ELETRÔNICA DIGITAL

Prof. Luís Caldas

Teoria

Eletrônica Digital

Módulo zero: PROGRAMA DE TEORIA - Disciplinas Eletrônica

Módulo	Cap.	Seção	Pág,	Assunto
1	5	5.1 5.5 5.6 a 5.11	172 a 173 174 a 188 188 a 198	Implementação do latch NE – Tabela de estados, de Excitação, Gráfico de resposta, Implementação dos F/Fs RS. JK, T e D. F/Fs síncronos e problemas seqüenciais.
2	7	7.14	353 a 354	Modelos de Mealy e Moore – Introdução às máquinas de estados finitos – F.S.M. Diagrama de estados – Implementação de lógica de estados codificados e Exercícios
3	7	7.14	354 a 362	Equações de estados – Implementação de lógica de estados não codificados. Exercícios.
4	7	7.14	363	Exercícios de F.S.M
5	7	7.1 e 7.2	307 a 310	Contador Assíncrono – Divisão de freqüência – Módulo 10 e outros módulos.
6	7	7.3 a 7.7 7.9,7.10	311 a 320 335 a 345	Contador Síncrono – Entradas paralelas crescente/decrescente, módulo de divisão e projeto de contador módulo variável. Projeto de motor de passo e outros.
7	7	7.15 a 7.17	377 a 389	Registrador de deslocamento, serial, paralelo – Conversão série-paralelo e vice versa, e implementação de contador.
8			Notas de aulas	Registrador de deslocamento esquerda para a direita e da direita para a esquerda. Contador Johnson, e anel.

Módulo zero: Descrição dos capítulos a serem abordados em cada módulo funcional de aula.

Objetivo: Estudo da lógica seqüencial. Estudo de sub-sistemas digitais como: Flip-Flops, contadores síncronos e assíncronos e registradores de deslocamentos. Solução de problemas seqüenciais, implementação das máquinas de estados e solução de problemas.

O módulo 1 estuda o elemento de memória chamado de latch-NE. Como é um elemento primitivo ele passa a ser utilizado na implementação de outros elementos de memória. Os elementos de memória servem como aplicações em contadores, registradores, memórias e outros. Os flip-flops são criados associando uma lógica de transformação do latch-NE em qualquer tipo de Flip-flop. Os fli-flops podem então atuar como blocos primitivos na aplicação de circuitos como contadores, registradores, memórias, divisores de freqüência e outros. Nesse mesmo módulo pode-se fazer a síntese de circuitos seqüenciais na solução dos problemas.

O estudo é somente voltado aos circuitos seqüenciais síncronos cujo relógio é referência de tempo para os circuitos. É introduzido o conceito de borda de subida e descida, os parâmetros tempo de preparação "set-up" e de manutenção "hold" dos dispositivos integrados. São desenvolvidos os flip-flops síncronos e sensíveis à borda do relógio. São realizados para cada flip-flop a forma de onda da resposta de saída tanto para a borda de subida como para a borda de descida. É implementado o diagrama de estado para cada flip-flop e a sua equação de estado.

O módulo 2 consiste no estudo dos modelos de descrições de sistemas seqüenciais conhecidos como Mealy e Moore. As equações de estados e saída de cada modelo e sua representação como F.S.M. é uma introdução às máquinas de estados finitos F.S.M. As F.S.M. são máquinas tipicamente seqüenciais e, portanto, descrevem somente problemas cujos eventos ocorrem de uma forma seqüencial e não concorrente ou paralela. Para esses casos uma outra ferramenta de descrição de sistemas é citada e é tópico a ser estudado em sistemas digitais. As máquinas de estados são desenvolvidas pela modelagem do problema através de diagrama de estados que pode ser usando o modelo de Mealy ou usando o modelo de Moore. Uma vez o modelo é desenvolvido um procedimento é usado para a implementação por tabela de estados e saída por estados codificados e flip-flops. São realizadas algumas simulações com a resposta da F.S.M.

O módulo 3 foca a implementação da F.S.M. usando um flip-flop por estado. Apesar de crescer o número de flip-flops simplifica a lógica de implementação. Dos diagramas de estados ou por Mealy ou por Moore são retiradas as expressões booleanas dos estados e saída e através dessas equações booleanas e dos flip-flops são implementados os circuitos. Uma série de novos exemplos como de codificação de transmissão de dados tipo RZ, NRZ, Manchester e outros são exemplos de implementações.

O módulo 4 é a implementação da F.S.M por dispositivo de lógica programável PLDs. É introduzido os blocos memória apenas de leitura MROM e o arranjo lógico programável ALP. Das tabelas de estados e de saída e a inclusão de elementos de memória com flip-flop tipo D transparente a ROM ou ALP realizam a lógica de implementação da F.S.M. Vários exemplos de F.S.M. são implementados e exercícios completos são resolvidos.

O módulo 5 é uma aplicação da associação de flip-flops tipo T na construção de contadores assíncronos. São construídos os contadores tipo ripple, conhecidos como assíncronos, modo crescente e decrescente e uma análise do funcionamento com formas de ondas de entrada e saída. É introduzido o conceito de módulo do contador, ou o número de estados percorridos pelo contador, O aluno tem a clara idéia de módulo da divisão de freqüência. São construídos vários divisores de freqüências com um contador universal de 4 bits. São montadas as tabelas da verdade para cada módulo de divisão e a realimentação necessária para se obter o módulo desejado. São mostrados os "glitches" de saída quando da passagem de um estado para outro durante as divisões de freqüências de módulos diferentes de 2^n.

O módulo 6 é uma aplicação da associação de flip-flop tipo T na construção de contadores síncronos. São construídos os contadores cujas troca de estado é simultânea com o mesmo sinal de relógio é aplicado simultaneamente a todos os flip-flops. São implementados os modos crescentes e decrescentes e introduzido o conceito de entradas paralelas síncronas e assíncronas. São realizados muitos exemplos de implementações utilizando as entradas paralelas. São simuladas as formas de ondas geradas pelas saídas do flip-flops com relógio borda de subida. Vários exemplos mostram o conceito de saída como sensíveis a borda positiva ou negativa e ambas sincronizadas com sinal de relógio.

O módulo 7 é uma aplicação da associação de flip-flops tipo D na construção de um registrador. Como um bloco da lógica seqüencial o registrador pela sua flexibilidade funcional é um elemento mais importante para a construção de arquiteturas de fluxo de dados e para níveis de projeto RTL. É introduzido o conceito de atraso sincronizado com o relógio e um exemplo de um MODEM com a conversão dos dados de série para paralelo e de paralelo para série.

É implementado 1 bit de um registrador de deslocamento com entrada serial esquerda para a direita e entradas paralelas sincronizadas com o relógio. Uma segunda aplicação é a construção de um contador de módulo variável e a montagem da malha de estados com deslocamentos da esquerda para a direita com controle lógico da entrada serial. Uma terceira aplicação do registrador também na construção de contadores módulo variável pelas entradas paralelas e síncronas com o relógio. No modo serial se consegue uma malha de estados em diversa de estados e múltiplos de potência de 2 com adição unitária dependendo do valor do bit de entrada. No modo paralelo qualquer estado pode ser alcançado através da carga paralela.

O módulo 8 é uma continuação da aplicação de registrador com a construção de contadores em anel e do tipo Johnson. São implementadas algumas aplicações que utilizam esse contador e mostrada uma aplicação de processamento de sinais como gerador randômico e pseudo-aleatório. É implementado um gerador de seqüência máxima de estados. Para o deslocamento da direita para a esquerda é realizada uma ligação física e através das entradas e saídas paralelas é realizada a ligação do bit n para a entrada n-1 e assim por diante, de modo que o bit menos significativo seja a saída serial do registrador e a entrada n a entrada serial do registrador. É construída a malha de estados que será idêntica a malha de estados obtida no modo serial esquerda para a direita, mas as evoluções invertidas. É como inverter a seqüência de estados. São realizados vários exemplos de implementações.

Bibliografia

Referência: Livro Texto - Sistemas Digitais Princípios e Aplicações - Tocci, R. J, Widmer, N. S e Moss, G.L. – ano 2011 - 11.a edição – Editora Pearson.

Outras Referências

1. Sistemas digitais projeto, otimização e HDLs – Vahid, F – ano 2007 – 1.a edição - Editora Bookman.
2. Eletrônica Digital Moderna e VHDL – Pedroni, V.A. - ano 2010 – 1.a edição – Editora Elsevier.
3. Sistemas Digitais uma abordagem integrada – Uyemura, J.P. ano 2000 – 1.a edição – Editora Thomson pioneira.
4. Sistemas Digitais – Fundamentos e Aplicações – Floyd, T – ano 2007 – 9.a edição – Editora Bookman Companhia ED
5. Introdução aos Sistemas Digitais - Ercegovac, M. D, Thomas, L e Moreno, J. H. - ano 2000 1.a edição – Editora Bookman Companhia ED

MÓDULO UM: Estudo dos sistemas sequenciais e desenvolvimento de elemento de memória latch-NE e nascimento dos flip-flops tipos T, RS, JK e D, solução de problemas sequenciais e transformações dos flip-flops.

Objetivo: Introdução aos Sistemas sequenciais e o desenvolvimento de uma máquina de estados cuja saída depende do estado atual do sistema e das entradas externas. Os estados atuais definidos como estados internos são armazenados na memória do sistema e é conhecido como a memória de estado. O circuito evolui de estado lógico para o próximo estado ou estado futuro pela combinação da entrada externa com o estado lógico da memória. Este novo estado passa a ser o estado atual e é armazenado na memória do sistema, assim alterando o estado lógico da memória. Para uma sequência de eventos, o sistema executa uma sequência de estados. Como a memória é finita o número de diferentes estados que o circuito pode percorrer será um número finito de estados.

