

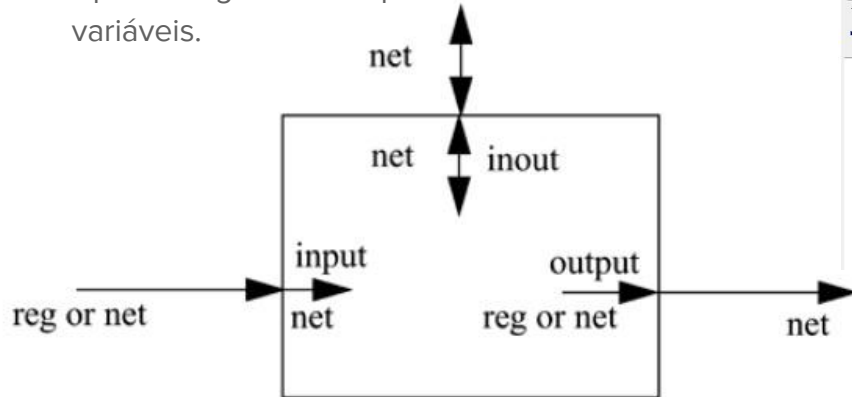
# Revisão de Verilog para o Projeto

---

Alef Gabryel

# Estrutura

- O nome do arquivo deve ser o mesmo nome do módulo.
- Deve se passar os outputs e inputs no módulo e especificá-los.
  - 'Reg' indica que é um registrador.
  - [0:X] ou [X:0] indica big-endian ou little-endian respectivamente. Pode mistura-los, mas é recomendado manter um padrão.
  - Apenas registradores podem ser usados em como variáveis.



```
1 module decodificador(number, display);
2
3 input [3:0] number;
4 output reg [0:6] display;
5
6 always @ (number) begin
7     case (number)
8         1: display <= 7'b1001111;
9         2: display <= 7'b0010010;
10        3: display <= 7'b0000110;
11        4: display <= 7'b1001100;
12        5: display <= 7'b0100100;
13        6: display <= 7'b0100000;
14        7: display <= 7'b0001101;
15        8: display <= 7'b0000000;
16        9: display <= 7'b0000100;
17        default: display <= 7'b0000001;
18    endcase
19 end
20
21 endmodule
22
```

# Decodificador

- 'Always @ (X or Y)' indica que irá realizar o que está dentro toda vez que X (ou Y se especificado) mudar.
- Não faz diferença utilizar '=' ou '<=' em um always, apenas que não é recomendado misturá-los.
  - Incentivamos a usar o '<=' pois é o que mais se aproxima do hardware.
- Este módulo recebe um número esperado ser de 0 a 9, e coloca a saída para refletir em um display de sete segmentos.

schematic in [Figure 4-10](#), the seven segments (common anode) are connected to pins on Cyclone IV E FPGA. Applying a low logic level to a segment will light it up and applying a high logic level turns it off.

Each segment in a display is identified by an index from 0 to 6, with the positions given in [Figure 4-10](#). [Table 4-4](#) shows the assignments of FPGA pins to the 7-segment displays.

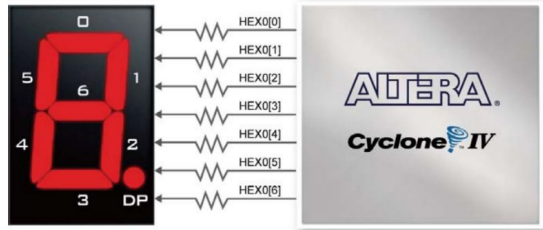
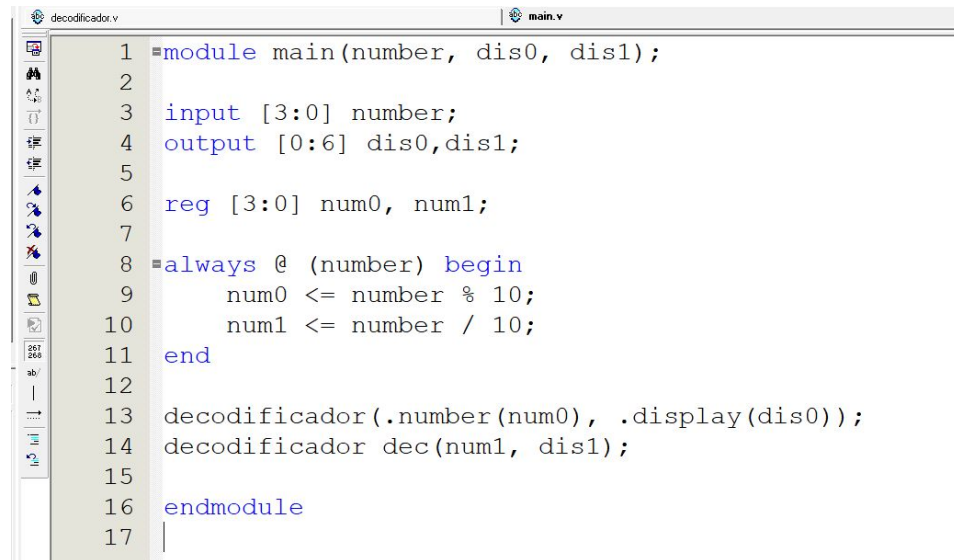


