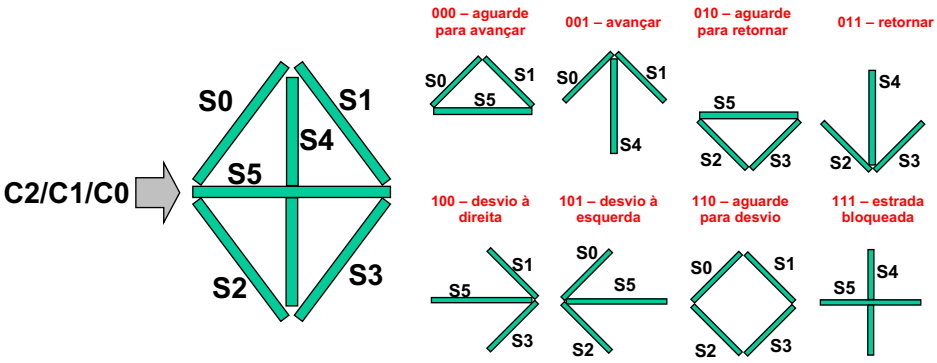
EXERCÍCIOS FSD – Unidades 2 e 3 – 21/abril/2025

1. **Circuitos combinacionais**. Qual é o circuito combinacional equivalente ao código *SystemVerilog* abaixo, e qual a sua função?

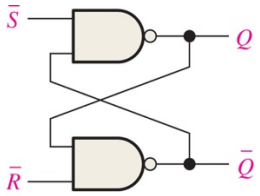
```
assign Y = (S == 2'b00) ? I[0] :
           (S == 2'b01) ? I[1] :
           (S == 2'b10) ? I[2] :
                          I[3];
```

2. **Circuitos combinacionais**. Um sinal luminoso é utilizado pela equipe de controle de tráfego para gerenciar o trânsito em uma avenida. Você deve desenvolver um circuito codificador que recebe como entrada um código de três bits, ***C***, e codifica a saída lógica deste sinal com 6 bits ***S***, conforme ilustrado na figura abaixo:



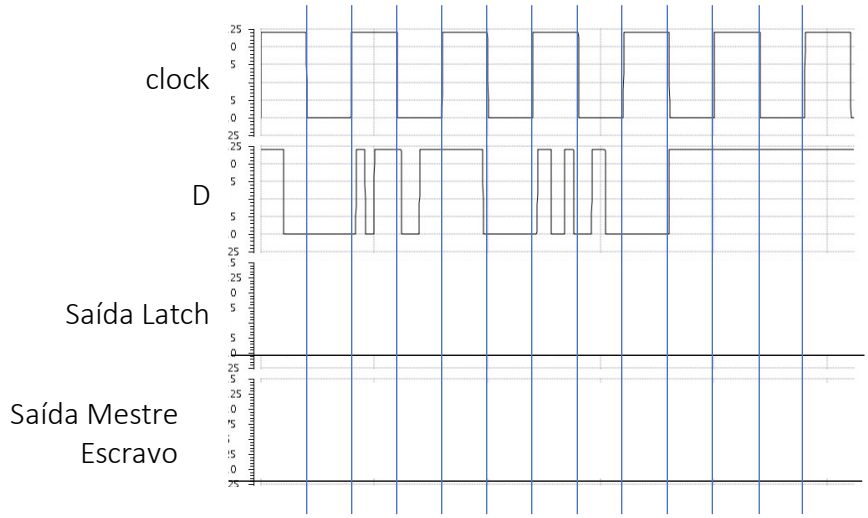
As saídas **S5, S4, S3, S2, S1 e S0** precisam estar ativadas (definidas como `1') para produzir luz no segmento correspondente. Implemente o código para gerar o sinal **S** utilizando *SystemVerilog*.

3. **Elementos de memória**. Considere a *latch* SR abaixo. Complete a tabela verdade.



S	R	Q	$\bar{Q}$	Operação
0	0			
0	1			
1	0			
1	1			

4. **Elementos de memória**. Considere uma *latch* D transparente no nível lógico ‘1’, e um1 *flip-flop* D *mestre-escravo* sensível à borda de subida do clock. Apresentar no diagrama de tempos abaixo a saída esperada para cada elemento de memória.



\

5. **Descrição RTL**. Desenhe o circuito lógico equivalente ao código *SystemVerilog* abaixo.