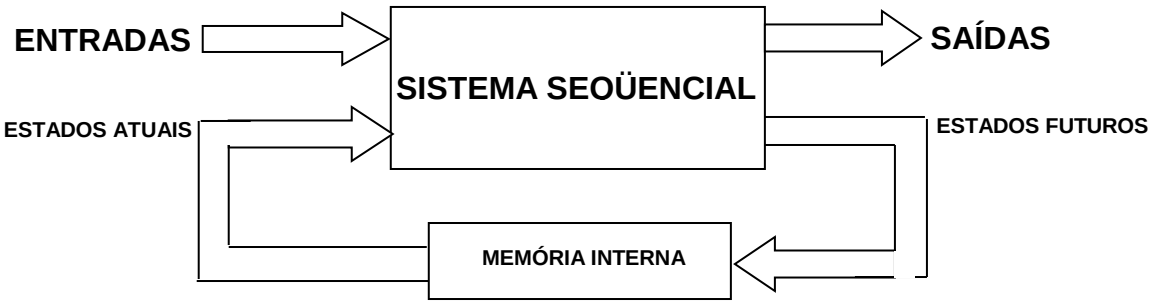
Um sistema no qual a sua saída futura depende não somente das entradas presentes como também do estado atual é chamado de sistema sequencial. Outro exemplo de um sistema sequencial é de um disco acionado por um motor com várias alternativas de giros. O sistema consiste de um motor que impulsiona o disco, um sensor X de posição e um botão de partida, no caso 2 entradas para o sistema. A descrição do movimento realizado pelo disco é a seguir. Após o acionamento do botão de partida, o disco deve ser impulsionado e largar a posição inicial monitorada pelo sensor X. O disco gira no sentido horário e quando novamente chegar à posição inicial, onde X =1, o disco deve inverter o sentido de rotação e girar 2 voltas no sentido anti-horário e parar na posição inicial. O sistema que descreve o modelo de acionamento do disco necessita de elementos de memória que indicam para o controle digital, o número de voltas que o disco realizou mais as entradas X e botão de partida e com essas informações comandar o disco. A partir da condição inicial a memória interna do sistema informa ao controle digital em que volta o disco está e daí quando o disco passa pela posição inicial, a entrada X do sistema ativa o controle digital que com as informações da entrada e do número de voltas do disco, decide a próxima ação que será a primeira ação que é de inverter a rotação do motor. A partir deste ponto o controle é informado pelos elementos de memória quando o disco cumpriu as duas voltas e deve parar quando atingir o ponto X. Fica assim caracterizado o sistema sequencial que sem a memória do sistema e somente com as entradas X e botão de partida não é possível implementar o modelo do sistema do disco.

Nos sistemas combinacionais a saída depende exclusivamente das condições das entradas, portanto o sistema não possui memória interna. Os sistemas que operam no modo sequencial podem ser sistemas síncronos onde a resposta de saída só se modifica ao comando de um sincronismo ou assíncrono, onde a resposta se modifica conforme a chegada dos sinais nas entradas do sistema e do tempo de propagação destes por cada bloco lógico. Um circuito que possui memórias internas implementadas através de flip-flops é capaz de realizar diversos tipos de aplicações tais como: contadores, registradores, memórias, geradores de formas de ondas etc...

Para os modos de operações síncronos e assíncronos, dois modelos podem descrever o comportamento dinâmico dos sistemas.

II – MODELO DE MEALY

O modelo de Mealy descreve o comportamento de um sistema sequencial conforme abaixo.



a) Sequencial assíncrono

$S(K+1) = u(K)$
$E(K+1) = \{u(K), E(K)\}$

onde S é a saída e E é o estado atual e K é o instante de tempo.

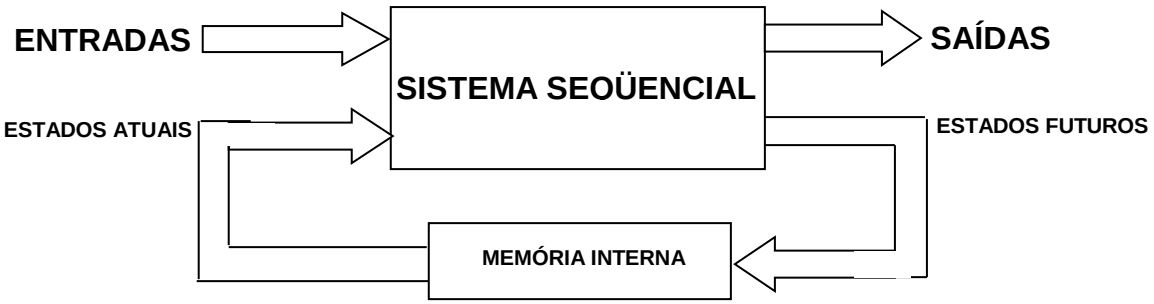
b) Sequencial assíncrono

$S(n+\Delta t) = u(n)$
$E(n+\Delta t) = \{u(n), E(n)\}$

onde n é o instante de tempo atual e $(n+\Delta t)$ e S é a saída e E é o estado atual.

III– MODELO DE MOORE

O modelo de Moore descreve o comportamento de um sistema seqüencial conforme abaixo.



a) Sequencial assíncrono

$S(K+1) = \{u(K), E(K)\}$
$E(K+1) = \{u(K), E(K)\}$
onde S é a saída e E é o estado atual e K é o instante de tempo.

b) Sequencial assíncrono
$S(n+\Delta t) = \{u(n), E(n)\}$
$E(n+\Delta t) = \{u(n), E(n)\}$
onde n é o instante de tempo atual e $(n+\Delta t)$ e S é a saída e E é o estado atual.

IV – DESCRIÇÃO DO SISTEMA ATRAVÉS DA TABELA DINÂMICA DE ESTADOS PRESENTES E FUTUROS.

A construção de uma tabela de estados e saída baseada no modelo de descrição do sistema.

ENTRADAS U(t)	**ESTADOS ATUAIS** Qn(t)	**ESTADOS FUTUROS** Qn+1(t)	**SAÍDAS** S(t)

EXEMPLO: Descrever o comportamento do sistema através de uma tabela dinâmica de estados presentes e futuros, para um sistema descrito a seguir. O sistema possui 2 entradas A e B cuja saída é ALTO sempre que B é ALTO independente do estado presente e é BAIXO sempre que A é ALTO independente do estado presente. Para condições de entrada onde A e B são BAIXO, a saída do sistema não se altera e quando A e B são ALTO, a saída do sistema inverte o estado atual.
Montagem da tabela dinâmica de estados
b) A equação de saída
c) A implementação do circuito lógico que define o comportamento do sistema.

a) Tabela dinâmica de estados

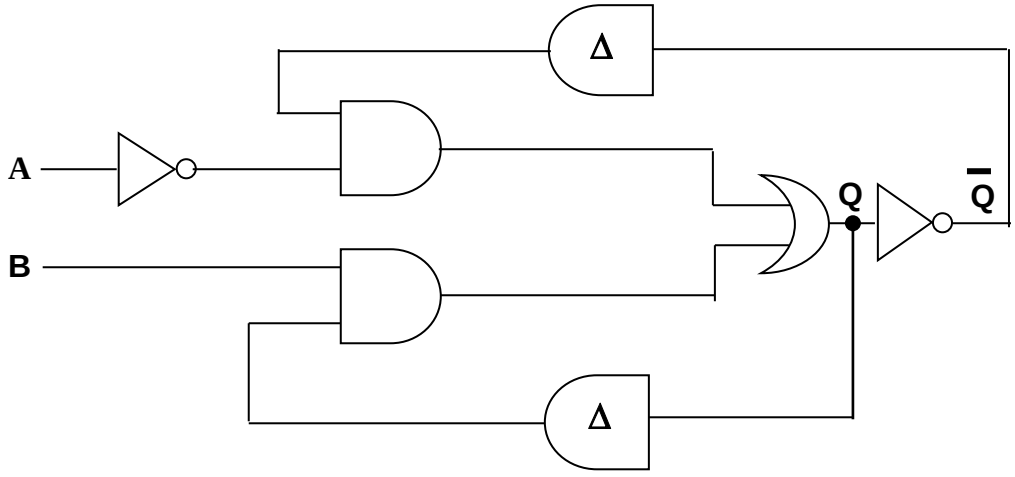
A	**B**	**Q0**	**Q (n + Δt)**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

b) Equação de estado do sistema

AB	00	01	11	10
Q 0	0	1	1	0
1	1	1	0	0

$Q(n + \Delta t) = \Delta(BQ + A\overline{Q})$ = Equação de estado.

Implementação do circuito lógico que define o comportamento dinâmico.



No exercício anterior é fácil perceber que o sistema possui realimentação da saída para a entrada, o qual retém o estado.

1 - INTRODUÇÃO AOS F/Fs.

Os F/Fs são circuitos realizados com portas lógicas interligadas através de uma realimentação positiva. A saída responde de acordo com as entradas e após o sinal de relógio que valida as entradas para o F/F. Cada F/F tem a sua aplicação, tais como, contadores, registradores, memória etc... Os tipos de F/Fs são: Tipo T (Toggle), Tipo RS (Set e Reset), Tipo JK, Tipo D e outros.

2. Latch primitivo SC

Este é um F/F conhecido como Set e Reset ou Set e Clear (Limpa), utilizado como memória (latch). A lógica deste F/F é descrita por uma tabela da verdade, cujas entradas são R e S ou S e C mais o seu estado interno e cuja saída é o instante futuro cuja combinação entre R, S e Qn.

