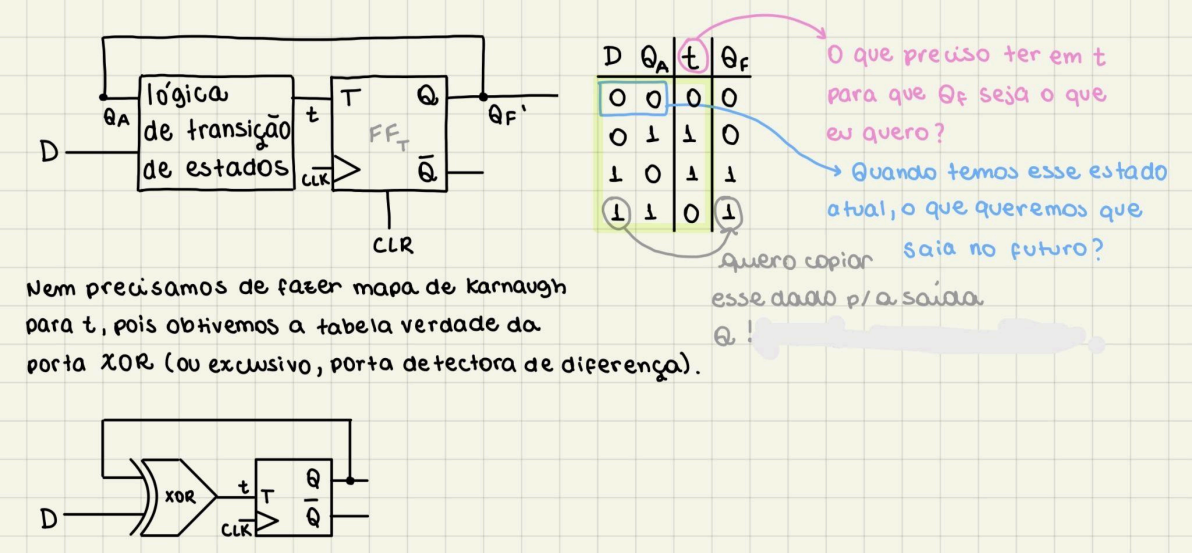
RAQUEL DE PARDE MOTTA matrícula: 854017
Arquitetura de Computadores I - Lista 02

01- Implementar um FF (flip-flop) tipo D usando FF tipo T e portas lógicas elementares/compostas.



O primeiro passo da resolução do problema é compreender que precisaremos de uma lógica de entrada, alimentada pela entrada principal *D* e realimentada pela saída principal *Q* do flip-flop T para conseguir implementar o flip-flop tipo D. O segundo passo é construir a tabela da lógica de transição de estados com coluna auxiliar para que as seguintes perguntas possam ser respondidas:

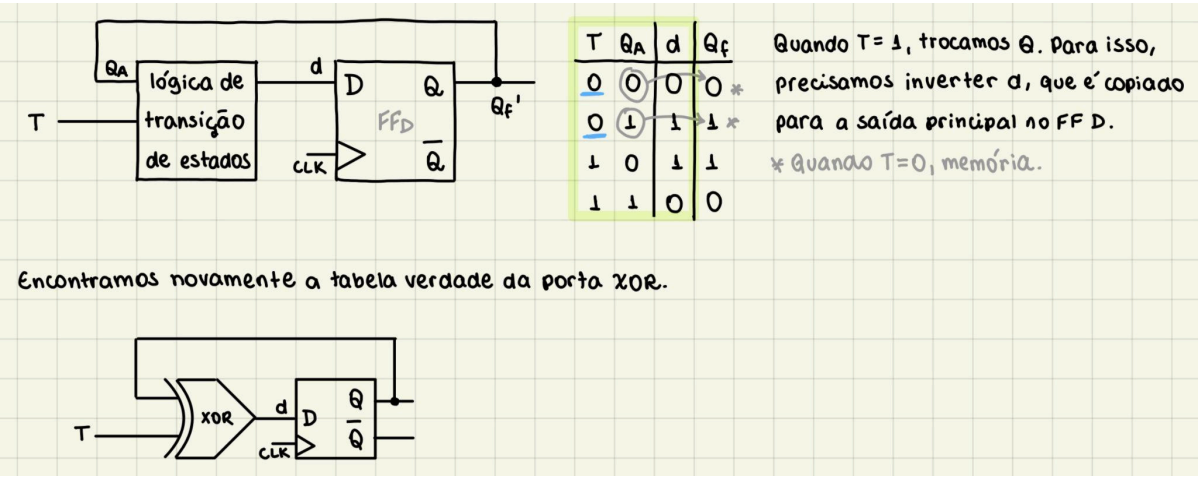
a)Quando temos um determinado estado XX na atualidade, o que quero que saia no próximo estado (“o futuro”)?

b)O que preciso ter em *t* para que o FF T produza essa saída futura que eu quero?