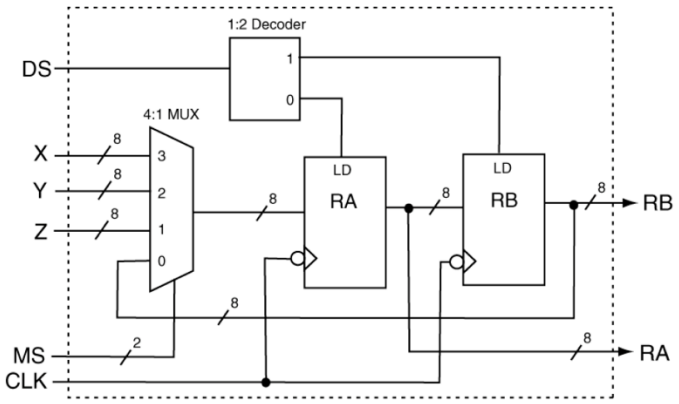
```
module exercicio2 #(parameter int N = 8) (
    input  logic            clock,
    input  logic            reset,
    output logic [N-1:0]    saida
);

    logic [N-1:0] opA, opB;

    assign saida = opA + opB;

    always_ff @(posedge clock or posedge reset) begin
        if (reset) begin
            opA <= '0;
            opB <= '0;
        end else begin
            opA <= opA + 1;
            opB <= saida;
        end
    end

endmodule
```

6. **Descrição RTL.** Considere o circuito descrito pelo código *SystemVerilog* abaixo, que é composto por três registradores de 8 bits, um multiplexador 2x1 e um somador de 8 bits. O circuito possui cinco entradas: dois sinais de 8 bits, chamados ***A*** e **B**; um sinal de seleção do multiplexador, chamado ***sel***; e os sinais de ***clock*** e **reset**. Como saídas, o circuito gera o valor do registrador REGC, denominado ***C***, e a saída do somador, chamada de ***saída***.

```
module rtl (
    input logic clock, reset, sel,
    input logic [7:0] A, B,
    output logic [7:0] saida,
    output logic [7:0] C
);
    logic [7:0] REGA, REGB, REGC;
    logic [7:0] op2;

    assign op2 = (sel == 1'b0) ? REGA : B;

    assign saida = REGB + op2;
    always_ff @(posedge clock or posedge reset) begin
        if (reset) begin
            REGA <= 8'b0;
            REGB <= 8'b0;
            REGC <= 8'b0;
        end else begin
            REGA <= A;
            REGB <= saida;
            REGC <= REGB;
        end
    end
    assign C = REGC;
endmodule
```

Sua tarefa é desenhar o circuito. Utilize blocos para representar cada componente (registradores, multiplexador e somador). Depois, ilustre as conexões entre esses blocos, mostrando como os componentes são interligados. Certifique-se de nomear todos os sinais corretamente.

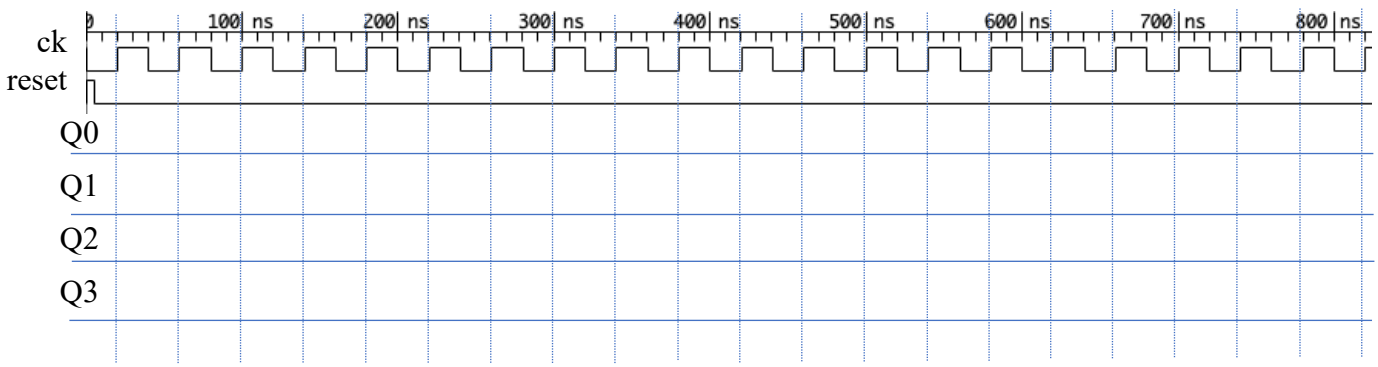
7. **Descrição RTL.** Escreva o *SystemVerilog* completo do seguinte diagrama de blocos RTL.
   a) *entity* e estrutura do código *SystemVerilog*
   b) elementos combinacionais
   c) elementos sequenciais



8. **Descrição RTL.** Considere o circuito descrito abaixo**.**
   a) O módulo "***FLOP***" é um flip-flop latch ou mestre escravo? Justifique.
   b) **Desenhar** o circuito lógico equivalente do módulo "***ckt***", com *flip-flops* e portas *and* e *or*.
   c) Determine o comportamento de "***ckt***" em relação ao sinal do clock (*ck*), explicando o seu comportamento (Floyd, figura 9.9, página 505).
   d) Apresente a forma de onda esperada para este circuito considerando pelo menos os 15 primeiros ciclos de *clock* (abaixo o diagrama de tempos para o *clock* e o reset).
   e) Qual o comportamento deste circuito?

```systemverilog
module FLOP (
    input  logic reset, ck, D,
    output logic Q,
    output logic nQ
);
    always_ff @(negedge ck or
        posedge reset) begin
        if (reset)
            Q <= 1'b0;
        else
            Q <= D;
    end

    assign nQ = ~Q;

endmodule
```