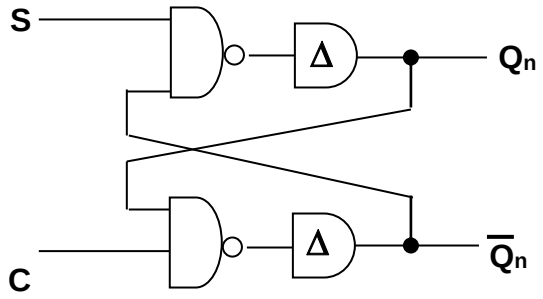


Tabela da verdade latch-SC

S	C	Qn+1
0	0	P
0	1	1
1	0	0
1	1	Qn

Q_{n+1} -> Estado Futuro
Q_n -> Estado Atual
P -> Proibido

A tabela expandida com todos estados presentes e futuros e a tabela de excitação utilizando as entradas R e S e o estado interno Qn.

S	C	Qn	Qn+1
0	0	0	P
0	0	1	P
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

S	C	$Q_n \rightarrow Q_{n+1}$
1	X	$0 \rightarrow 0$
0	1	$0 \rightarrow 1$
1	0	$1 \rightarrow 0$
X	1	$1 \rightarrow 1$

A equação de estado do F/F Qn é conforme a tabela a seguir.

SC	00	01	11	10
Qn 0	X	1	0	0
1	X	1	1	0

A expressão da equação de estado na forma disjuntiva de $Q_{n+1} = \Delta [S' + CQn)]$
A expressão na forma conjuntiva de $Q_{n+1} = \Delta [C . (S' + Q_n)]$

2.1 Flip-Flop RS

Este é um F/F conhecido como Set e Reset ou Set e Clear (Limpa), utilizado como memória (latch). A lógica deste F/F é descrita por uma tabela da verdade, cujas entradas são R e S ou S e C mais o seu estado interno e cuja saída é o instante futuro cuja combinação entre R, S e Q_n.

R	S	Qn
0	0	Q_n
0	1	1
1	0	0
1	1	P

Q_{n+1} -> Estado Futuro
Q_n - > Estado Atual
P -> Proibido
A tabela expandida com todos estados presentes e futuros da tabela da verdade é apresentada utilizando as entradas R e S e o estado interno Q_n.

R	S	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	P
1	1	1	P

A equação de estado do F/F Qn é conforme a tabela a seguir.

RS	00	01	11	10
Qn 0	0	1	X	0
1	1	1	X	0

A expressão da equação de estado na forma disjuntiva de $Q_{n+1} = \Delta [S + R'Q_n)]$
A expressão na forma conjuntiva de $Q_{n+1} = \Delta [R'.(S + Q_n)]$
Transformando a expressão de Q_n na forma conjuntiva para ser implementada com portas NOR, teremos:

$Q_{n+1} = S + RQ_n$
$Q_{n+1} = R + (S + Q_n)$

A equação de estado de Qn é conforme a seguir.

RS	00	01	11	10
Qn 0	1	0	X	1
1	0	0	X	1

$Q_{n+1} = \Delta[R + S'Q_n']$

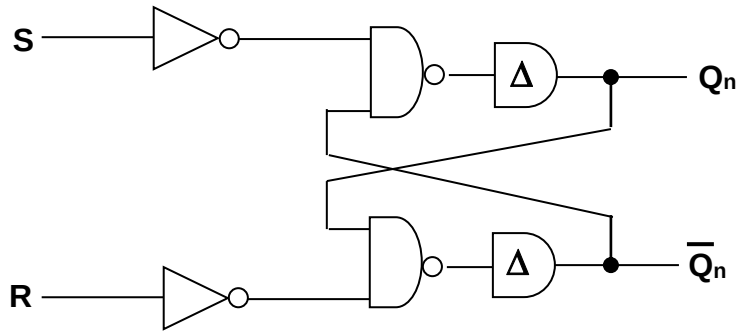
2.1.1 Implementação do circuito Flip-Flop tipo RS usando a memória latch.

Q_n \ RS	00	01	11	10
0	X	1	P	X
1	1	1	P	0

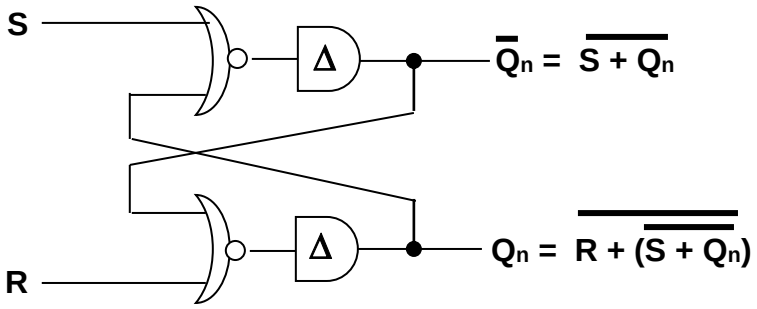
$C = \overline{R}$

Q_n \ RS	00	01	11	10
0	1	0	P	1
1	X	X	P	1

$S = \overline{S}$



A figura mostra a configuração do circuito de um Flip-Flop RS, implementada a partir de portas lógicas do tipo NOR.



NASCIMENTOS DOS FLIP-FLOPS – IMPLEMENTAÇÕES

a) Implementação de um F/F tipo D a partir do Latch primitivo

Inicialmente partimos da tabela de transição do LATCH primitivo e a tabela de estados do F/F a ser implementado.

a.1 LATCH SC

S	C	$Q_n \rightarrow Q_{n+1}$
1	X	0 → 0
0	1	0 → 1
1	0	1 → 0
X	1	1 → 1

a.2 FF tipo D

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

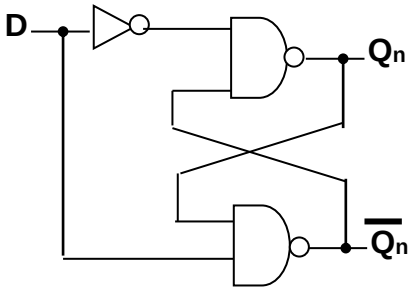
A lógica a ser implementada, deve-se proceder da seguinte forma para cada célula do mapa de "Karnaugh", existe a mesma linha na tabela de estados do F/F desejado. Encontrar a linha e verificar qual o estado futuro. Daí ir para a tabela de transição do "latch" e pegar a transição correspondente. Em seguida preencher os mapas de acordo com a transição correspondente no LATCH.

Q_n \ D	0	1
0	1	0
1	1	X

Q_n \ D	0	1
0	X	1
1	0	1

$S = \overline{D}$ $C = D$

A implementação do F/F usando o bloco primitivo latch, temos:



IMPLEMENTAÇÃO F/F TIPO D SÍNCRONO

A implementação do F/F tipo D síncrono, é realizada pela tabela de estados com o clock.
Tabela de estados do F/F tipo D, síncrono, fica:

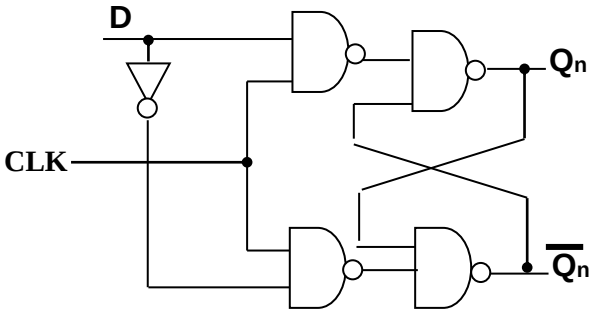
a) Sensível à borda positiva (subida)

D	CLK	Q_{n+1}
0	↑	0
1	↑	1

b) Sensível à borda negativa (descida)

D	CLK	Q_{n+1}
0	↓	0
1	↓	1

A implementação do FF D síncrono é mostrada a seguir:
O circuito FF D síncrono borda de subida fica:



c) Implementação do F/F JK, a partir do LATCH primitivo.

Inicialmente partimos da tabela de transição do LATCH primitivo e a tabela de estados do F/F a ser implementado.

c.1 LATCH SC

S	C	$Q_n \rightarrow Q_{n+1}$
1	X	$0 \rightarrow 0$
0	1	$0 \rightarrow 1$
1	0	$1 \rightarrow 0$
X	1	$1 \rightarrow 1$

c.2 FF Tipo JK

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

A lógica a ser implementada, deve-se proceder da seguinte forma para cada célula do mapa de "Karnaugh", existe a mesma linha na tabela de estados do F/F desejado. Encontrar a linha e verificar qual o estado futuro. Daí ir para a tabela de transição do "latch" e pegar a transição correspondente. Em seguida preencher os mapas de acordo com a transição correspondente no "LATCH".

