UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

Módulo contador binário parametrizável

Entrega 2

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

2° Entrega

2° parte do trabalho de recuperação da disciplina SEL 0628
Professor: Maximiliam Luppe

São Carlos – SP

01/07/2024

1.Introdução

Os contadores binários parametrizáveis permitem ajustar a largura dos bits e as funcionalidades específicas, como habilitação e reset, conforme requerido pelas aplicações. Este tipo de contador é projetado para operar de maneira síncrona ou assíncrona, proporcionando precisão e consistência na contagem ou flexibilidade na reinicialização, respectivamente.

Os contadores binários síncronos trabalham em sincronia com um sinal de clock. Eles incluem entradas de habilitação e reset síncrono, permitindo controle detalhado sobre o início e a reinicialização da contagem. Quando o contador atinge o valor máximo especificado, ele é automaticamente reiniciado. Em contraste, os contadores binários assíncronos empregam um reset assíncrono, oferecendo maior flexibilidade na reinicialização, independente do sinal de clock, e voltam a zero imediatamente ao atingir o valor máximo.

Na segunda parte do projeto, foi implementado um módulo contador binário parametrizável de 'width' bits e saída 'q'. Este contador possui um parâmetro de entrada que limita a contagem máxima do contador e gera uma saída igual a ‘1’ quando a contagem atinge valor máximo. Também possui sinais de controle que inicializam e habilitam a contagem, além de permitir a entrada de clock.

2. Desenvolvimento

A seguir, constam as tabelas verdade e os mapas de karnaugh referentes a esse contador, assim como a construção destes em modo *assíncrono*, *síncrono* e utilizando *declarações comportamentais*.

Tabela verdade (truth table)

Com auxílio da tabela verdade, nos é permitido representar de maneira sistemática todas as combinações possíveis dos valores lógicos das entradas de um circuito, juntamente com os valores lógicos correspondentes das saídas. Através da tabela verdade, é possível identificar todas as situações lógicas de um circuito em todas as situações possíveis.

A seguir, apresentamos a tabela verdade de 16 bits dos nossos circuitos lógicos. Esta tabela abrange desde B0 até B3, e os valores vão de 0 a 9, que é o maior número que o contador de décadas deve alcançar. Quando o contador atinge o valor 9, ele deve ser reiniciado para 0.

Com essa análise, é possível haver a previsibilidade do comportamento do circuito através da Figura 1 a seguir.

N	A3	A2	A1	A0	B3	B2	B1	B0
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0
2	0	0	1	0	0	0	0	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	0	0	1
5	0	1	0	1	0	1	0	0
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	0	0	1
9	1	0	0	1	1	0	0	0
10	1	0	1	0	1	1	X	X
11	1	0	1	1	1	1	X	X
12	1	1	0	0	X	X	X	X
13	1	1	0	1	X	X	X	X
14	1	1	1	0	X	X	X	X
15	1	1	1	1	X	X	X	X

Tabela 1 - Tabela verdade do contador

Expressões booleanas

Usando a tabela acima, separa-se os segmentos e seus respectivos mapas de Karnaugh:

- Segmento B_0

$$B_0 = /A_0 \quad (1)$$

- Segmento B_1

$$B_1 = (A_1 /A_0) + (/A_3 /A_1 A_0) \quad (2)$$

- Segmento B_2

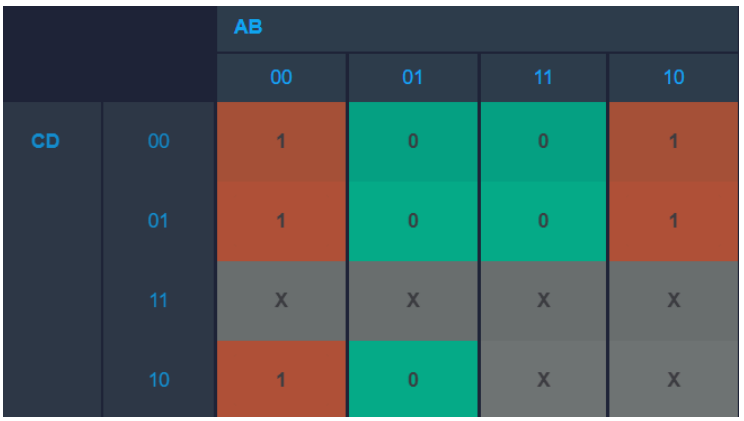
$$B_2 = (A_2 /A_1) + (A_2 /A_0) + (/A_2 /A_1 A_0) \quad (3)$$

- Segmento B_3

$$B_3 = (A_3 /A_0) + (A_2\ A_1\ A_0) \quad (4)$$

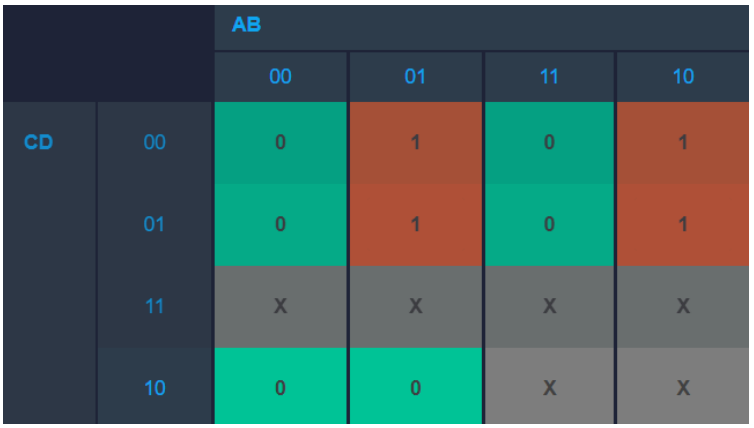
Mapa de Karnaugh

À seguir são ilustrados os mapas de Karnaugh referentes aos segmentos acima pelas suas respectivas equações. As ilustrações foram feitas no site do ITC Lab [2].



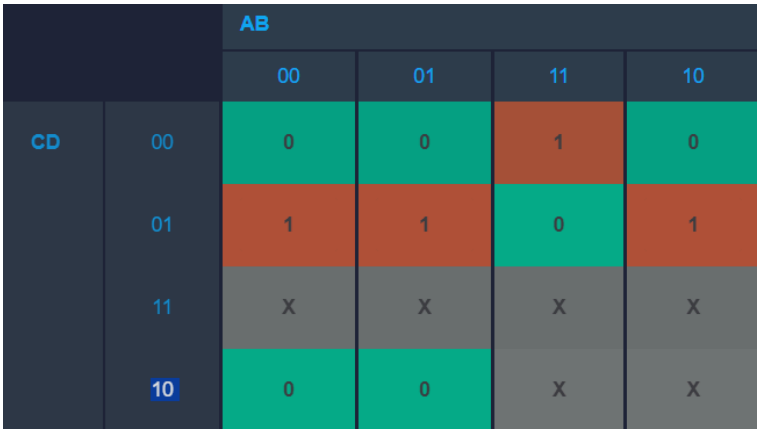
		AB			
		00	01	11	10
CD	00	1	0	0	1
	01	1	0	0	1
	11	X	X	X	X
	10	1	0	X	X

Figura 2 - Mapa do segmento B_0 [2]



		AB			
		00	01	11	10
CD	00	0	1	0	1
	01	0	1	0	1
	11	X	X	X	X
	10	0	0	X	X

Figura 3 - Mapa do segmento B_1 [2]



		AB			
		00	01	11	10
CD	00	0	0	1	0
	01	1	1	0	1
	11	X	X	X	X
	10	0	0	X	X

Figura 4 - Mapa do segmento B_2 [2]