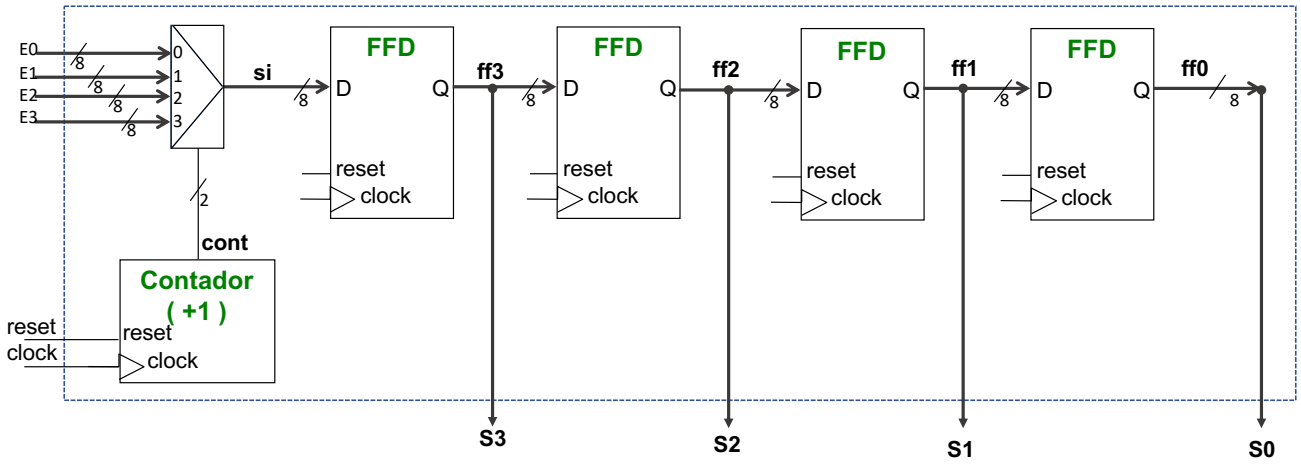
```systemverilog
module ckt (
    input  logic        reset,
    input  logic        ck,
    output logic [3:0]  Q
);
    logic [3:0] nq;
    logic rst_int;

    assign rst_int = reset | (Q[1] & Q[3]);

    // Instâncias dos flip-flops com borda de descida
    FLOP f0 ( .D(nq[0]), .ck(ck),   .reset(rst_int), .Q(Q[0]), .nQ(nq[0]));
    FLOP f1 ( .D(nq[1]), .ck(Q[0]), .reset(rst_int), .Q(Q[1]), .nQ(nq[1]));
    FLOP f2 ( .D(nq[2]), .ck(Q[1]), .reset(rst_int), .Q(Q[2]), .nQ(nq[2]));
    FLOP f3 ( .D(nq[3]), .ck(Q[2]), .reset(rst_int), .Q(Q[3]), .nQ(nq[3]));

endmodule
```