Q_n \ JK	00	01	11	10
0	1	1	0	0
1	X	1	1	X

$S = \overline{J} + Qn$

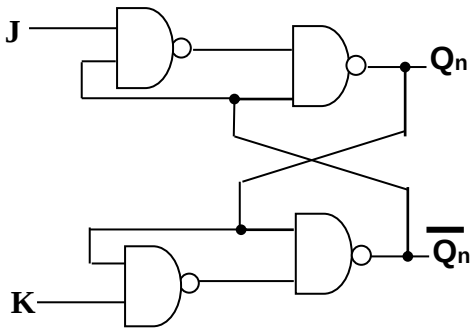
Q_n \ JK	00	01	11	10
0	X	X	1	1
1	1	0	0	1

$C = \overline{K} + \overline{Q}n$

Em termos de blocos NE, fica:

$S = \overline{J\overline{Q}n}$ e $C = \overline{KQn}$

A implementação do F/F JK usando o bloco primitivo latch, temos:



d. IMPLEMENTAÇÃO F/F TIPO JK SÍNCRONO

A implementação do F/F tipo JK síncrono, é realizada pela tabela de estados com o clock.
Tabela de estados do F/F tipo JK, síncrono, fica:

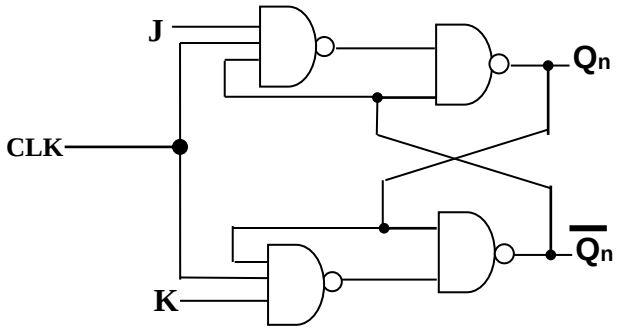
d.1) Sensível à borda positiva (subida)

J	K	CLK	Q_{n+1}
0	0	↑	Qn
0	1	↑	0
1	0	↑	1
1	1	↑	Qn’

d.2) Sensível à borda negativa (descida)

J	K	CLK	Q_{n+1}
0	0	↓	Qn
0	1	↓	0
1	0	↓	1
1	1	↓	Qn’

A implementação do F/F JK com clock, fica:



c) Implementação do F/F T, a partir do LATCH primitivo.
Inicialmente partimos da tabela de transição do LATCH primitivo e a tabela de estados do F/F a ser implementado.

c.1 LATCH

S	C	$Q_n \rightarrow Q_{n+1}$
1	X	$0 \rightarrow 0$
0	1	$0 \rightarrow 1$
1	0	$1 \rightarrow 0$
X	1	$1 \rightarrow 1$

c.2 FF Tipo T

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

A lógica a ser implementada, deve-se proceder da seguinte forma para cada célula do mapa de “Karnaugh”, existe a mesma linha na tabela de estados do F/F desejado. Encontrar a linha e verificar qual o estado futuro. Daí ir para a tabela de transição do “latch” e pegar a transição correspondente. Em seguida preencher os mapas de acordo com a transição correspondente no “LATCH”.

Q_n \ T	0	1
0	1	0
1	X	1

Q_n \ T	0	1
0	X	1
1	1	0

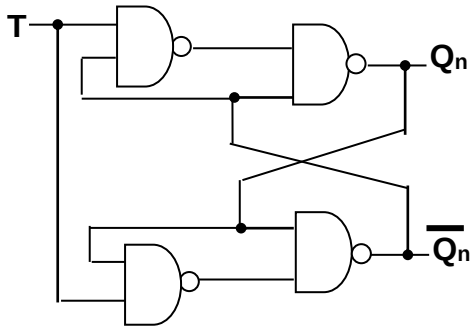
$S = \overline{T} + Qn$

$C = \overline{T} + \overline{Q}n$

Em termos de blocos NE, fica:

$S = \overline{T\overline{Q}n}$ e $C = \overline{TQn}$

A implementação do F/F usando o bloco primitivo latch, temos:



IMPLEMENTAÇÃO F/F TIPO T SÍNCRONO

A implementação do F/F tipo T síncrono, é realizada pela tabela de estados com o clock.
Tabela de estados do F/F tipo T, síncrono, fica:

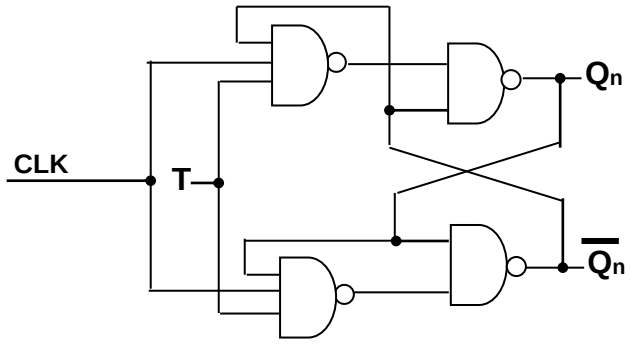
a) Sensível à borda positiva (subida)

T	CLK	Q_{n+1}
0	↑	Qn
1	↑	Qn'

b) Sensível à borda negativa (descida)

T	CLK	Q_{n+1}
0	↓	Q_n
1	↓	Qn'

A implementação do F/F T borda positiva com clock, fica:



Implementações de outros Flip-Flop a partir do FF tipo JK

A partir do Flip-flop primitivo RS pode-se gerar outros tipos de Flip-flop como o JK que é utilizado em um grande número de aplicações tais como na implementação de contadores, registradores e e pode ser considerado como um F/F primitivo na geração de outros F/Fs, como na emulação de Flip-Flops, como tipo T, tipo D etc... Este F/F possui 2 entradas J e K e não possui condições proibidas. A tabela de estados a seguir mostra a dinâmica de evolução deste tipo de circuito.

J	K	Qn+1
0	0	Qn
0	1	0
1	0	1
1	1	$\overline{Qn}$

A tabela expandida com todos estados presentes e futuros da tabela da verdade é feita utilizando as entradas, J e K e mais o estado interno Qn.

J	**K**	**Q_n**	**Q_{n+1}**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

A equação de estado do F/F Qn é conforme a tabela a seguir.

JK	**00**	**01**	**11**	**10**
Q_n 0	0	0	1	1
1	1	0	0	1

$Qn+1 = \Delta [J\overline{Q}n + \overline{K}Qn]$

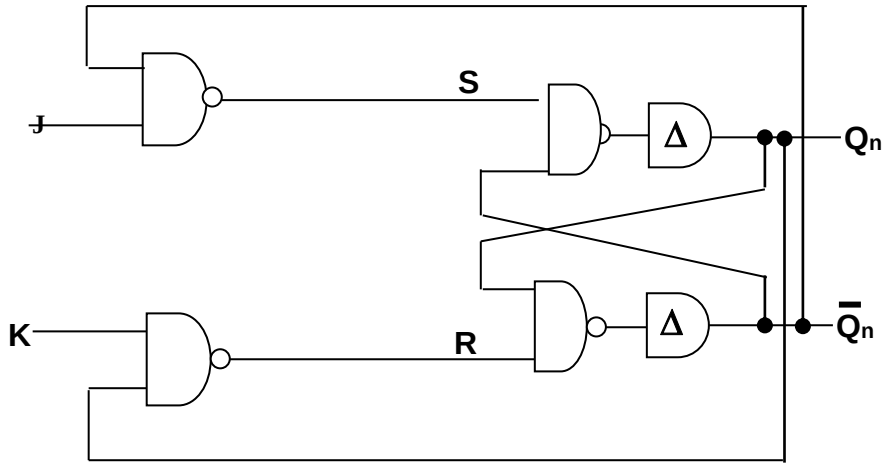
A equação de estado de $\overline{Q}_{n+1}$ é conforme a seguir.

JK	**00**	**01**	**11**	**10**
Q_n 0	0	0	1	1
1	1	0	0	1

$\overline{Qn+1} = \Delta [\overline{J}\overline{Q}n + KQn]$

Implementação do circuito Flip-Flop JK.

A expressão acima pode ser manipulada utilizando o Flip-Flop primitivo RS para gerar o JK. Tomando a expressão da equação de estado do Flip-Flop RS na forma disjuntiva, teremos:
Equação de estado do F/F RS = $Q_{n+1} = \Delta [\overline{S} + RQ_n)]$, fazendo $S = J\overline{Q}_n$ e $R = KQ_n$,



Flip-Flop tipo T

Este Flip-Flop é um dos mais importantes na confecção de contadores, possui uma única entrada T, não possui condições proibidas e pode ser implementado a partir de um F/F primitivo. A tabela a seguir mostra a dinâmica deste F/F.

T	**Q_{n+1}**
0	Q_n
1	$Q_{n'}$

A tabela expandida do Flip-Flop tipo T, resulta em:

T	**Q_n**	**Q_{n+1}**
0	0	0
0	1	1
1	0	1
1	1	0

Pela tabela expandida é possível verificar que toda vez que a entrada T é igual a 1 o F/F troca de estado e quando T=0 o F/F não muda de estado. A equação de estado do F/F é mostrada a seguir.

T/Q	**0**	**1**
	0	1
	1	0

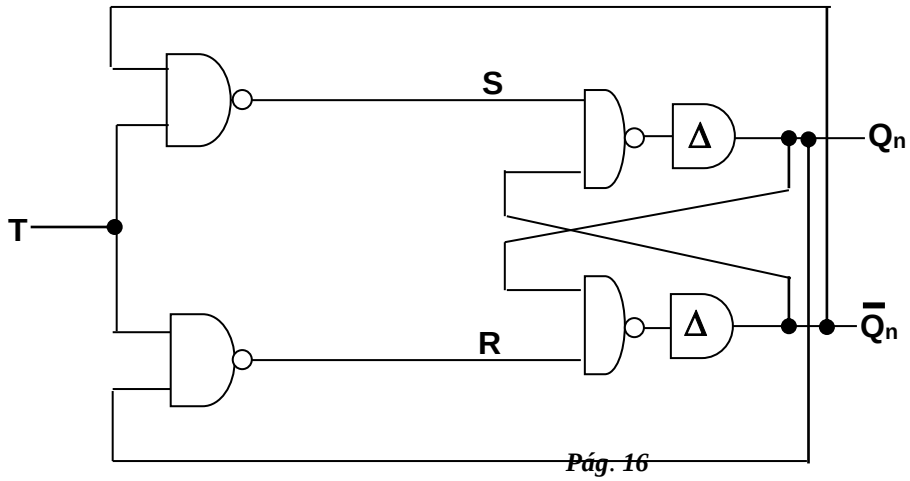
A equação de estado do F/F será:

$Q_{n+1} = T\,\overline{Q_n} + \overline{T}\,Q_n$

Implementação do F/F tipo T

Este F/F pode ser implementado a partir de F/F primitivo. Utilizando a equação de estados do F/F tipo JK e fazendo J=K, estaremos implementando o F/F tipo T. Sendo a equação de estado do F/F JK conforme apresentada a seguir, temos:

T = J e T = K, ou seja, ligando J=K=T o F/F tipo T pode ser implementado, conforme a seguir.