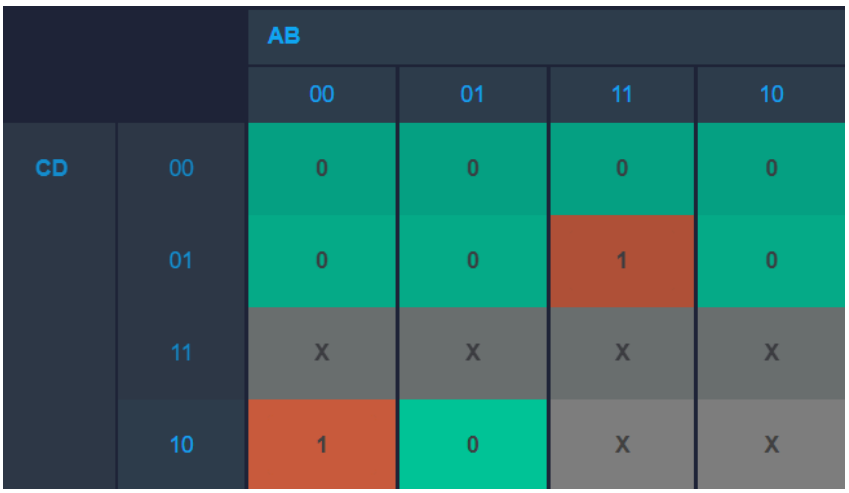


Figura 5 - Mapa do segmento B_3 [2]

Circuitos HDL

Com base nos dados anteriormente coletados sobre o comportamento do circuito (Tabela verdade e mapas de Karnaugh), construiu-se seguintes circuitos HDL:

- Assíncrono, utilizando o Flip-flop Tipo-D descrito anteriormente, com generate e Rede de Ligações;
- Síncrono, utilizando o Flip-flop Tipo-D descrito anteriormente, com generate e Rede de Ligações;
- Com incremento, utilizando Declaração Procedural ou Comportamental (always if-else);

Assíncrono

módulo d_ff é uma implementação de um flip-flop tipo D com entradas de reset e clock habilitado de forma síncrona. Ele possui quatro portas principais: d para a entrada de dados, clk para o sinal de clock, reset para o reset síncrono (ativo em nível alto) e ce para a habilitação do clock (clock enable). A saída q é o resultado do flip-flop tipo D, que armazena o valor da entrada d na borda de subida do clock quando a habilitação ce está ativa. Se o reset (reset em nível alto) estiver ativado, a saída q é definida como 0. Esse módulo é fundamental na construção de circuitos sequenciais em lógica digital.

O módulo contador_binario_async representa um contador binário assíncrono que também utiliza o módulo d_ff, mas com entradas de set assíncrono. Similar ao contador síncrono, ele usa um "generate loop" para instanciar flip-flops D individuais para cada bit do contador. As três entradas principais deste módulo são: enb para a habilitação, rst_s para o reset assíncrono (ativo em nível alto) e ck para o sinal de clock. A saída q indica o valor atual do contador binário, enquanto cnt_max é um sinal que