Figure 4-10 Connections between the 7-segment display HEX0 and Cyclone IV E FPGA

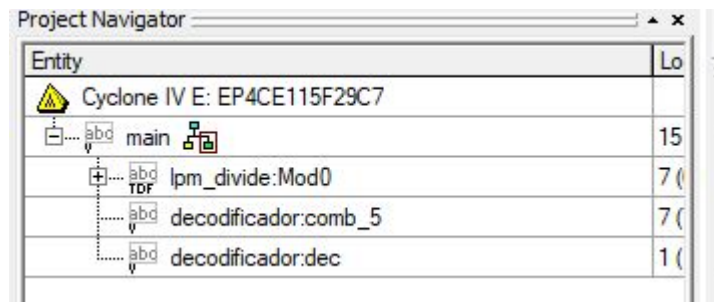
```
1 module decodificador(number, display);
2
3   input [3:0] number;
4   output reg [0:6] display;
5
6   always @ (number) begin
7       case (number)
8           1: display <= 7'b1001111;
9           2: display <= 7'b0010010;
10          3: display <= 7'b0000110;
11          4: display <= 7'b1001100;
12          5: display <= 7'b0100100;
13          6: display <= 7'b0100000;
14          7: display <= 7'b0001101;
15          8: display <= 7'b0000000;
16          9: display <= 7'b0000100;
17          default: display <= 7'b0000001;
18      endcase
19   end
20
21 endmodule
22
```

# Importando um módulo

- Para importar um módulo você não precisa criar um símbolo, é necessário apenas que o quartus possua o arquivo no mesmo projeto.
  - Você pode dar ou não um nome pro módulo.
  - Pode especificar ou não as portas em ordem. Portas não mapeadas serão ligadas ao ground.
- O quartus cria módulos automaticamente, e possui módulos para operações algébricas (divisão/multiplicação/mod) com registradores.



```
1 module main(number, dis0, dis1);
2
3   input [3:0] number;
4   output [0:6] dis0,dis1;
5
6   reg [3:0] num0, num1;
7
8   always @ (number) begin
9       num0 <= number % 10;
10      num1 <= number / 10;
11  end
12
13  decodificador(.number(num0), .display(dis0));
14  decodificador dec(num1, dis1);
15
16 endmodule
17
```



Project Navigator	
Entity	Lo
Cyclone IV E: EP4CE115F29C7	
main	15
lpm_divide:Mod0	7(
decodificador:comb_5	7(
decodificador:dec	1(

# Utilizando o Clock

- A placa possui um oscilador de 50 megahertz, isso é  $50 \times 10^6$  ciclos por segundo.
  - Posedge: Do nível lógico baixo para o alto.
  - Negedge: Do nível alto para o baixo.

The clock distribution on the DE2-115 board is shown in [Figure 4-11](#). The associated pin assignments for clock inputs to FPGA I/O pins are listed in [Table 4-5](#).