Ao constituir a tabela, percebemos que quando *D* = 0 e *Qa* = 0, ou quando *D* for 1 e *Qa* também for 1, é preciso que o flip-flop faça memória para manter a saída *Qf’* igual, ou seja, que a entrada *t do* FF T seja 0. Agora, se *D* = 1 e *Qa* = 0, ou vice-versa, preciso que o flip-flop faça TROCA para que a saída *Qf’* passe a ser idêntica a *D* e efetue a cópia do dado imputado para a saída principal que é tradicionalmente feita por um FF D.

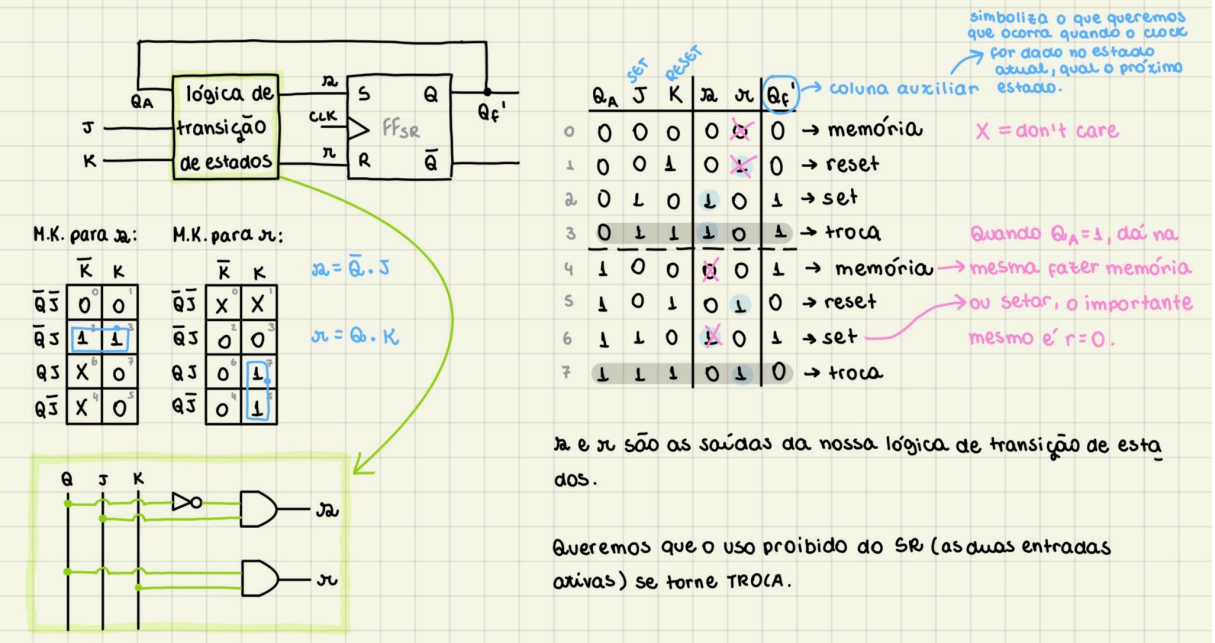
O passo final seria fazer um Mapa de Karnaugh, mas não foi necessário, pois alcançamos a tabela verdade de uma porta composta conhecida: a XOR.

02- Implementar um FF tipo T usando FF tipo D e portas lógicas auxiliares.



A estratégia de resolução segue os mesmos passos da questão 01. Ao montar a tabela desse problema, percebemos que quando $T = 0$, queremos que o valor de Qa seja mantido, independentemente de qual ele for. Já quando $T = 1$ queremos que o valor da saída Qa seja invertido. Para isso, d sempre deve ser igual à Qa quando $T = 0$, e o complemento (oposto) de Qa quando a entrada T estiver ativa para que a troca (toggle) do flip-flop T seja implementada. Novamente obtivemos uma porta XOR e a realimentação da saída do flip-flop para a entrada desta porta lógica na transição de estados.

03- Implementar um FF tipo JK usando FF tipo SR e portas lógicas.



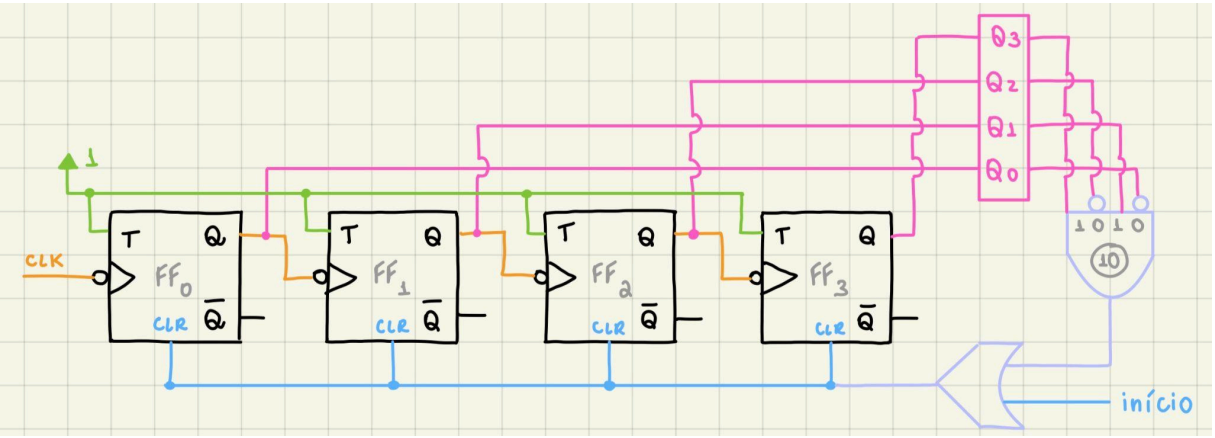
Precisamos transformar o uso proibido do FF SR (quando as duas entradas são ativadas simultaneamente) na função de TROCA do FF JK.
Ao elaborar a TV (tabela verdade) da lógica de transição de estados, temos o seguinte:
-Quando $J = 0$ e $K = 0$, queremos memória, o que significa preservar o valor de Qa qualquer que ele seja.
-Quando $J = 1$ e $K = 0$, setamos Qf (fazemos seu valor ser 1), não importa qual o valor de Qa.
-Quando $J = 0$ e $K = 1$, resetamos Qf (fazemos seu valor ser 0), não importa qual o valor de Qa.
-Quando $J = 1$ e $K = 1$, invertemos a saída principal.