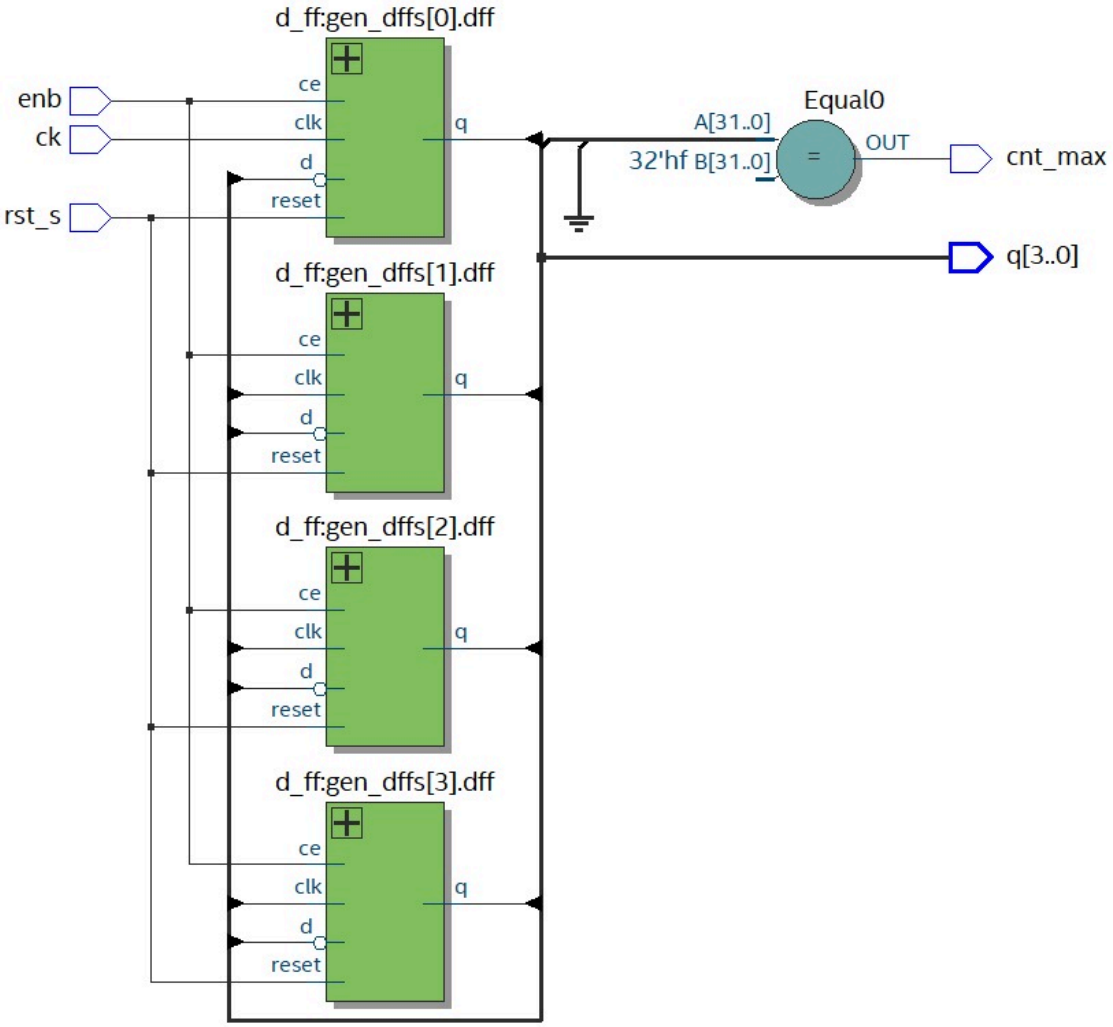
indica quando o contador atinge o valor máximo (max_value). O contador assíncrono incrementa seu valor sempre que a habilitação enb está ativa e oferece maior flexibilidade em relação ao reset, que ocorre de forma assíncrona em relação ao sinal de clock. Quando o reset assíncrono (rst_s) é ativado, o contador é redefinido para 0. Se o contador atinge o valor máximo (max_value), ele retorna a 0 e o sinal cnt_max é ativado.

