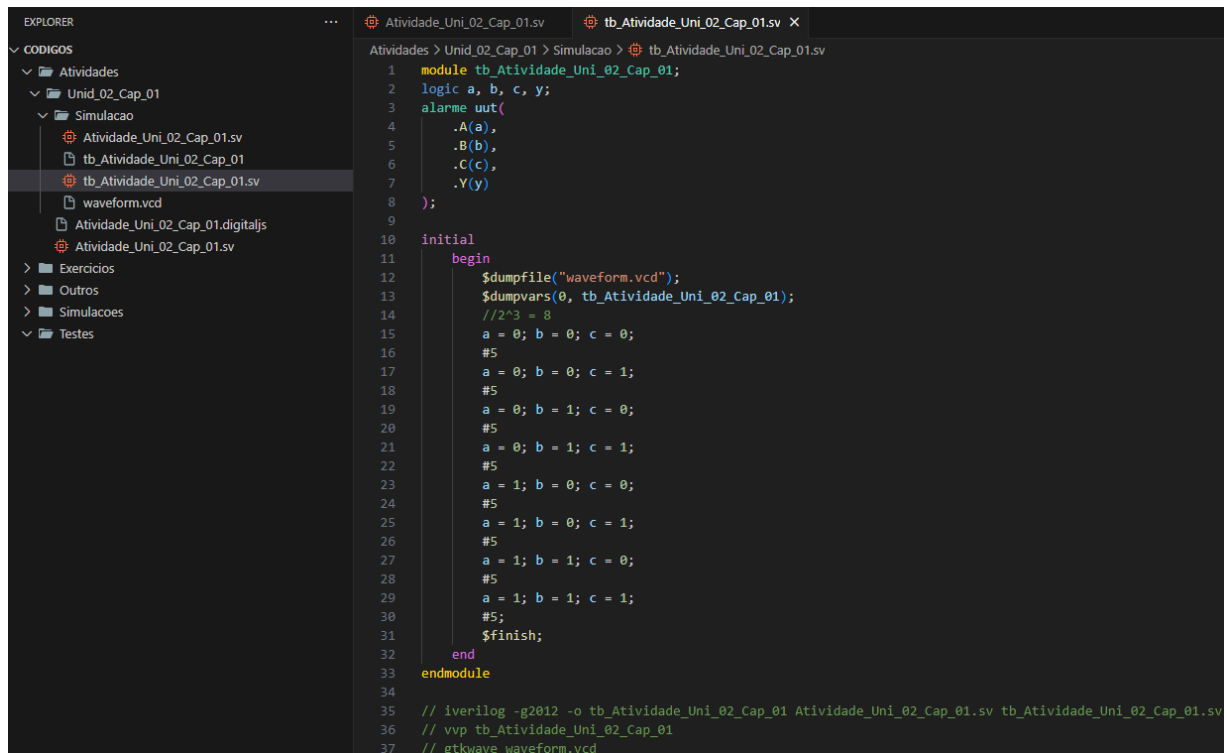


6) Simulação



The screenshot shows the Verilog code for the 'alarme' module. The Explorer panel on the left shows the project structure with 'Atividade_Uni_02_Cap_01' selected. The main editor displays the following code:

```
1  /*
2  * @file Atividade_Uni_02_Cap_01.sv
3  * @version 1.0
4  * @date 17/08/2025
5  * @author Manoel Felipe Costa Furtado
6  * @copyright 2025 Manoel Furtado (MIT license) (veja LICENSE.md)
7  * @brief Psistema de alarme simplificado
8  *
9  * @details
10 * Módulo: alarme
11 * Descrição: Implementa um sistema de alarme simplificado que é ativado
12 * se pelo menos dois de três sensores estiverem ativos.
13 * Entradas: A, B, C (sinais dos sensores)
14 * Saída: Y (sinal de ativação do alarme)
15 */
16
17 module alarme (
18     input wire A, // Entrada do sensor de porta
19     input wire B, // Entrada do sensor de janela
20     input wire C, // Entrada do sensor de presença
21     output wire Y // Saída para o LED do alarme
22 );
23
24 // Fios (wires) intermediários para armazenar os resultados das
25 // verificações de pares de sensores.
26 wire AB, AC, BC;
27
28 // Lógica Combinacional usando atribuições contínuas (assign)
29
30 // A primeira porta AND verifica se os sensores A e B estão ativos
31 assign AB = A & B;
32
33 // A segunda porta AND verifica se os sensores A e C estão ativos
34 assign AC = A & C;
35
36 // A terceira porta AND verifica se os sensores B e C estão ativos
37 assign BC = B & C;
38
39 // A porta OR final combina os resultados. O alarme (Y) será ativado
40 // se QUALQUER uma das condições anteriores (AB, AC ou BC) for verdadeira.
41 assign Y = AB | AC | BC;
42
43 endmodule
44
```



The screenshot shows the Verilog testbench code for the 'alarme' module. The Explorer panel on the left shows the project structure with 'tb_Atividade_Uni_02_Cap_01' selected. The main editor displays the following code:

```
1  module tb_Atividade_Uni_02_Cap_01;
2  logic a, b, c, y;
3  alarme uut(
4      .A(a),
5      .B(b),
6      .C(c),
7      .Y(y)
8  );
9
10 initial
11     begin
12         $dumpfile("waveform.vcd");
13         $dumpvars(0, tb_Atividade_Uni_02_Cap_01);
14         // 2^3 = 8
15         a = 0; b = 0; c = 0;
16         #5
17         a = 0; b = 0; c = 1;
18         #5
19         a = 0; b = 1; c = 0;
20         #5
21         a = 0; b = 1; c = 1;
22         #5
23         a = 1; b = 0; c = 0;
24         #5
25         a = 1; b = 0; c = 1;
26         #5
27         a = 1; b = 1; c = 0;
28         #5
29         a = 1; b = 1; c = 1;
30         #5;
31         $finish;
32     end
33 endmodule
34
35 // iverilog -g2012 -o tb_Atividade_Uni_02_Cap_01 Atividade_Uni_02_Cap_01.sv tb_Atividade_Uni_02_Cap_01.sv
36 // vvp tb_Atividade_Uni_02_Cap_01
37 // gtkwave waveform.vcd

```

```
module tb_Atividade_Uni_02_Cap_01;
logic a, b, c, y;
alarme uut(
    .A(a),
    .B(b),
    .C(c),
    .Y(y)
);

initial
    begin
        $dumpfile("waveform.vcd");
        $dumpvars(0, tb_Atividade_Uni_02_Cap_01);
        //2^3 = 8
        a = 0; b = 0; c = 0;
        #5
        a = 0; b = 0; c = 1;
        #5
        a = 0; b = 1; c = 0;
        #5
        a = 0; b = 1; c = 1;
        #5
        a = 1; b = 0; c = 0;
        #5
        a = 1; b = 0; c = 1;
        #5
        a = 1; b = 1; c = 0;
        #5
        a = 1; b = 1; c = 1;
        #5;
        $finish;
    end
endmodule

// iverilog -g2012 -o tb_Atividade_Uni_02_Cap_01 Atividade_Uni_02_Cap_01.sv tb_Atividade_Uni_02_Cap_01.sv
// vvp tb_Atividade_Uni_02_Cap_01
// gtkwave waveform.vcd
```

Formato de onda da saída Y.

Podemos verificar de acordo com a imagem abaixo a saída Y.

De acordo com a tabela verdade do problema apresentado

Quando:

- (a=0; b=1; c=1; y=1)
- (a=1; b=0; c=1; y=1)
- (a=1; b=1; c=0; y=1)
- (a=1; b=1; c=1; y=1)
- Nas outras condições a saída Y é 0(baixa).