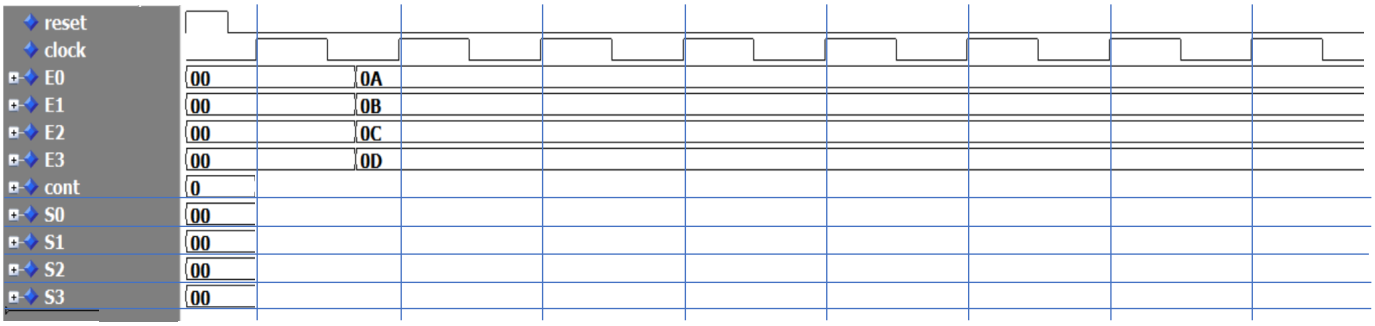


9. **Descrição RTL.** Desenhar o circuito digital equivalente à descrição *SystemVerilog* abaixo utilizando portas lógicas, multiplexadores, registradores e contadores. Indicar no desenho os sinais que são do tipo entrada e saída (utilize uma representação de seta, similar à figura acima), e o número de fios em cada sinal (também similar à figura acima).

```
module RTL (
    input logic reset, clock,
    input logic [7:0] E0, E1, E2, E3,
    output logic [7:0] S0, S1, S2, S3
);
    logic [1:0] cont;
    logic [7:0] mux;

    assign mux = (cont == 2'b00) ? E0 :
                 (cont == 2'b01) ? E1 :
                 (cont == 2'b10) ? E2 :
                 E3;

always_ff @(posedge clock or posedge reset) begin
        if (reset) begin
            S3 <= 8'b0;
            S2 <= 8'b0;
            S1 <= 8'b0;
            S0 <= 8'b0;
            cont <= 2'b00;
        end else begin
            S3 <= mux & ~S0;
            S2 <= ~S3;
            S1 <= ~S2;
            S0 <= S1;
            cont <= cont + 1;
        end
    end

endmodule
```

10. **Descrição RTL.** Considere o circuito digital abaixo, o qual contém um multiplexador, registradores e um contador. Pede-se: completar o diagrama de tempos para os estímulos fornecidos.





11. **Máquina de estados (FSM).** Considere o circuito "encontra_padrão". Este circuito possui uma entrada de dados **Din**, e um **padrão** de **4** bits a ser encontrado no fluxo de bits recebidos em Din. Assumir que padrão seja inicializado no reset do sistema (ou seja, não é alterado durante a execução da busca pelo padrão). A **saída** do circuito é o resultado de um contador, sinal **vezes**, o qual é incrementado a cada vez que ocorrerem 4 bits na sequência Din correspondente a padrão.

```
module encontra_padrao (
    input  logic         clock,
    input  logic         reset,
    input  logic [3:0]   padrao,
    input  logic         Din,
    output logic [3:0]   vezes
);
```

a. Desenhar e explicar a FSM (**não apresentar código *SystemVerilog* nesta questão**).
b. Explique como é feito o controle do contador, e em qual/quais estados deve-se incrementar o contador.

Dica: avalie se sua FSM opera corretamente usando por exemplo: padrão="1010" (3 downto 0) com a sequência:

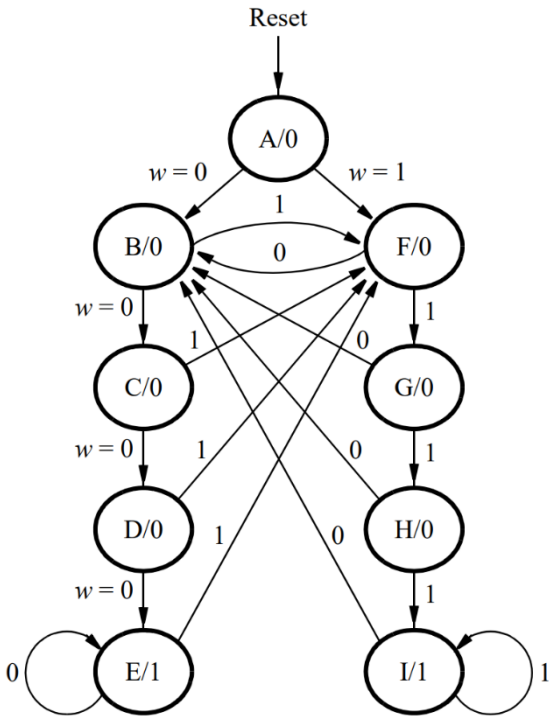
0 → ✓ (ok com padrão(0))
0 → ✓ (não coincidiu padrão(1) mas coincidiu com padrão(0))
1 → ✓ (ok com padrão(1))
0 → ✓ (ok com padrão(2))
1 → ✓ (ok com padrão(3), incrementa o contador)
1 → *x* (deveria ser 0)
0 → ✓ (ok com padrão(0))

12. **Máquina de estados (FSM).** Distribuidor de Café. Este distribuidor vende café a R$ 0,75, aceitando moedas de 25 e 50 centavos. Existe na máquina uma fenda para inserir moedas com um circuito capaz de reconhecer moedas de R$ 0,25 e R$ 0,50, e é capaz de devolver qualquer outro tipo de moeda ou objeto não reconhecido. Além disso, o usuário pode desistir da transação e apertar a tecla DEV que devolve as moedas inseridas até o momento. A devolução de excesso de moedas é automática sempre que o valor inserido antes de retirar um café ultrapassar R$ 0,75. **Café só é ativado se o usuário pressionar ASK e a soma acumulada for R$ 0,75**.