Para os casos de memória, set e reset, basta copiar os valores de J e K para s e r. O valor de Qa só verdadeiramente interfere nos casos em que J e K são ambos 1. Nesses cenários, quando Qa = 0 temos que setar para inverter seu valor e quando $Qa = 1$ temos que resetar o flip-flop SR para que sua saída principal passe a ser zero.

Se fizermos uma análise mais profunda, no entanto, veremos que em alguns casos setar e fazer memória ou resetar e fazer memória produzem o mesmo resultado, e portanto teremos campos de "don't care".

Por fim, ao elaborar os Mapas de Karnaugh, conclui-se que a lógica de transição de estados que será conectada ao FF SR para transformá-lo em um FF JK é composta por duas portas AND e uma porta NOT.

04- Implementar um contador BCD assíncrono, sensível à borda de descida, com sinal de CLEAR.



Um contador BCD constituído por 4 flip-flops possui 4 bits de saída e, portanto, vai de 0 a 9. Para contar números com dois algarismos, precisaríamos de 8 bits e de 8 flip-flops.

Um contador comum de 4 flip-flops seria de módulo 16, contando de 0 a 15 ciclicamente. Como estamos tratando de um ripple counter BCD que contará ciclicamente de 0 a 9, precisamos usar uma porta AND para detectar o output do número binário 1010 (10 em decimal) e então resetar todos os flip-flops ao mesmo tempo com um CLEAR assíncrono (fazendo com que todos os FF retornem ao 0 imediatamente e reiniciem a contagem a partir daí). A saída desta porta AND está ligada a uma OR, cuja segunda entrada é a de início do contador para que se possa fazer o startup do sistema.

É importante notar que se a porta AND detectasse o número decimal 9, ele não seria mostrado na saída por tempo suficiente para ser lido, pois todos os FFs seriam instantaneamente resetados com o CLEAR assíncrono sem aguardar pela próxima borda de descida do clock. Por isso, usamos o número 10 como referência para reiniciar a contagem. Assim o contador efetivamente conta de 0000 a 1001 e ao tentar avançar para 1010, é resetado para 0000 de imediato.

É importante ressaltar que, em um contador assíncrono, os FFs não recebem o clock simultaneamente; em vez disso, cada flip-flop é acionado pela saída Q do flip-flop anterior. O clock se propaga, um FF após o outro. Isso significa que os clocks dos flip-flops subsequentes estão defasados em relação ao clock do flip-flop inicial. Portanto, utilizar um clear síncrono em um ripple counter não resetará todos os flip-flops simultaneamente, pois eles não compartilham o mesmo clock. Por esse motivo, não poderíamos usar um CLEAR síncrono que detecta o 9 e apenas muda as saídas depois da transição ativa do clock (dando tempo para a leitura antes do reset), pois cada FF seria resetado em um momento diferente e as saídas seriam incorretas.