Flip-Flop tipo D (Latch)

O F/F tipo D é conhecido como latch, porque é um F/F transparente, isto quer dizer que a saída copia a informação de entrada. Não possui condições proibidas e é utilizado em aplicações como memória temporária, entrada ou saída de dados, usado na implementação de registradores de deslocamentos, na lógica genérica de dispositivos programáveis como PAL, GAL etc... A tabela da verdade a seguir expressa a dinâmica do F/F.

D	**Q_{n+1}**
0	0
1	1

A tabela do F/F tipo D expandida fica como a seguir:

D	**Q_n**	**Q_{n+1}**
0	0	0
0	1	0
1	0	1
1	1	1

A equação de estado é muito simples porque a saída é igual ao valor da entrada e assim a equação de estado do F/F fica:

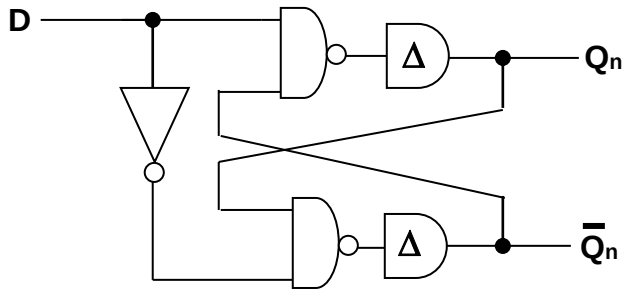
D/Q	**0**	**1**
	0	1
	0	1

A equação de saída é dada por:

$$Q_{n+1} = D$$

Implementação do F/F tipo D

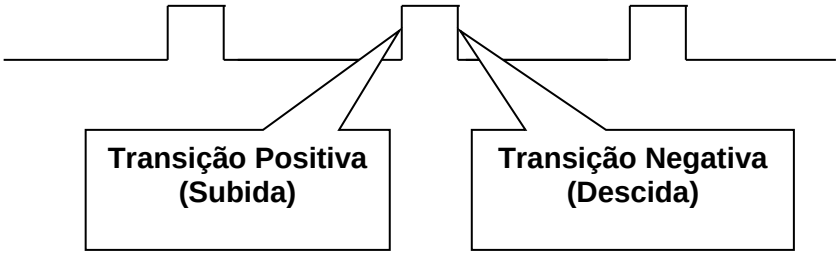
A partir do F/F primitivo RS ou tipo JK, podemos verificar através das equações destes F/Fs, que o tipo D pode ser implementado as a partir destes F/Fs.



Até o presente estudo mostramos os F/Fs sem a introdução do clock para a sincronização. Como estamos estudando os circuitos síncronos cujas entradas e saídas dependem do clock, a seguir introduziremos o conceito dos circuitos síncronos.

Sincronização dos Flip-Flops

Os Flip-Flops podem operar nos modos síncronos ou assíncronos. Para o F/F assíncrono conforme é visto na figura .1 como as entradas podem trocar de estado aleatoriamente, a saída Qn é alterada pela mudança das entradas. Não há intervalos de tempos definindo a operação do circuito. Estes circuitos assíncronos são mais difíceis de projetar e é estudado no capítulo seguinte. Para o F/F síncrono a saída só pode alterar em intervalos de tempos definidos por um relógio, denominado por clock. Este sinal de relógio é geralmente um trem de pulsos e é gerado por um circuito oscilador de frequência. O sinal de clock comanda integralmente o circuito lógico e as saídas do circuito são somente válidas na transição deste sinal, que pode ser positiva ou negativa. Se a mudança de estado de um circuito lógico ocorre com a subida do relógio, o circuito é sensível à transição positiva e caso a mudança de estado do circuito lógico ocorre na descida do sinal de relógio, o circuito lógico é sensível a transição negativa.
Os circuitos lógicos cujas variáveis mudam de estado sob o controle de um relógio, são chamados de circuitos síncronos. Se uma variável pode alterar seu estado em qualquer instante não relacionado às transições do relógio, ela é chamada de variável assíncrona.
Em síntese se uma máquina troca de estado em resposta à um sinal de relógio, esta máquina é classificada como um sistema síncrono. Se uma máquina troca de estado sincronizado apenas com a mudança de estado das variáveis de entrada ao invés do sinal de relógio, esta máquina é classificada como um sistema assíncrono.
A figura mostra um sinal de relógio e as transições positiva e negativa para este sinal.

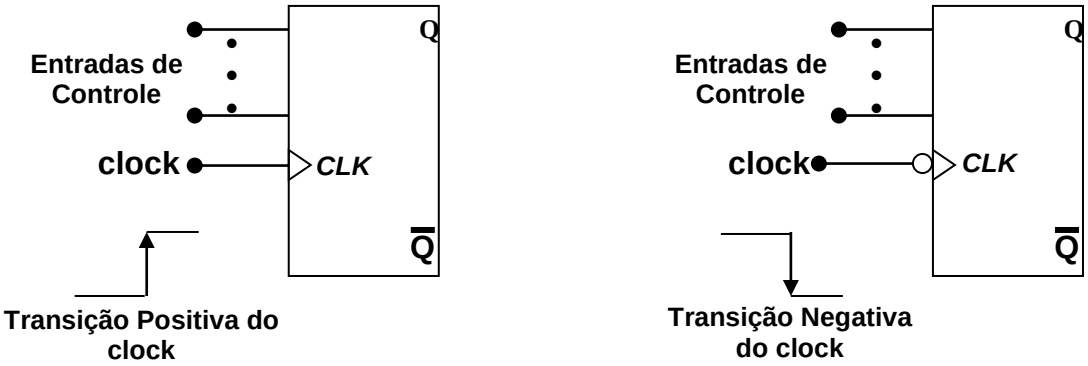


Flip-Flops com sinal de relógio (clock)

Os F/Fs que possuem clock, são conhecidos como F/Fs síncronos e o que determinam a mudança de estado de saída são as seguintes condições:

a) Uma mudança nas entradas do F/F conhecidas como entradas de controle, deixam o F/F pronto para ser alterado o estado da saída;
b) Uma transição ativa do clock é que determina o instante da mudança de estado da saída, ou seja o que realmente efetiva.

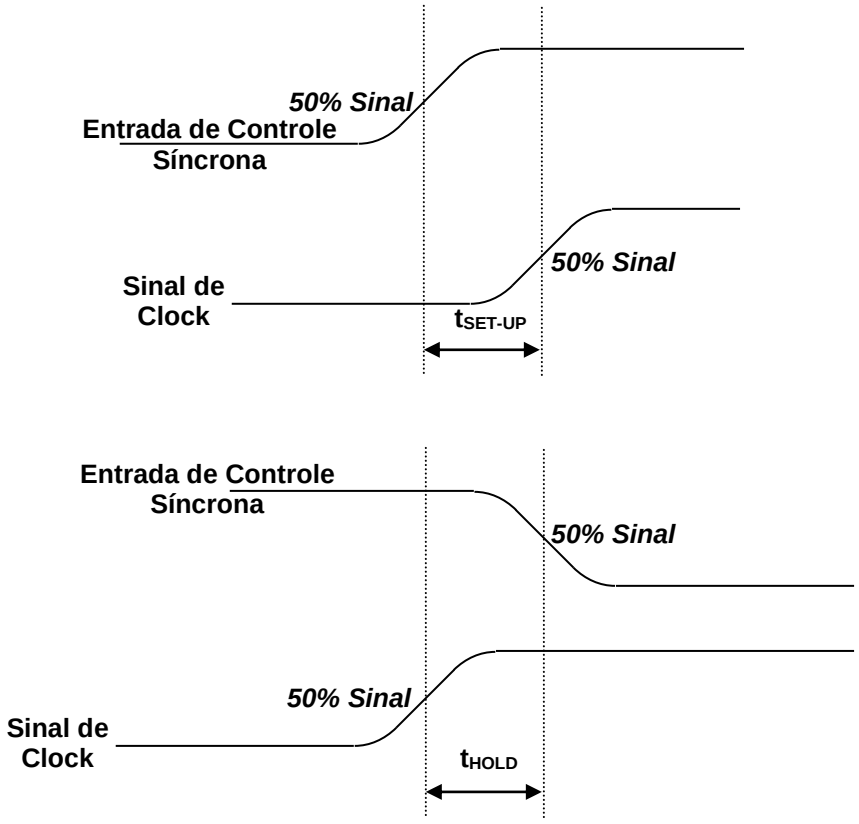
Em resumo, uma mudança das variáveis de entrada não afeta a saída do F/F mas com a chegada do sinal ativo do clock esta condição é efetuada.
A nomenclatura utilizada para o sinal de clock encontrada nos circuitos integrados comerciais é CLK, CK ou CP. A figura mostra uma simbologia no circuito quando este é sensível à transição do clock. Um sinal triangular indica que o circuito é sensível a transição positiva do clock, enquanto o mesmo sinal com uma “bolinha”, a qual indica inversão da transição, o F/F é sensível à transição negativa do clock.



Nos circuitos síncronos existe uma preocupação com os tempos de preparação e de manutenção das entradas com a chegada do clock. As entradas de controle não devem mudar na transição do clock, pois isto pode acarretar em uma saída errada ou gerar uma perturbação indesejada na saída, por exemplo um glitch, característico nos circuitos assíncronos na comutação simultâneas nos sinais de entrada. Para não criar uma situação de incerteza na saída do sinal, isto nos leva a perguntar?