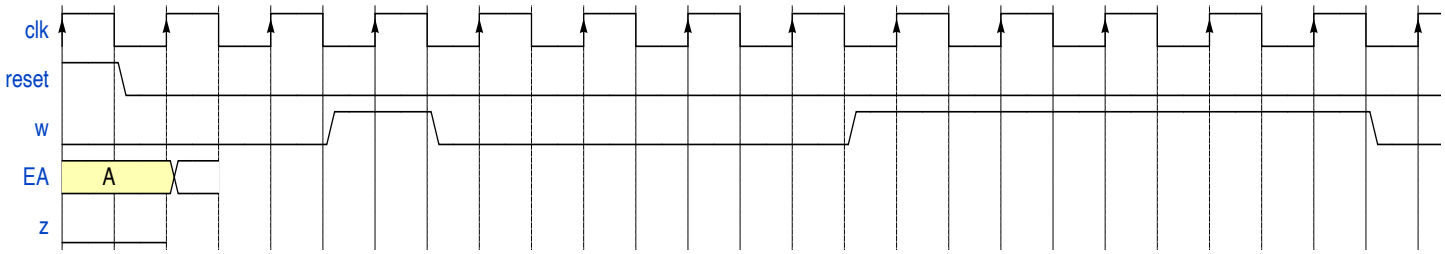
- **Entradas**: M25 (moeda de 25 centavos), M50 (moeda de 50 centavos), DEV (pedido de devolução do valor inserido), ASK (pedido de café), clock, reset. Apenas uma entrada M25/M50/DEV/ASK pode ser ativada por vez – **não** há sobreposição das entradas.
- **Saídas**: D25 (devolução de moeda de 25 centavos), D50 (devolução de moeda de 50 centavos), café (café fornecido).

Modele o distribuidor de café como uma máquina de estados finita (FSM) – desenhar e explicar a FSM (**não apresentar código SystemVerilog nesta questão**). **Apresentar as condições em que cada saída é ativada e apresente um texto sucinto das ações que ocorrem em cada estado.**

13. **Máquina de estados (FSM).** Considere a máquina de estados finita (FSM) de Moore al lado, que possui 9 estados válidos (A, B, C, D, E, F, G, H, I), uma entrada (w) e uma saída (z). Lembre-se que por se tratar de uma máquina de Moore, as saídas dependem unicamente do estado atual (EA) da máquina.



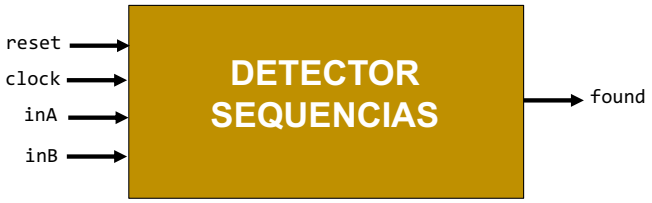
Considerando esta FSM e as entradas de *clk*, *reset* e *w* descritas na forma de onda abaixo, determine o comportamento dos sinais *EA* e *z*. Considere o valor de *w* apenas na borda de subida do sinal *clk*.

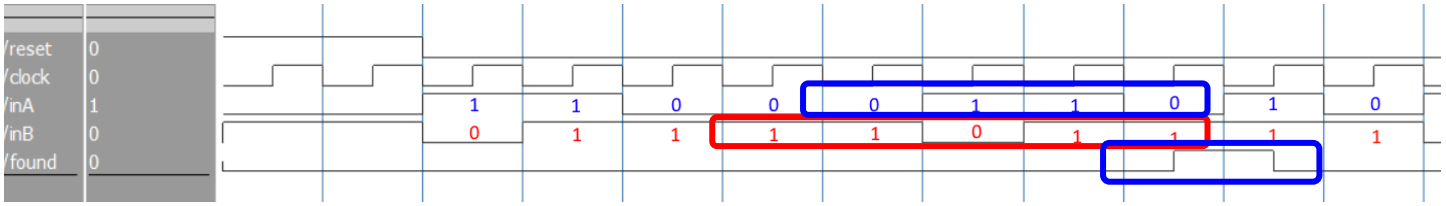


14. **Máquina de estados (FSM).** Desenvolva um circuito capaz de detectar duas sequências de dados em dois canais de entrada distintos. O circuito deve sinalizar a presença **simultânea** das sequências A e B em seus respectivos canais de entrada (**inA** e **inB**) através da saída **found**, que deve assumir o valor '1' quando ambas as sequências forem identificadas ao mesmo tempo.

Sequências a serem detectadas:

- Sequência A: **0110**
- Sequência B: **11011**





1. Desenho do Circuito: Projete seu circuito (utilizando FSM de Moore) para a detecção das sequências no papel. Certifique-se de que o diagrama esteja claro e organizado. Explique o mesmo.
2. Implementação: modele o circuito em SystemVerilog.