Este contador binário assíncrono, assim como o flip-flop D, são componentes cruciais na criação de sistemas digitais, permitindo a implementação de contagem e armazenamento de dados em diversos tipos de circuitos eletrônicos.

```
module d_ff (
    input wire d, clk, reset, ce,
    output reg q
);
    always @(posedge clk or posedge reset) begin
        if (reset)
            q <= 1'b0;
        else if (ce)
            q <= d;
    end
endmodule

module contador_binario_async #(parameter WIDTH = 4, parameter MAX_VALUE = 2**WIDTH-1) (
    input wire enb, rst_s, ck,
    output wire [WIDTH-1:0] q,
    output wire cnt_max
);
    wire [WIDTH-1:0] d;
    wire [WIDTH-1:0] ff_q;

    assign cnt_max = (ff_q == MAX_VALUE);

    genvar i;
    generate
        for (i = 0; i < WIDTH; i = i + 1) begin: gen_dffs
            if (i == 0) begin
                d_ff dff (
                    .d(~ff_q[i]),
                    .clk(ck),
                    .reset(rst_s),
                    .ce(enb),
                    .q(ff_q[i])
                );
            end else begin
                d_ff dff (
                    .d(~ff_q[i]),
                    .clk(ff_q[i-1]),
                    .reset(rst_s),
                    .ce(enb),
                    .q(ff_q[i])
                );
            end
```

```
        end
    endgenerate

    assign q = ff_q;
endmodule

module tb_contador_binario_async;
    parameter WIDTH = 4;
    parameter MAX_VALUE = 2**WIDTH-1;
    reg clk, enb, rst_s;
    wire [WIDTH-1:0] q;
    wire cnt_max;

    contador_binario_async #(WIDTH, MAX_VALUE) uut (
        .enb(enb),
        .rst_s(rst_s),
        .ck(clk),
        .q(q),
        .cnt_max(cnt_max)
    );

    initial begin
        clk = 0; enb = 0; rst_s = 1;
        #10 rst_s = 0;
        #10 enb = 1;
        #100 $stop;
    end

    always #5 clk = ~clk;

    initial begin
        $monitor("Time = %0t | q = %0b | cnt_max = %b", $time, q, cnt_max);
    end
endmodule
```

Figura 1 - RTL ilustrativo referente ao circuito Assíncrono



Síncrono

Este código implementa um contador binário síncrono parametrizável utilizando Flip-Flops Tipo-D em Verilog.

Módulo d_ff

O módulo d_ff define um Flip-Flop Tipo-D com reset síncrono (reset) e habilitação de clock (ce). Na borda de subida do clock (clk), se o reset estiver ativo, a saída (q) é zerada. Caso contrário, se o clock enable estiver ativo, a saída recebe o valor da entrada (d).

Módulo contador_binario

O módulo contador_binario é um contador binário síncrono parametrizável, com largura definida por WIDTH e valor máximo por MAX_VALUE. Possui três entradas principais:

- enb: Habilitação do contador.
- rst_s: Reset síncrono (ativo em nível alto).
- ck: Sinal de clock.

A saída q representa o valor atual do contador, enquanto cnt_max indica se o contador atingiu o valor máximo. O contador utiliza um loop generate para criar Flip-Flops Tipo-D individuais para cada bit do contador. Cada Flip-Flop recebe seu valor a partir do bit anterior e incrementa o contador quando habilitado. Quando o contador atinge MAX_VALUE, ele retorna a zero e cnt_max é ativado.

Testbench tb_contador_binario

O testbench tb_contador_binario verifica o funcionamento do contador. Ele instancia o contador, aplica sinais de clock, habilitação e reset, e monitora as saídas q e cnt_max.