1) Quanto tempo antes de chegar a transição do clock pode-se alterar as entradas de controle para que não haja uma situação de incerteza na saída do F/F e a fim de evitar glitches?
2) Quanto tempo devo ainda manter as entradas de controle estáveis para que se efetive a mudança correta de estado de saída do F/F?

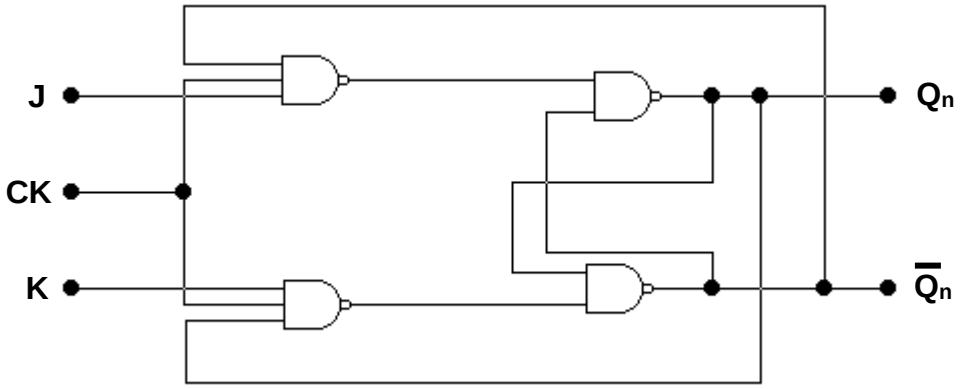
Estas perguntas podem ser respondidas, observando o manual dos circuitos integrados e a família de circuitos integrados, pois é característico de cada família.
Para a família TTL-Standard o tempo de preparação conhecido tempo de set-up, t_S é da ordem 5ns, chegando até 50ns e o tempo de manutenção, conhecido como tempo de hold t_H é da ordem de 0 a 10ns. A medição do t_{SET}-UP e t_{HOLD} são feitos conforme os sinais descritos pelas formas de ondas abaixo, na figura a seguir.



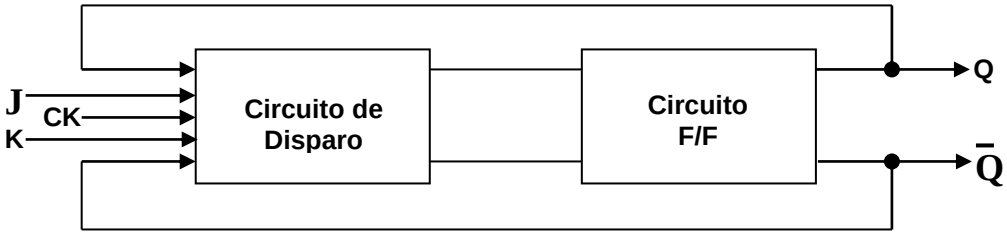
Para que a saída seja confiável é necessário obedecer aos tempos de preparação, preparando antecipadamente as entradas de controle no tempo mínimo de set-up até a chegada da transição do clock e manter as entradas de controle estáveis pelo tempo mínimo de hold após a transição do clock.

DISPARO POR BORDA x DISPARO POR NÍVEL

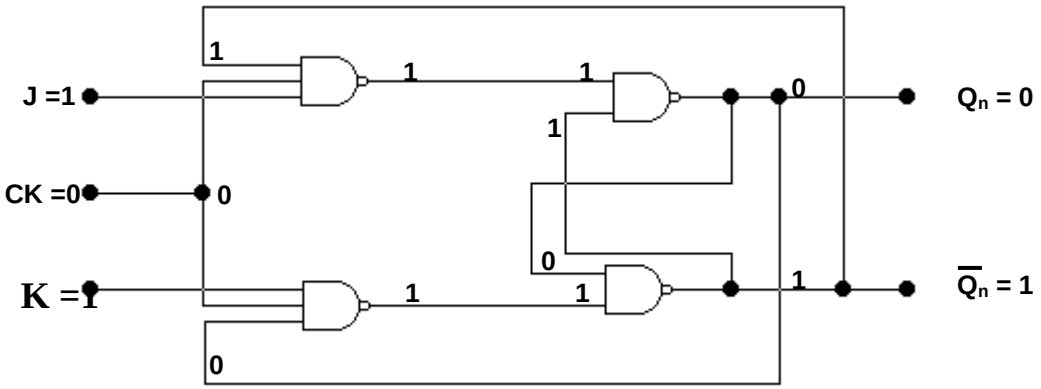
Os F/Fs podem ser sincronizados por um sinal de clock e são projetados para serem sensíveis às transições ou positivas ou negativas do clock ou sincronizados por sinais que alcançam algum nível particular. Quando a mudança de estado lógico do F/F ocorre somente com a borda do sinal de clock é dito que o F/F é sensível a borda do sinal. Fazendo a analogia com sistemas lineares, a borda é um sinal diferenciado onde somente a transição de nível lógico é que determina as mudanças no estado lógico. Quando a mudança de estado lógico do F/F ocorre quando o sinal atinge um determinado nível de tensão é dito que o F/F é sensível a nível de tensão. Nos circuitos síncronos a borda do sinal é identificada pelo triangulo na entrada do sinal do clock. Quando não aparece a identificação o circuito pode ser sensível a nível de tensão. Para o circuito JK com clock, vamos submeter este circuito à variações na largura dos sinais de clock e analisar o resultado da sua saída. A figura .X mostra o circuito do F/F JK sensível a transição positiva do sinal de clock.



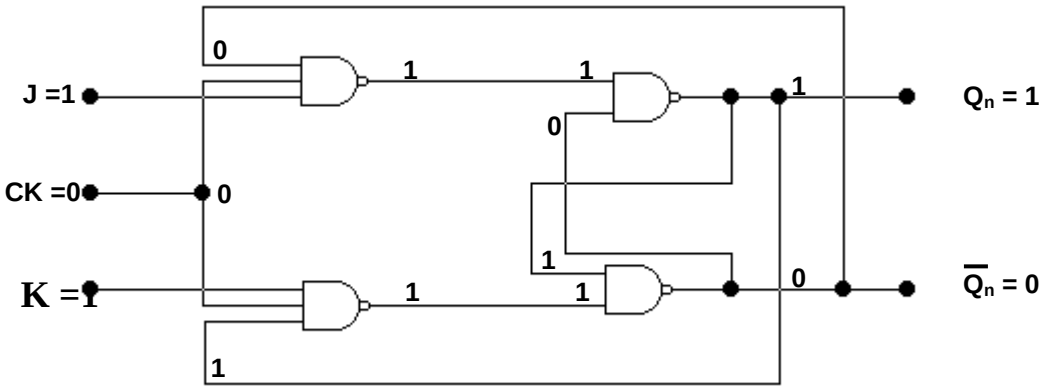
Inicialmente vamos dividir o circuito em 02 partes. As portas NAND referentes às entradas J e K denomina-se de circuito de disparo e a outra parte do circuito continuação do circuito de disparo denomina-se de F/F.



Vamos considerar a fim de análise do circuito, J = K = 1 no circuito de disparo e o estado inicial do F/F em NL0 com Qn = 0. Uma terceira entrada referente ao sinal de clock dá o instante da transição do circuito de disparo e consequentemente do F/F. Inicialmente o sinal de clock se encontra em NL0.



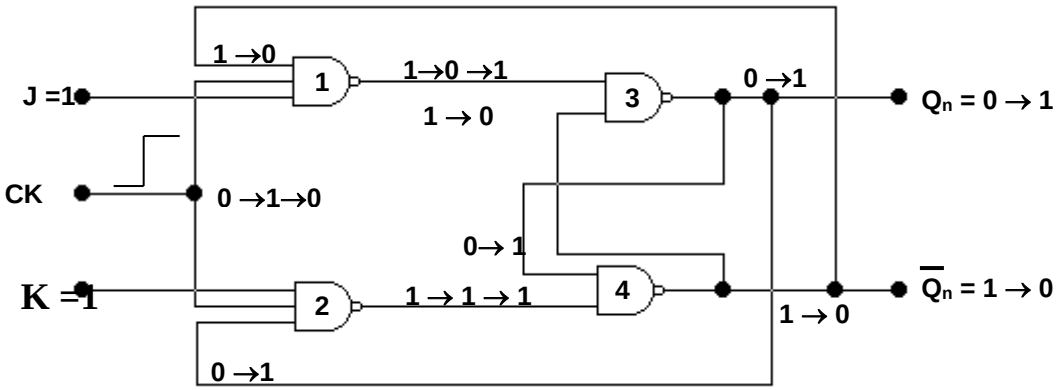
Enquanto o sinal de clock CK permanecer neste estado lógico, a saída do F/F permanece inalterada. As saídas das portas lógicas NANDs do circuito de disparo permanecem inalteradas e em nível lógico 1 (pois como CK = 0 implica, se uma das entradas de uma porta NAND é igual a nível baixo implica na saída da porta em nível alto). Pelo circuito, as entradas das portas NAND do F/F SR recebem às saídas das portas NAND do circuito de disparo, portanto recebem o NL1 destas portas e sendo assim não há alteração no estado lógico do F/F. Isto pode ser verificado assim, como $Q_n = 0$ e há realimentações das saídas do F/F RS para as suas entradas, o nível de Q_n é transmitido para uma das entradas da porta NAND sendo que a outra entrada permanece em NL1 e fazendo a lógica vemos que o F/F mantém Q_n' = 1. Da mesma forma Q_n, é transmitido para uma das entradas da outra porta NAND do F/F RS e a outra entrada permanece em NL1, fazendo a lógica $Q_n = 0$. Quando o sinal de clock CK = 0 não há alteração no estado lógico do F/F. Para confirmar outra condição, supomos que $Q_n = 1$ e /Qn = 0, como CK = 0 as saídas das portas NANDs do circuito de disparo estão em NL1 e assim a lógica dos sinais na entrada de cada porta NAND do F/F RS recebe o sinal de saída das portas NAND de entradas e dos sinais Q_n e Q_n', fazendo a lógica das portas do F/F com Q_n e Q_n' se confirma o estado lógico da saída do F/F RS pois não há alteração permanecendo $Q_n = 1$ e Q_n' = 0.