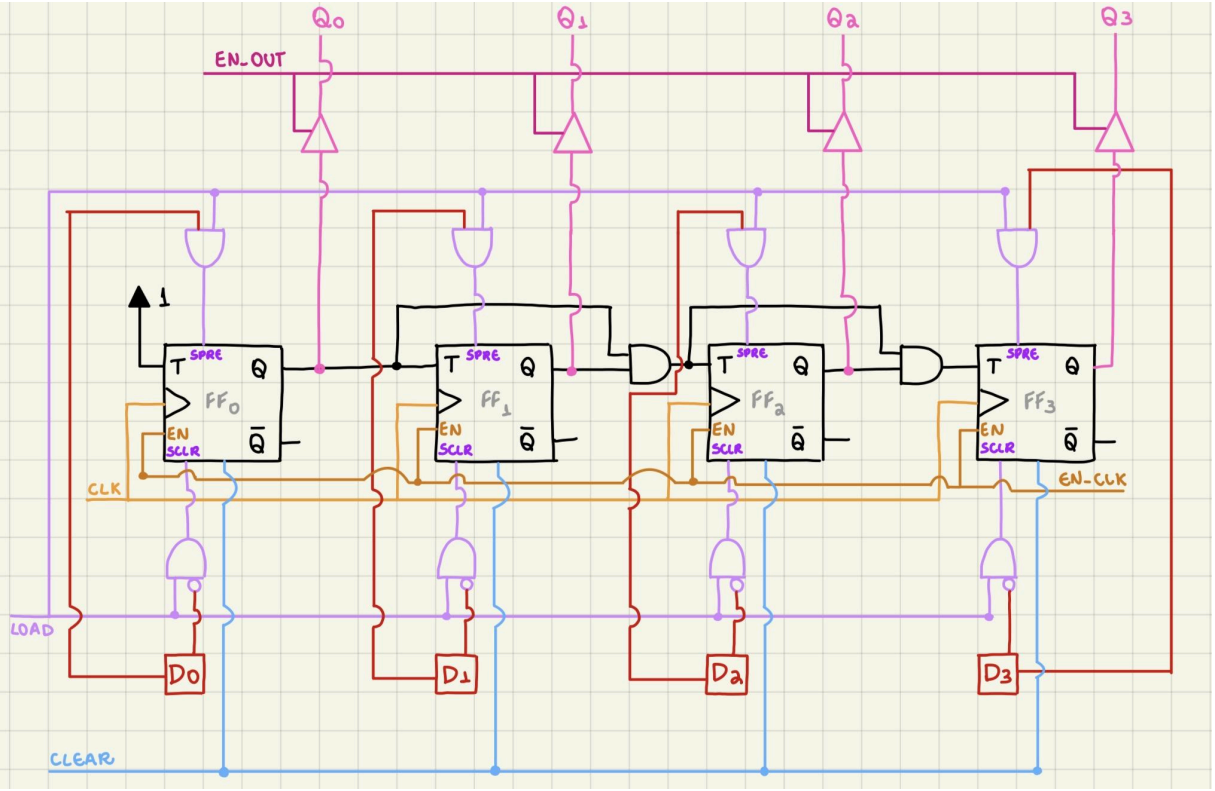
Exemplo de passo a passo do funcionamento:

1. Estado Inicial (CLR de início foi ativado momentaneamente): Todos os flip-flops estão em 0 (Q3 Q2 Q1 Q0 = 0000).
2. Primeira Transição (Clock Externo faz borda de descida):
    - FF0: Clock externo desce de 1 para 0 → FF0 inverte seu estado (Q0 passa de 0 para 1).
    - FF1: Não sofre alteração, pois seu clock (Q0) não mudou ainda.
    - Contagem = 1 (0001)

3. Segunda Transição (Clock Externo):
    - FF0: Clock externo desce novamente → Q0 inverte de 1 para 0.
    - FF1: Clock (Q0) desce de 1 para 0 → FF1 inverte seu estado (Q1 passa de 0 para 1).
    - FF2 e FF3: Não sofrem alteração.
    - Contagem = 2 (0010)
4. Terceira Transição (Clock Externo):
    - FF0: Q0 inverte de 0 para 1.
    - FF1: Clock (Q0) sobe de 0 para 1 → FF1 não muda (só muda na borda de descida).
    - Contagem = 3 (0011)

E assim por diante… até chegar no 10 e depois reiniciar.

05- Implementar um contador módulo 16 síncrono, sensível à borda de subida, com carga paralela síncrona, sinal de LOAD, EN_CLK, EN_OUT (enable output) e CLR assíncrono.



Este contador utiliza quatro flip-flops tipo T. Possui carga paralela síncrona de dados controlada pelo sinal **LOAD**, entradas síncronas de preset (**Spre**) e clear (**Sclr**) ativadas com nível alto (1), habilitação do clock (**EN**), saída com buffer de três estados controlada por **EN_OUT** e um **CLEAR** assíncrono para inicialização.

Flip-Flops T: Invertem o estado na borda de subida do clock quando T = 1. Formam o núcleo do contador para a contagem binária síncrona.

Clock (CLK): Sincroniza todos os flip-flops; mudanças ocorrem simultaneamente na borda de subida.

Habilitação do Clock (EN): Controla a contagem. **EN = 1** permite a contagem; **EN = 0** pausa o contador, mantendo o valor atual.

Carga Paralela Síncrona (LOAD): Quando **LOAD = 1**, permite carregar dados específicos nos flip-flops na próxima borda do clock.

Entradas Spre e Sclr:
-Spre: Ligada a uma porta AND que recebe o dado e LOAD. Ativa o set síncrono quando o dado é 1.
-Sclr: Ligada a uma porta AND que recebe o dado invertido e LOAD. Ativa o reset síncrono quando o dado é 0.

Dados a Serem Carregados: Especificam o valor a ser carregado nos flip-flops durante a carga paralela.