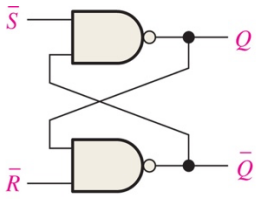
SOLUÇÃO 1

Multiplexador 4 x 1

SOLUÇÃO 2

```
logic [5:0] S;
logic [2:0] C;

assign S = (C == 3'b000) ? 6'b100011 :
           (C == 3'b001) ? 6'b010011 :
           (C == 3'b010) ? 6'b101100 :
           (C == 3'b011) ? 6'b011100 :
           (C == 3'b100) ? 6'b101010 :
           (C == 3'b101) ? 6'b100101 :
           (C == 3'b110) ? 6'b001111 :
                           6'b110000;
```

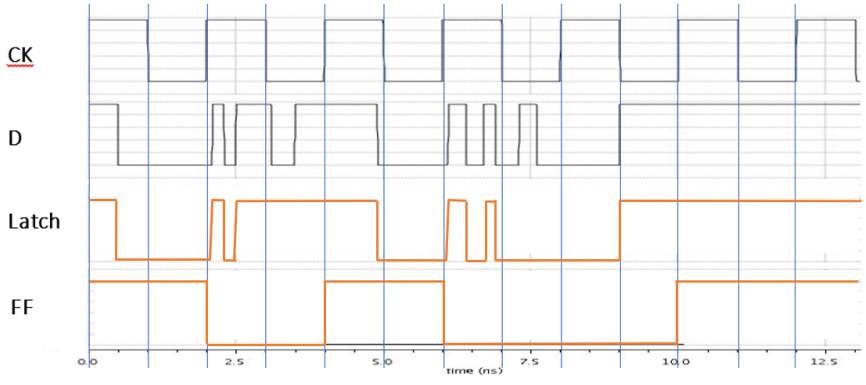
SOLUÇÃO 3



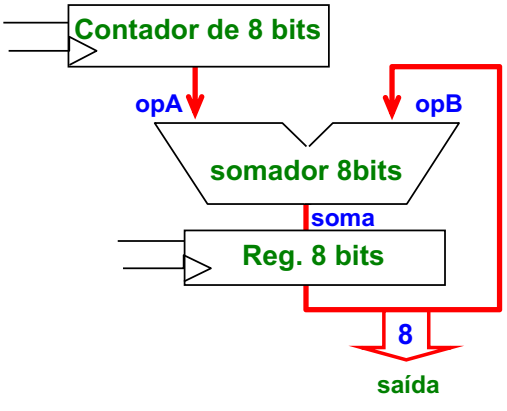
S	R	Q	$\bar{Q}$	Operação
0	0	Q	$\bar{Q}$	Mantém valor
0	1	0	1	reseta
1	0	1	0	seta
1	1	1	1	Inválido

A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

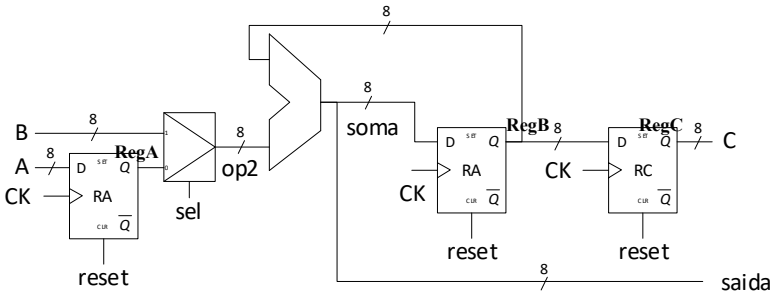
SOLUÇÃO 4



SOLUÇÃO 5

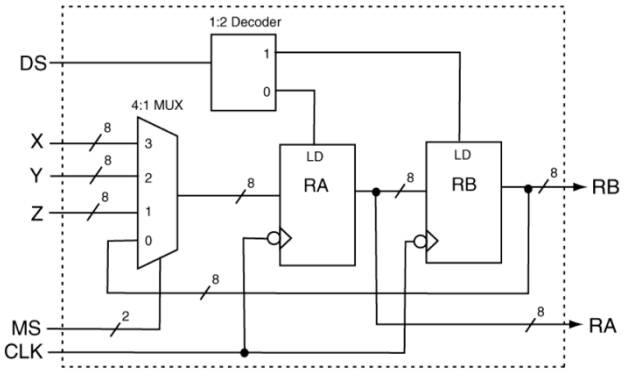


SOLUÇÃO 6



SOLUÇÃO 7

```
module RTL (
    input  logic             DS,
    input  logic             CK,
    input  logic [7:0]       X, Y, Z,
    input  logic [1:0]       MS,
    output logic [7:0]       RA,
    output logic [7:0]       RB
);

    logic [7:0] A, B;

    logic [7:0] outMux;

    assign outMux = (MS == 2'b11) ? X :
                    (MS == 2'b10) ? Y :
                    (MS == 2'b01) ? Z :
                                    B;

    always_ff @(negedge CK) begin
        if (DS == 1'b0)
            A <= outMux;
        else
            B <= A;
    end

    assign RA = A;
    assign RB = B;

endmodule
```