Voltando a condição de $Q_n = 0$ e Q_n' =1, vamos realizar uma análise dinâmica do circuito com transição do clock de NL0 para NL1 e considerar o instante em que o sinal de clock CK muda de estado. Para a porta NAND ligada a entrada K Porta 2, não há qualquer alteração na sua saída pois uma das suas 03 entradas está ligada à saída $Q_n = 0$ e daí a saída da porta NAND permanece em NL1. Para a outra porta NAND de 03 entradas ligada a entrada J Porta 1, temos as seguintes condições de entrada: uma entrada ligada a J = 1 uma outra entrada ligada a CK = 1 e a terceira entrada que é uma realimentação da saída para a entrada ligada a Q_n' = 1 (condição inicial). Como as 03 entradas estão em nível alto, a saída desta porta NAND 2 vai para NL0, resultado da função lógica. A entrada da porta NAND do F/F RS Porta 3 recebe NL0 e daí resulta saída $Q_n = 1$. Ocorreram 02 tempos de atrasos de propagação dos sinais, referentes à propagação na porta 1 e 2 e após na porta 3 e 4, embora as portas afetadas são somente as portas 1 e 3.

Após os 2 tempos de atrasos o sinal de clock deve ir a zero pois isso garante o estado imposto pelas entradas J e K. Assim a janela do clock deve ser igual a 2 tempos de atrasos. Vamos analisar o período do clock para observar o que ocorre com o F/F caso este tempo de propagação do clock variar:

a) Caso 1: CK = 0 após o 2.o tempo de propagação do clock. Para análise tomamos a seguinte condição J = K = 1 e CK = 0 e Q_n = 1, considere esta situação, onde Q_n ainda não se propagou após os 2 tempos de atrasos, a mudança de Q_n de 0 para 1. Como existe realimentação das saídas para as entradas, tanto das portas 1 e 2 como das portas 3 e 4, a saída Q_n é retornada para a porta 2 e para a porta 4. A porta 2 será somente influenciada pelo CK = 0 mantendo-se após o tempo de propagação em NL1 e a porta 4 sofrerá uma mudança pois as 2 condições de entrada geradas pelas portas 2 e 3 estão NL1 fazendo a saída alterar para NL0. A porta 1 não sofre qualquer alteração pois CK = 0 impõe a saída da porta 1 em NL1. Sabemos que se o circuito de disparo mantiver NL1 a saída fica inalterada assim o F/F passou do estado Q_n = 0 para Q_n =1. A leitura do status do F/F deveria somente ser feita após o 3.o tempo de propagação referente a porta 4.



2) Caso 2: Considere CK ainda em NL1 após o 2.o tempo de propagação. Continuando a análise, mantendo-se as mesmas condições para as variáveis do caso 1, ou seja, J = K = CK =1 e a saída Q_n =1 e é realimentada pela ligação para a entrada da porta NAND do circuito de disparo ligada a entrada K Porta 2 e também realimentada para o próprio F/F para a entrada da porta NAND 4. Considerando que existe um atraso na propagação do sinal através das portas lógicas, após o atraso das saídas tanto da porta NAND Porta 2 como também em seguida da porta NAND Porta 4, são modificados os seus estados lógicos. A saída da porta K Porta 2 vai para NL0 e a saída do F/F RS Porta 4 Q_n' vai também para NL0. Após o 3.o atraso de propagação o ciclo inicia novamente pois o clock permanece em NL1.
Como existe realimentação da saída Q_n' para a entrada da porta de entrada NAND Porta 1, começa a ocorrer uma alteração nesta porta NAND. Ao mesmo tempo ocorre uma modificação na porta 4. Após o 4.o atraso de propagação a saída Q_n' vai para NL1 e a saída da porta 2 também vai para NL1. A porta 1 recebe 2 sinais de NL1 ao mesmo tempo e o 5.o tempo de atraso leva Q_n a NL0. Esta mudança em Qn vai provocar nova troca de saída, ou seja, o circuito vai ficar oscilando em uma freqüência igual ao inverso do atraso de propagação. Esta oscilação na saída do F/F é inconveniente e dura enquanto o sinal CK = 1, no instante em que o CK = 0 o F/F amarra o estado presente, que poderá ser Q_n = 0 ou Q_n =1, ou seja, aquele que era presente no instante da comutação de estado do clock de nível alto para nível baixo. Pode-se concluir que quanto maior for a atividade do clock, maior será o tempo de ocorrência da oscilação.

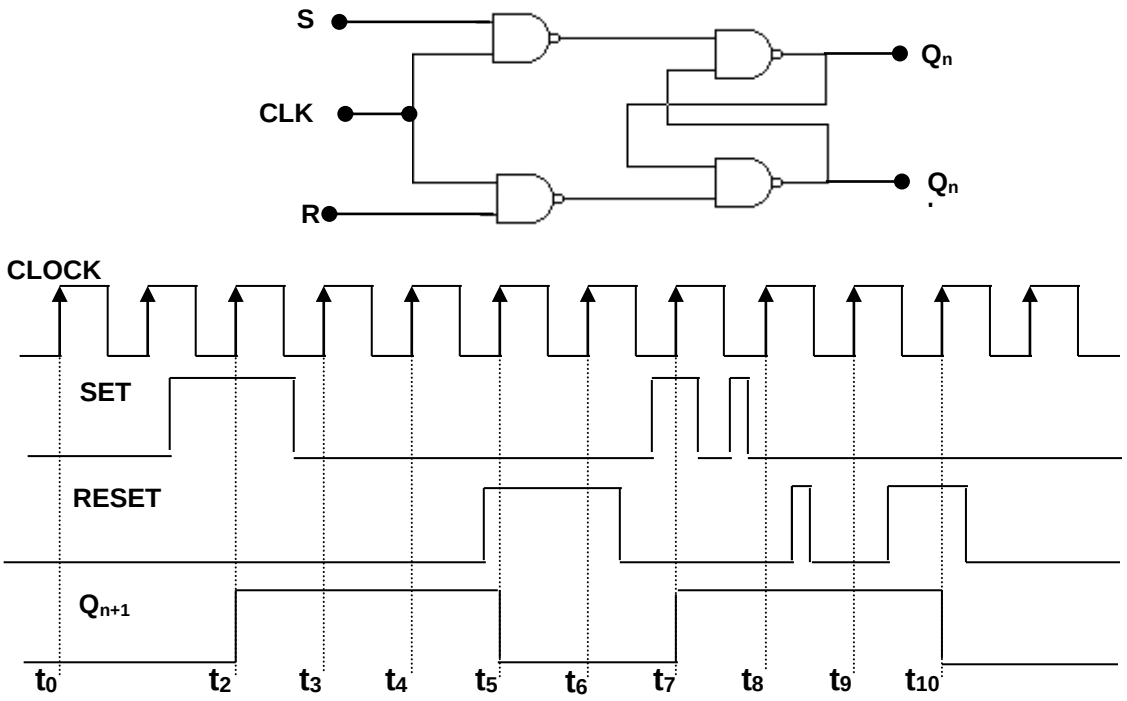
Flip-Flop RS com clock

Para a descrição da dinâmica do F/F sincronizado com o sinal de relógio, a tabela da verdade da figura .X a seguir, mostra a performance do circuito. Pode-se observar que o sinal do clock participa da tabela

da verdade e a cada transição deste existe uma saída correspondente. Para o F/F RS, as entradas de controle devem mudar obedecendo as condições do tempo tS e tH.

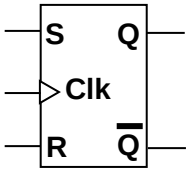
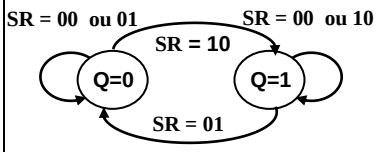
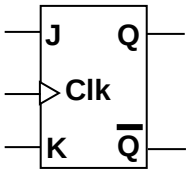
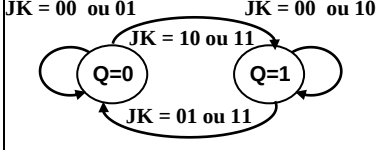
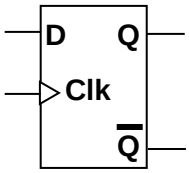
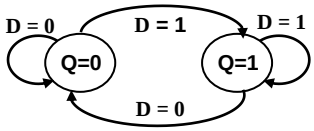
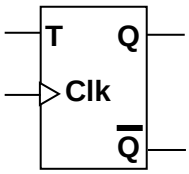
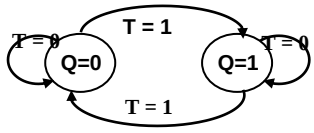
R	S	CLK	Q_{n+1}
0	0	↑	Q_n (não muda)
0	1	↑	1
1	0	↑	0
1	1	↑	Proibido

A nomenclatura do clock ↑ significa que as mudanças na saída do F/F ocorrem na transição positiva do clock.O circuito da figura .X mostra o F/F RS sincronizado com o clock. A seguir vamos a análise do circuito.