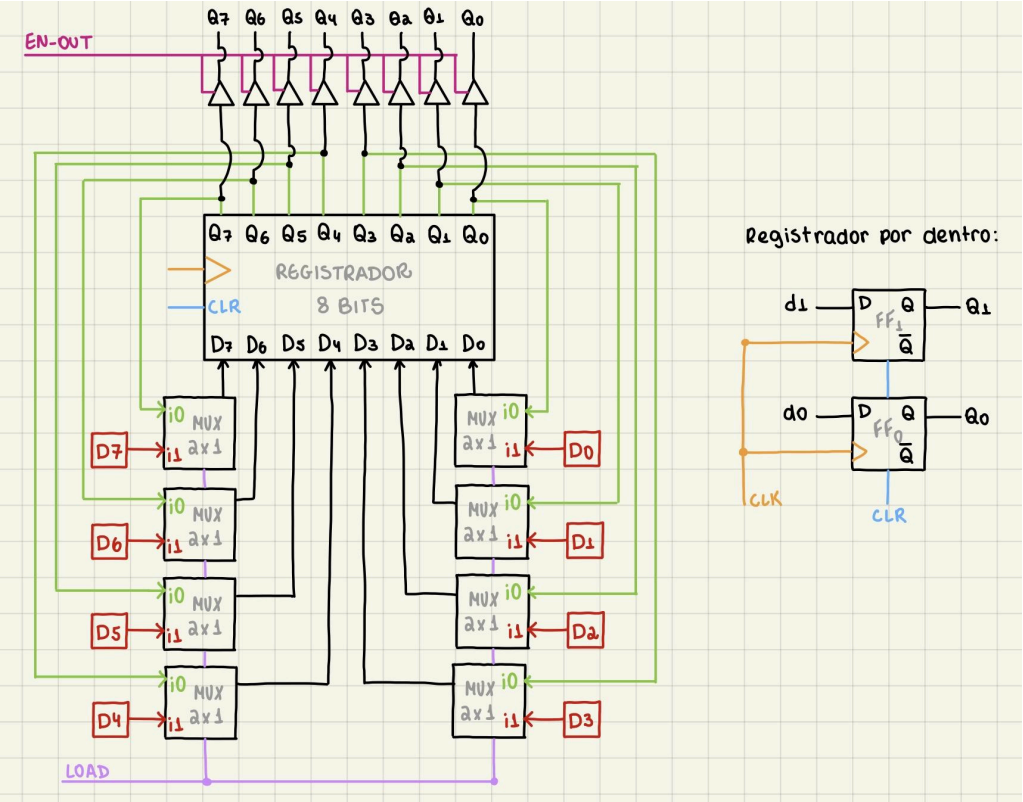
Enable Output (EN_OUT): Controla as saídas do contador. EN_OUT = 1 ativa as saídas; EN_OUT = 0 coloca-as em alta impedância (três estados).

Clear Assíncrono (CLEAR): Reseta imediatamente todos os flip-flops para 0, independente do clock, inicializando o sistema.

Operação Geral:

- Contagem Normal: EN = 1, LOAD = 0; o contador incrementa a cada borda de subida do clock.
- Carga Paralela: LOAD = 1; dados são carregados nos flip-flops via Spre e Sclr na próxima borda do clock.
- Pausa na Contagem: EN = 0; o contador mantém o valor atual, ignorando o clock.
- Controle de Saída: EN_OUT determina se as saídas estão ativas ou em alta impedância.

06- Implementar um registrador de dados de 8 bits, com enable output e carga paralela síncrona controlada pelo sinal de load.



Um registrador de dados de 8 bits formado por 8 flip-flops tipo D funciona como um dispositivo de armazenamento que captura e mantém um conjunto de 8 bits simultaneamente. Cada flip-flop D armazena um bit individual, e todos compartilham o mesmo sinal de clock, garantindo que os dados sejam armazenados de forma síncrona na borda ativa do clock. O controle de carga paralela síncrona é realizado pelo sinal de load; quando este sinal está ativo, os dados presentes nas entradas dos flip-flops são carregados no registrador na próxima borda do clock. Se o load estiver inativo, os flip-flops mantêm seus estados anteriores, ignorando mudanças nas entradas de dados. O recurso de enable output permite controlar a disponibilização das saídas do registrador. Quando o enable output está ativo, os dados armazenados nos flip-flops são disponibilizados nas saídas, permitindo que outros componentes do sistema os leiam. Quando desativado, as saídas entram em estado de alta impedância (três estados), isolando o registrador do barramento ou circuito externo e evitando conflitos de sinal em sistemas onde múltiplos dispositivos compartilham o mesmo barramento de dados. Assim, este registrador permite não apenas o armazenamento controlado e síncrono de dados de 8 bits, mas também o gerenciamento da comunicação com outros componentes do sistema por meio do controle das saídas.

Para fazer com que, quando o sinal de **load** estiver inativo, os flip-flops tipo D mantenham seu estado (ou seja, "façam memória" e não mudem a saída), é necessário controlar as entradas **D** dos flip-flops de forma que eles recebam seus próprios valores de saída quando **load** estiver inativo, e os novos dados quando **load** estiver ativo. Para isso, utilizamos 8 multiplexadores 2x1.

Entrada D de cada Flip-Flop: Conectada à saída de um multiplexador 2x1.