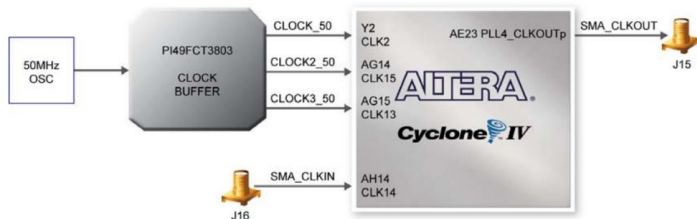


Figure 4-11 Block diagram of the clock distribution

Table 4-5 Pin Assignments for Clock Inputs

Signal Name	FPGA Pin No.	Description	I/O Standard
CLOCK_50	PIN_Y2	50 MHz clock input	3.3V
CLOCK2_50	PIN_AG14	50 MHz clock input	3.3V
CLOCK3_50	PIN_AG15	50 MHz clock input	Depending on JP6

```
Compilation Report | clock.v

1 module clock(clk, btn, dis0, dis1, led_btn);
2
3   input clk, btn;
4   reg pressed = 0;
5   reg [0:30] i = 0;
6   parameter segundo = 50000000;
7   reg [3:0] number = 0;
8   output led_btn; assign led_btn = btn;
9
10  always @ (posedge clk) begin
11      if (!btn) pressed <= 1;
12      if (btn && pressed) begin
13          pressed <= 0;
14          i <= 0;
15          number <= 0;
16      end
17      else if (i + 1 == segundo) begin
18          number <= number + 1;
19          i <= 0;
20      end
21      else i <= i + 1;
22  end
23  // O resto é o mesmo do main.v
24
25  output [0:6] dis0,dis1;
```

# Utilizando os botões

- Os botões são pull-up. Mantém o nível lógico 1, até serem pressionados, então possuem nível 0.
- Eles possuem debouncing integrados. Então não haverá ruídos ao serem pressionados.

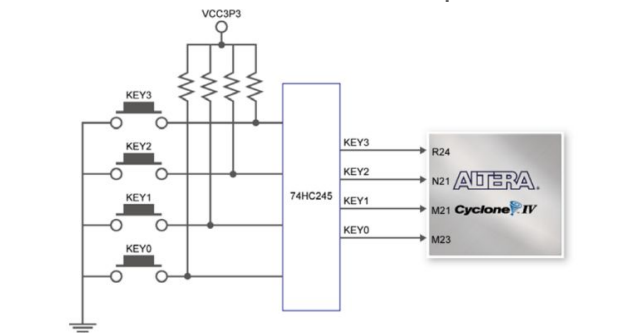
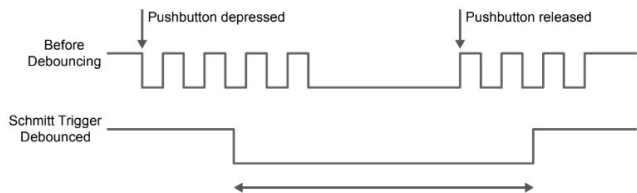


Figure 4-6 Connections between the push-button and Cyclone IV E FPGA

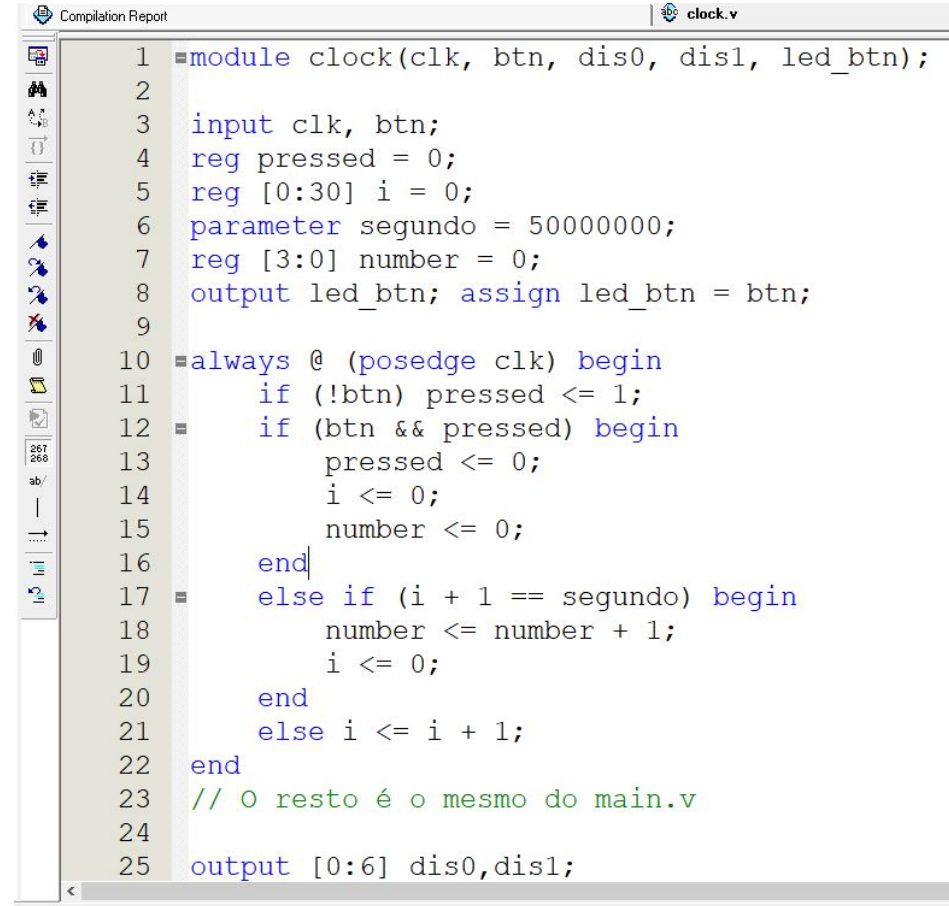


```
Compilation Report | clock.v

1 module clock(clk, btn, dis0, dis1, led_btn);
2
3 input clk, btn;
4 reg pressed = 0;
5 reg [0:30] i = 0;
6 parameter segundo = 50000000;
7 reg [3:0] number = 0;
8 output led_btn; assign led_btn = btn;
9
10 always @ (posedge clk) begin
11     if (!btn) pressed <= 1;
12     if (btn && pressed) begin
13         pressed <= 0;
14         i <= 0;
15         number <= 0;
16     end
17     else if (i + 1 == segundo) begin
18         number <= number + 1;
19         i <= 0;
20     end
21     else i <= i + 1;
22 end
23 // O resto é o mesmo do main.v
24
25 output [0:6] dis0,dis1;
```