Fluxo do Código

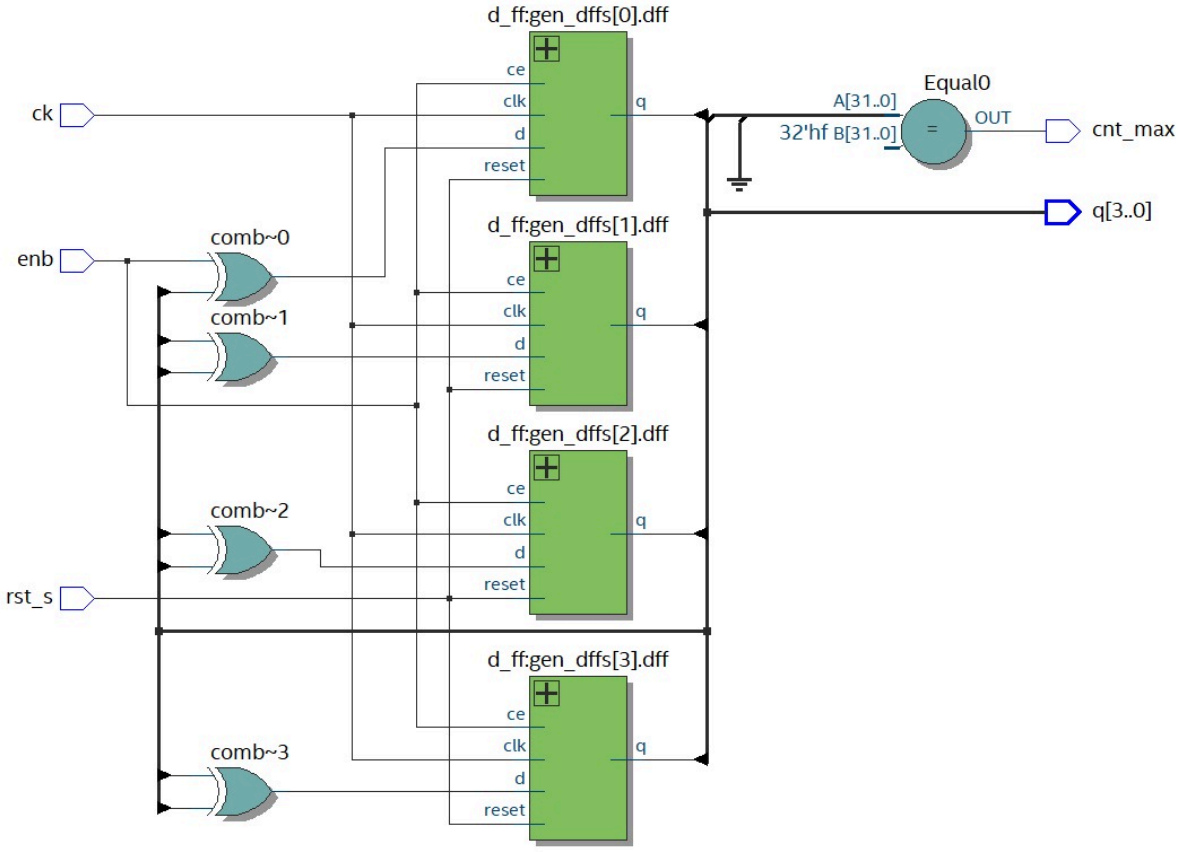
1. O Flip-Flop Tipo-D armazena o valor da entrada d na borda de subida do clock quando habilitado.
2. O contador incrementa em cada ciclo de clock enquanto enb está ativo.
3. Quando rst_s é ativado, o contador é zerado.
4. Quando o valor do contador atinge MAX_VALUE, o sinal cnt_max é ativado.

Este código é útil para criar contadores síncronos em projetos de sistemas digitais, oferecendo controle preciso sobre a contagem e reset.

```
module d_ff (
    input wire d, clk, reset, ce,
    output reg q
);
    always @(posedge clk or posedge reset) begin
        if (reset)
            q <= 1'b0;
        else if (ce)
            q <= d;
    end
endmodule

module contador_binario_sync #(parameter WIDTH = 4, parameter MAX_VALUE = 2**WIDTH-1) (
```

```
    input wire enb, rst_s, ck,
    output wire [WIDTH-1:0] q,
    output wire cnt_max
);
    wire [WIDTH-1:0] ff_q;

    assign cnt_max = (ff_q == MAX_VALUE);

    genvar i;
    generate
        for (i = 0; i < WIDTH; i = i + 1) begin: gen_dffs
            d_ff dff (
                .d(ff_q[i] ^ (i == 0 ? enb : ff_q[i-1])),
                .clk(ck),
                .reset(rst_s),
                .ce(enb),
                .q(ff_q[i])
            );
        end
    endgenerate

    assign q = ff_q;
endmodule

module tb_contador_binario_sync;
    parameter WIDTH = 4;
    parameter MAX_VALUE = 2**WIDTH-1;
    reg clk, enb, rst_s;
    wire [WIDTH-1:0] q;
    wire cnt_max;

    contador_binario_sync #(WIDTH, MAX_VALUE) uut (
        .enb(enb),
        .rst_s(rst_s),
        .ck(clk),
        .q(q),
        .cnt_max(cnt_max)
    );

    initial begin
        clk = 0; enb = 0; rst_s = 1;
        #10 rst_s = 0;
        #10 enb = 1;
        #100 $stop;
    end

    always #5 clk = ~clk;

    initial begin
        $monitor("Time = %0t | q = %0b | cnt_max = %b", $time, q, cnt_max);
    end
endmodule
```