Entradas do Multiplexador:

-Entrada 0 (quando load = 0): Conectada à saída Q do próprio flip-flop (realimentação).

-Entrada 1 (quando load = 1): Conectada ao bit correspondente do dado externo a ser carregado.

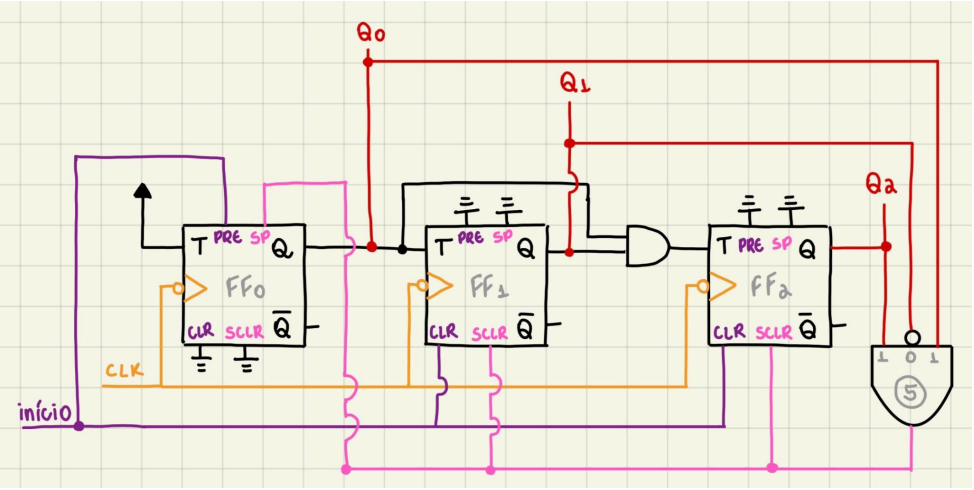
Seleção do Multiplexador: Controlada pelo sinal load.

Funcionamento:

Quando load = 0 (inativo): O multiplexador seleciona a entrada 0. A entrada D do flip-flop recebe o valor da saída Q atual. Na próxima borda de clock, o flip-flop mantém seu estado, pois D = Q.

Quando load = 1 (ativo): O multiplexador seleciona a entrada 1. A entrada D do flip-flop recebe o dado externo. Na próxima borda de clock, o flip-flop carrega o novo dado.

07- Implementar um contador síncrono que conte indefinidamente a sequência 1, 2, 3, 4, 5, 1, 2 … A cada pulso negativo de clock a contagem avança. O sinal de início inicia a contagem em 1.



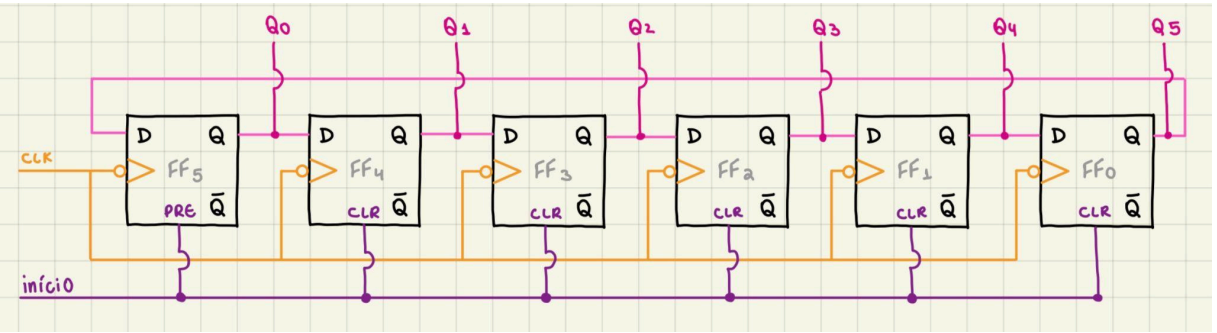
Um contador comum de 3 flip-flops seria de módulo 8, contando de 0 a 7 ciclicamente. Como estamos tratando de um sistema que contará ciclicamente de 1 a 5, precisamos usar uma porta AND para interromper o ciclo ordinário ao detectar o output do número binário 101 (5 em decimal) e então ativar os presets e clears síncronos necessários para voltar ao 001 e recomeçar a contagem a partir daí.

A entrada de início ativa o preset assíncrono do FF0 e o clear assíncrono de FF1 e FF2, para que, de imediato (sem precisar de clock) o contador produza o output [Q0 Q1 Q2] = {100}, que é o número 1 em decimal.

A porta AND, ligada às saídas principais de cada FF, produz output 1 apenas quando Q0 = 1, Q2 = 0 e Q3 = 1 (ou seja, o contador produz o número 5). Sua saída, por sua vez, ativa o preset síncrono do FF0 e o clear síncrono de FF1 e FF2. Assim, na próxima borda de descida, o contador volta a contar a partir do número 1.