SOLUÇÃO 8 – contador decimal

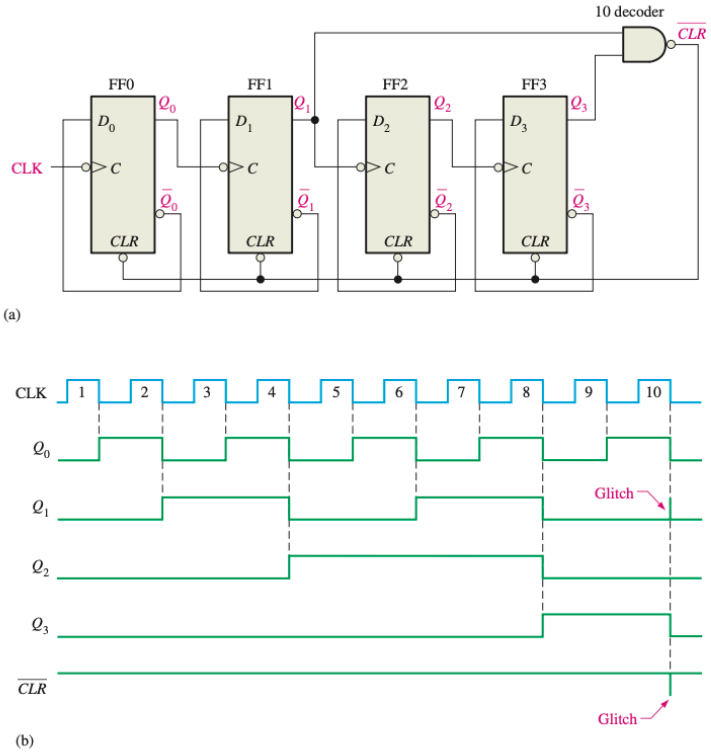
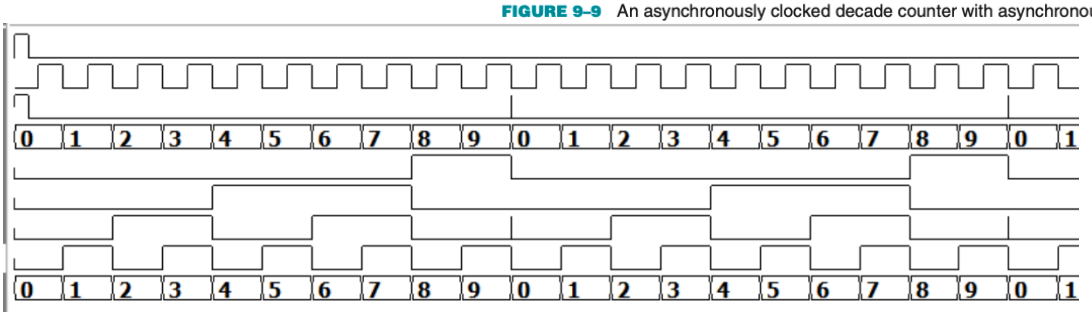
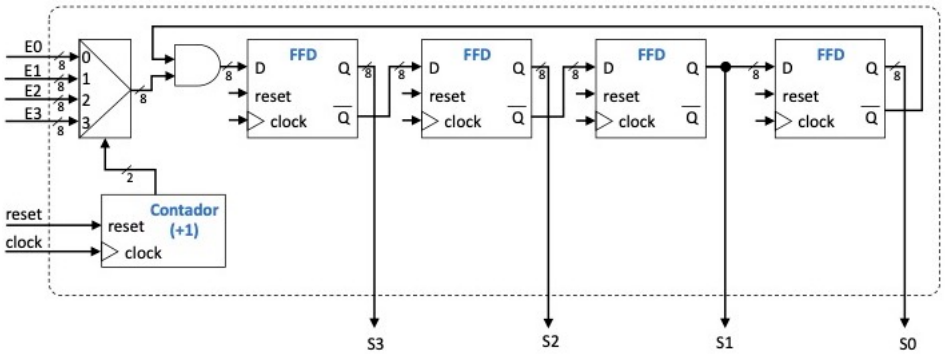


FIGURE 9–9 An asynchronously clocked decade counter with asynchronous recycling.