# Contador

- Esse módulo adiciona um ao número a cada segundo até dar overflow.
- Ao pressionar o botão, no momento que o solta a contagem deve zerar. É utilizado um registrador para verificar se o botão foi pressionado.
- É usado o posedge para entrar apenas uma vez a cada ciclo.
- Parâmetros são como o define em C, utilizados apenas para definir valor e não devem (não podem convencionalmente) ser alterados no decorrer do programa.
- Só é possível alterar o registrador em um always.



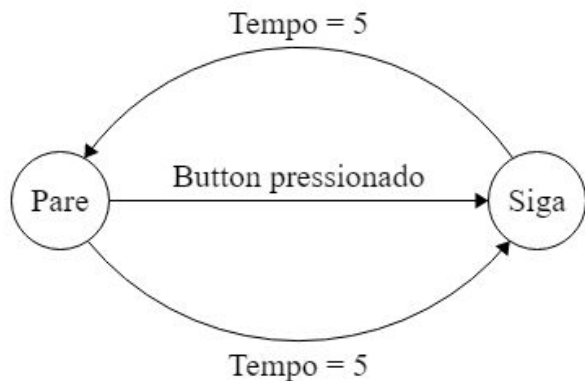
```
1 module clock(clk, btn, dis0, dis1, led_btn);
2
3   input clk, btn;
4   reg pressed = 0;
5   reg [0:30] i = 0;
6   parameter segundo = 50000000;
7   reg [3:0] number = 0;
8   output led_btn; assign led_btn = btn;
9
10  always @ (posedge clk) begin
11      if (!btn) pressed <= 1;
12      if (btn && pressed) begin
13          pressed <= 0;
14          i <= 0;
15          number <= 0;
16      end
17      else if (i + 1 == segundo) begin
18          number <= number + 1;
19          i <= 0;
20      end
21      else i <= i + 1;
22  end
23  // O resto é o mesmo do main.v
24
25  output [0:6] dis0,dis1;
```



# Máquina de estados finitas

## - Semáforo

- Uma máquina de estados finitas quer dizer que o circuito pode estar em uma ou outra configuração.
- Ela tem certos modos de entrar ou sair dela, que devem ser obedecidos.
- É bom costume atribuir todas as variáveis usadas em um always.



Compilation Report - Flow Summary

sinal.v

```
10 // Antes é similar ao clock
11 parameter pare = 0, siga = 1;
12 reg state = 0, next_state = 0;
13 output light;
14 assign light = state;
15
16 always @ (posedge clk) begin
17     if (state == next_state) begin
18         if (i + 1 == segundo) begin
19             number <= number + 1;
20             i <= 0;
21         end
22         else i <= i + 1;
23     end
24     else begin
25         state <= next_state;
26         i <= 0;
27         number <= 0;
28     end
29 end
30
31 always @ (negedge clk) begin
32     case (state)
33     pare: begin
34         if (!btn) pressed <= 1;
35         if (number == 5 || (btn && pressed)) begin
36             next_state <= siga;
37             pressed <= 0;end
38     end
39     siga: begin
40         pressed <= 0;
41         if (number == 5) begin
42             next_state <= pare;end
43         end
44     endcase
45 end
46 // O resto é igual ao main
47
```