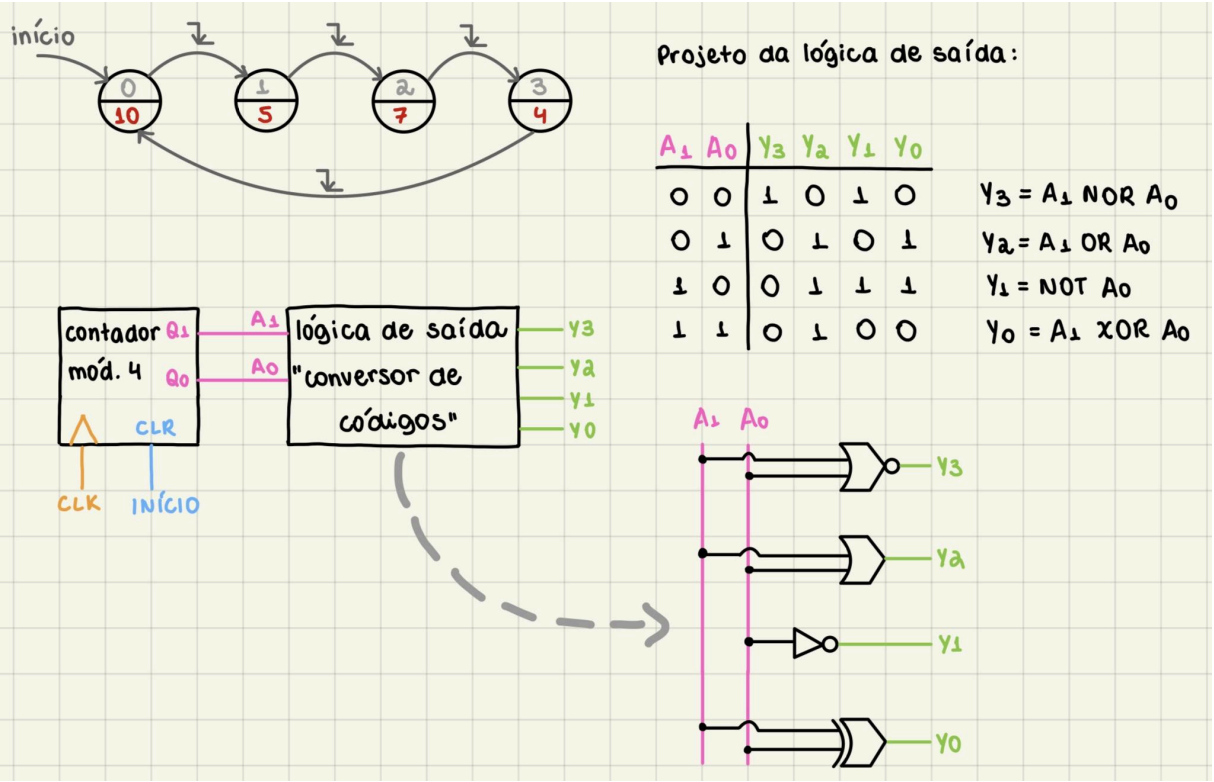
Se a porta AND detectasse o número decimal 5 e sua saída fosse ligada às entradas assíncronas de preset e clear, o número não seria mostrado na saída por tempo suficiente para ser lido, pois os estados de todos os FFs seriam instantaneamente alterados sem aguardar pela próxima borda de descida do clock. Por isso, usamos as entradas síncronas. Elas permitem apenas mudar as saídas depois da transição ativa do clock (dando tempo para a leitura antes de reiniciar o ciclo). Só é possível usar essas entradas pois o contador é síncrono e todos os flip-flops recebem o clock simultaneamente.

08- Implemente um contador em anel sensível à borda negativa de clock que gere a sequência: [Q5 Q4 Q3 Q2 Q1 Q0] = {000001, 000010, 000100, 001000, 010000, 100000} ciclicamente.