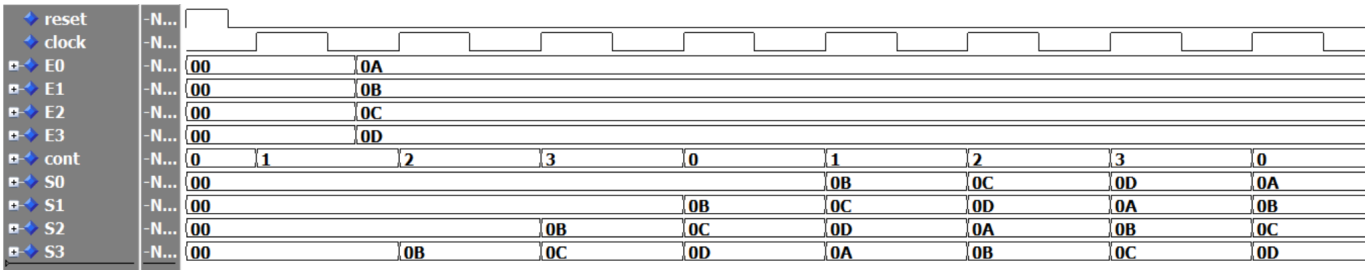


1

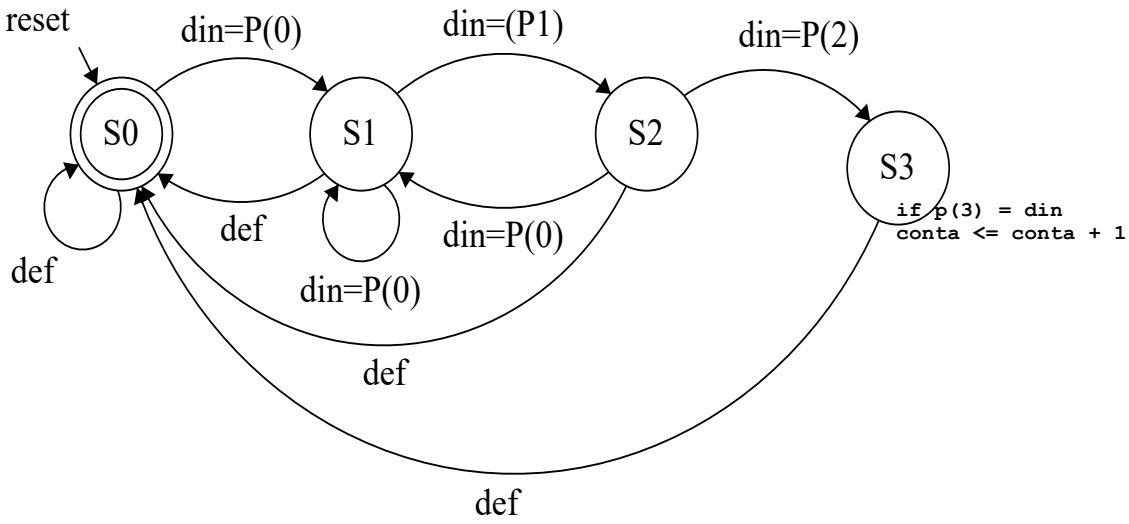
SOLUÇÃO 9



SOLUÇÃO 10

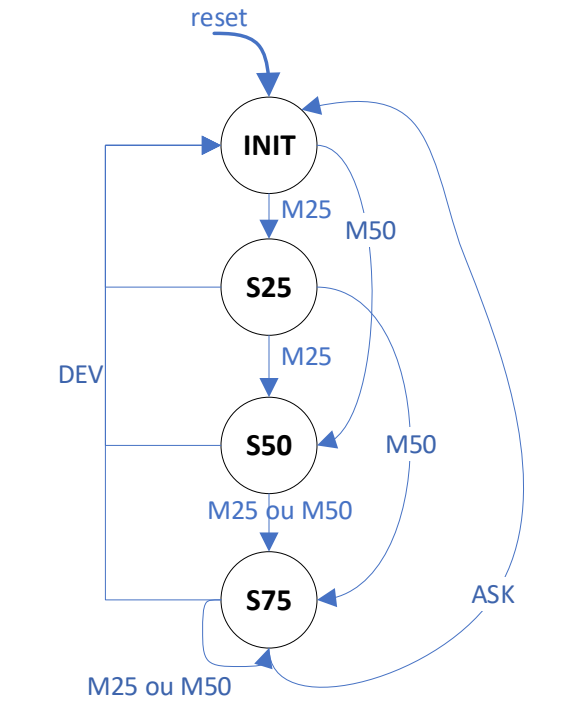


SOLUÇÃO 11

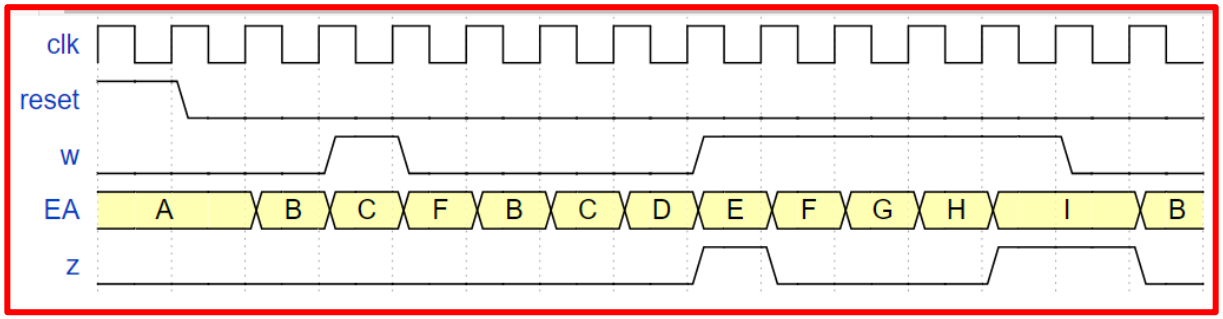


Cont incrementa no estado S3

SOLUÇÃO 12



SOLUÇÃO 13



SOLUÇÃO 14

Sem solução. Dica: usar 2 máquinas de estados.