Figura 2 - RTL ilustrativo referente ao circuito Síncrono



Declaração Comportamental

Este código implementa um contador binário síncrono parametrizável em Verilog usando declarações procedurais (always e if-else).

Módulo contador_binario

- **Entradas:**
  - enb: Habilitação do contador.
  - rst_s: Reset síncrono (ativo em nível alto).
  - ck: Sinal de clock.
- **Saídas:**
  - q: Valor atual do contador (em formato binário).
  - cnt_max: Indicador de valor máximo (ativo quando q atinge MAX_VALUE).
- **Parâmetros:**
  - WIDTH: Define a largura (número de bits) do contador.
  - MAX_VALUE: Define o valor máximo do contador.

O bloco always sensível à borda de subida do clock (ck) ou do reset (rst_s) controla o comportamento do contador:

- Se rst_s estiver ativo, o contador (q) é resetado para zero e cnt_max é desativado.
- Se enb estiver ativo, o contador incrementa a cada ciclo de clock. Quando q atinge MAX_VALUE, ele volta a zero e cnt_max é ativado.

Testbench tb_contador_binario

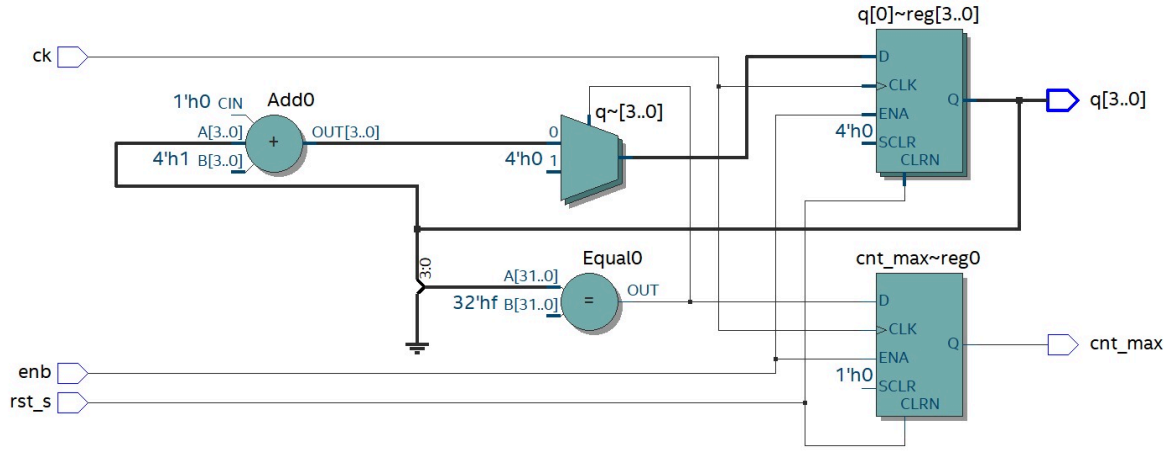
O testbench simula o funcionamento do contador:

- Inicializa os sinais clk, enb e rst_s.
- Aplica estímulos para verificar o comportamento do contador.
- Monitora e exibe os valores das saídas q e cnt_max.

```
module contador_binario #(parameter WIDTH = 4, parameter MAX_VALUE = 2**WIDTH-1) (
    input wire enb, rst_s, ck,
    output reg [WIDTH-1:0] q,
    output reg cnt_max
);
    always @(posedge ck or posedge rst_s) begin
        if (rst_s) begin
            q <= 0;
            cnt_max <= 0;
        end else if (enb) begin
            if (q == MAX_VALUE) begin
                q <= 0;
                cnt_max <= 1;
            end else begin
                q <= q + 1;
                cnt_max <= 0;
            end
        end
    end
endmodule

module tb_contador_binario;
    parameter WIDTH = 4;
    parameter MAX_VALUE = 2**WIDTH-1;
    reg clk, enb, rst_s;
    wire [WIDTH-1:0] q;
    wire cnt_max;

    contador_binario #(WIDTH, MAX_VALUE) uut (
        .enb(enb),
        .rst_s(rst_s),
        .ck(clk),
```

```
    .q(q),
    .cnt_max(cnt_max)
  );

  initial begin
    clk = 0; enb = 0; rst_s = 1;
    #10 rst_s = 0;
    #10 enb = 1;
    #100 $stop;
  end

  always #5 clk = ~clk;

  initial begin
    $monitor("Time = %0t | q = %0b | cnt_max = %b", $time, q, cnt_max);
  end
endmodule
```

Figura 3 - RTL ilustrativo referente ao circuito Comportamental



3. Conclusão

Modo Assíncrono

A implementação do contador binário assíncrono utiliza Flip-Flops Tipo-D conectados em cascata, onde o clock de cada Flip-Flop é derivado da saída do Flip-Flop anterior. Esse método é simples e fácil de implementar, mas pode levar a atrasos na propagação do sinal, especialmente em contadores de largura maior, devido à natureza assíncrona da contagem.

Modo Síncrono

Na implementação síncrona, todos os Flip-Flops recebem o mesmo sinal de clock, o que sincroniza a contagem e elimina os atrasos na propagação do sinal observados no contador assíncrono. Esse método é mais eficiente e rápido, tornando-se adequado para aplicações que exigem precisão temporal.

Modo Comportamental (Always If-Else)

A implementação utilizando declarações comportamentais (always e if-else) é altamente legível e intuitiva. Este método define claramente o comportamento do contador em termos de habilitação (enb), reset síncrono (rst_s) e incremento a cada borda de subida do clock (ck). É uma abordagem prática para descrever o comportamento do hardware de maneira abstrata, facilitando a verificação e modificação do design.

HDL e Ferramentas Online

O EDA Playground é uma excelente ferramenta online para simulação e validação de códigos HDL. Ele permite que se possa verificar designs de hardware de maneira eficiente, sem a necessidade de construir fisicamente o hardware. Utilizando o EDA Playground, é possível identificar e corrigir erros de projeto antecipadamente, economizando tempo e recursos no desenvolvimento de sistemas digitais.

O uso de HDL (Hardware Description Language) como Verilog e VHDL é essencial para o desenvolvimento de sistemas digitais complexos. Ele permite a descrição precisa do comportamento do hardware, facilita a simulação e validação do design antes da implementação física e oferece uma plataforma para otimização e testes rigorosos. A capacidade de testar e validar todo um hardware sem construí-lo fisicamente reduz significativamente os riscos e custos associados ao desenvolvimento de novos produtos.

Referências:

Contador binário, ascendente e descendente. Acesso em 01 de julho. Link: https://dspace.utpl.edu.ec/bitstream/20.500.11962/24499/1/Contador%20ascendente%20y%20descendente%20en%20VHDL%28tutorial%29.pdf

Karnaugh map maker. Acesso em 01 de julho. Link: https://ictlab.kz/extra/Kmap/