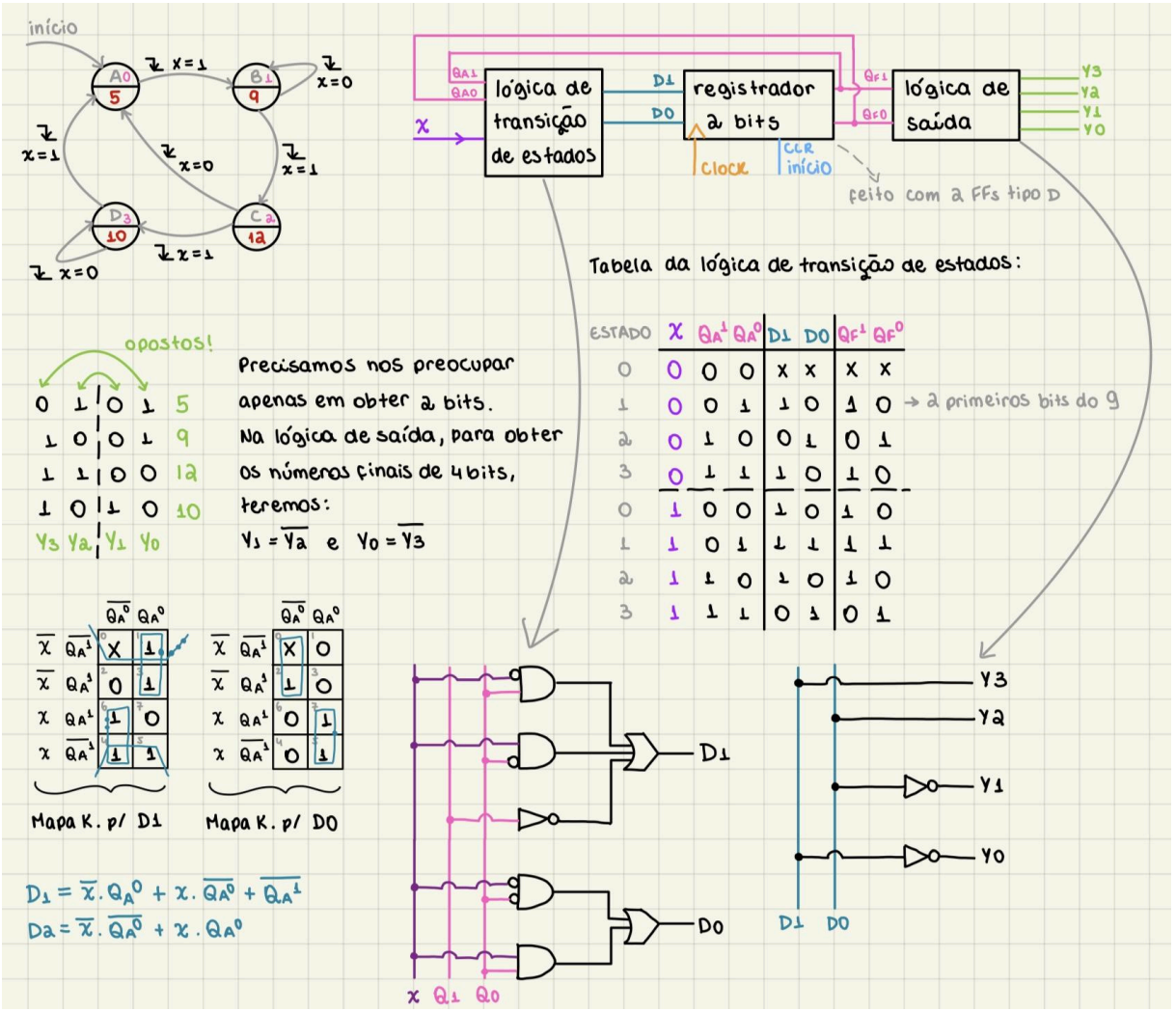
Um contador síncrono em anel é um circuito sequencial onde um único bit de valor '1' circula através de uma série de flip-flops conectados em forma de anel. Todos os flip-flops são disparados simultaneamente pelo mesmo clock, garantindo que as transições ocorram de forma sincronizada. Inicialmente, apenas um flip-flop é setado para 1 e os demais estão em 0. A cada pulso de clock, o bit '1' é transferido para o próximo flip-flop na cadeia, enquanto os demais retornam a '0'. Esse movimento contínuo cria uma sequência cíclica de estados, com o número de estados sendo igual ao número de flip-flops utilizados. Por exemplo, em um contador de 4 bits, a sequência seria: 1000, 0100, 0010, 0001 e então retorna a 1000.

09- Implemente uma máquina de estados finitos que realize o seguinte diagrama de transição de estados:



Utilizaremos um contador básico de módulo 4 para transitar entre os três estados e uma lógica de saída para produzir o output desejado a cada etapa. Para elaborar o “conversor de códigos”, é preciso montar a tabela verdade com os dois bits de saída do contador como sendo as entradas para os 4 bits de saída finais (10, que é o maior output, requer 4 bits para ser representado em binário). Não foi preciso montar Mapas de Karnaugh pois a simples observação da tabela permite deduzir quais portas implementar para manipular A1 e A0 e obter os Y pretendidos.

10- Implemente uma máquina de estados finitos que realize o seguinte diagrama de transição de estados:



A imagem apresenta a elaboração e o funcionamento de uma Máquina de Estados Finitos (MEF), organizada em torno de um registrador de 2 bits e uma lógica de transição de estados, para produzir uma sequência de saídas binárias específicas.

- A máquina possui quatro estados principais: A, B, C e D, cada um representado por valores binários atribuídos aos estados dos flip-flops cujas saídas principais são Q1 e Q0.
- O sinal de entrada X controla a transição entre os estados.
- A tabela de transição de estados especifica as condições de transição para cada estado.
- Para cada combinação de estado e entrada, a tabela define os próximos valores de D1 e D0 (entradas dos flip-flops), que determinarão o próximo estado.
- Essa tabela serve como referência para preencher os mapas de Karnaugh, facilitando a simplificação da lógica para D1 e D0.
- A lógica de saída é responsável por gerar os bits de saída (Y3, Y2, Y1, Y0) a partir dos valores de D1 e D0.
- A lógica de saída usa Y1 = NOT Y2 e Y0 = NOT Y3, conforme indicado, para obter a sequência desejada, garantindo que os bits finais representem corretamente os estados desejados.