Inicialmente no instante t0, o estado lógico do F/F é igual $Q_n = 0$, o circuito é sensível a transição positiva do clock, a as entradas de controle R = S = 0, com a subida do clock o estado futuro do F/F a saída $Q_{n+1} = 0$, mantendo-se o valor atual de Q_n. A mudança na entrada ocorre entre os instantes t1 e t2, com S =1 e R = 0, após a subida do clock em t2, a saída do F/F vai para o estado lógico Qn = 1. Entre os instantes t2 e t3, a entrada S = 0 e R = 0, o F/F mantém a saída atual, ou seja, $Q_n = 1$. Até o presente momento, observamos que o F/F somente é afetado a sua saída para $Q_n = 1$, quando S =1 e R =0 e quando a entrada S retorna a 0 ou seja S = 0 e R = 0, a saída mantém o valor $Q_n = 1$, daí o F/F é conhecido como latch, memória temporária ou buffer temporário. Dando continuidade na análise do circuito, entre os instantes t4 e t5 ocorre uma mudança de estado na entrada R = 1 e S = 0 e após a transição positiva do clock, a saída do F/F imediatamente vai para o estado lógico $Q_n = 0$. Desta forma a entrada R = 1 provocou um Reset no estado lógico do F/F, mandando a saída do F/F para Qn = 0. Entre os instantes t6 e t7, a entrada S = 1 e R = 0, o F/F após a transição do clock, manda a saída do F/F, para o estado lógico $Q_n = 1$, pois S=1, impõe a saída $Q_n = 1$. Entre os instantes t7 e t8 ocorre uma troca de estado completa na entrada S, porém este ciclo completo não afetaria a saída do F/F pois a transição positiva do clock ocorre depois deste ciclo, daí a saída Q_n se mantém com o valor atual $Q_n = 1$. Mesmo processo ocorre com a entrada R, nos instantes t8 e t9, onde uma troca completa na entrada R, não afeta o F/F pois a transição positiva do clock só ocorre depois do ciclo, daí mantendo o estado atual da saída do F/F, ou seja, $Q_n = 1$. Uma última variação na entrada ocorre entre os instantes t9 e t10, onde R =1 até a transição positiva do clock, daí a saída do F/F é alterada para $Q_n = 0$, ou seja, a entrada R =1 Reset para $Q_n = 0$ a saída do F/F RS.Pode-se concluir que quanto maior for a atividade do clock, maior será o tempo de ocorrência da oscilação.

II – RESUMO - TABELA DE ESTADOS DOS F/Fs SR, JK, D e T

Nome	Símbolo F/F	Tabela da Verdade	Diagrama de Estados Equações de Estados	Tabela de Excitação
SR		S R | Q_{n+1} 0 0 | Q_n 0 1 | 0 1 0 | 1 1 1 | NA	 $Q_{next} = S + R'Q$	S R | $Q_n \rightarrow Q_{n+1}$ 0 X | 0 → 0 1 0 | 0 → 1 0 1 | 1 → 0 X 0 | 1 → 1
JK		J K | Q_{n+1} 0 0 | Q_n 0 1 | 0 1 0 | 1 1 1 | $\overline{Q_n}$	 $Q_{next} = K'Q + JQ'$	J K | $Q_n \rightarrow Q_{n+1}$ 0 X | 0 → 0 1 X | 0 → 1 X 1 | 1 → 0 X 0 | 1 → 1
D		D | Q_{n+1} 0 | 0 1 | 1	 $Q_{next} = D$	D | $Q_n \rightarrow Q_{n+1}$ 0 | 0 → 0 1 | 0 → 1 0 | 1 → 0 1 | 1 → 1
T		T | Q_{n+1} 0 | Q_n 1 | $\overline{Q_n}$	 $Q_{next} = T \oplus Q$	T | $Q_n \rightarrow Q_{n+1}$ 0 | 0 → 0 1 | 0 → 1 1 | 1 → 0 0 | 1 → 1

SÍNTESE DE SISTEMAS SEQÜENCIAIS

Os sistemas sequenciais são caracterizados pelo elemento de memória que pode ser de 1 a n elementos, dependendo do número de estados internos. A saída do sistema pode ser caracterizada pelo tipo de modelo de síntese de sistemas sequenciais: Modelo de Mealy ou de Moore.

Exemplo: Um sistema sequencial é definido pela tabela da verdade a seguir. Pede-se:

a) Implementação do sistema usando F/F do tipo JK.

b) Circuito sequencial

X	Y	Z	a	b
0	0	0	1	0
0	0	1	0	0
0	1	0	0	1
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	0

XY	00	01	11	10
Za-00	1	X	X	1
01	X	X	X	X
11	X	X	X	X
10	X	1	1	1

J = 1

XY	00	01	11	10
Za-0	X	0	0	X
01	0	1	1	0
11	1	0	0	0
10	0	X	X	X

K = X'Y'Za + YZ'a

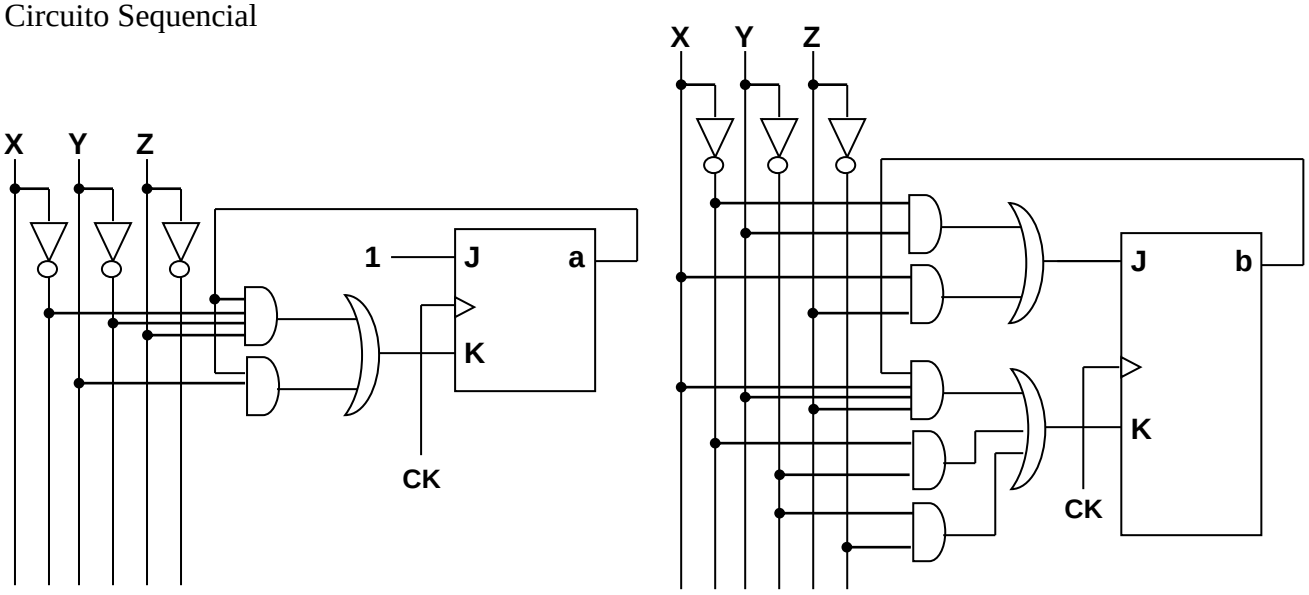
XY	00	01	11	10
Zb-00	0	1	0	0
01	x	x	x	x
11	x	x	x	x
10	0	1	x	1

J = X'Y + XZ

XY	00	01	11	10
Zb-00	x	x	x	x
01	1	0	0	1
11	1	0	1	0
10	x	x	0	x

K = Y'Z' + X'Y' + XYZb

Circuito Sequencial



Módulo 02 - MÁQUINAS DE ESTADOS FINITOS – F.S.M.

1. Introdução: As máquinas de estados finitos F.S.Ms. são sistemas preparados para a solução de problemas tipicamente sequenciais. Um exemplo de um sistema sequencial é um contador de estados o qual percorre sucessivamente uma malha de estados que podem ser sucessivos ou não e que repetem essa sequência sempre que forem excitados. A resposta do sistema digital quando excitado por uma fonte externa é percorrer uma determinada malha de estados. A máquina de estados num sentido amplo serve para implementar sistemas sequenciais e existem muitas formas diferentes de construções. A máquina de estados finitos pode ser considerada como uma automata. Devido a sua estrutura de implementação pode ser considerada como uma automata programável. As aplicações com as máquinas de estados incluem projeto de automação eletrônica, projeto de protocolos de comunicações, em área da inteligência artificial com sistemas neurológicos e na área linguística descreve a gramática da linguagem natural. O exemplo a seguir mostra um diagrama de estado de uma máquina de estado a qual descreve o comportamento de um flip-flop do tipo T.

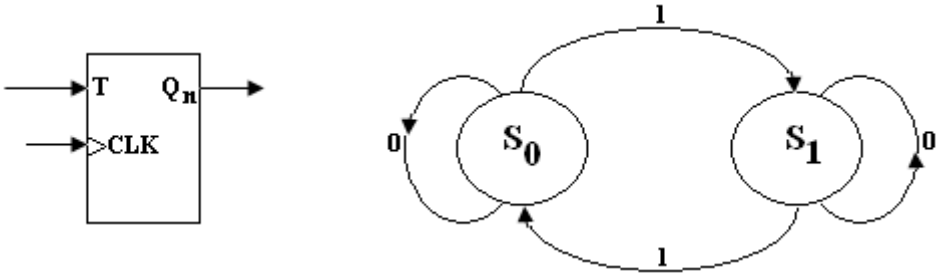


Figura: Subsistema digital flip- flop do tipo T em bloco e descrito por uma Máquina de estados finitos.

2. SISTEMAS SEQÜENCIAIS

Um sistema é construído para a implementação das máquinas de estados finitos e a base do sistema sequencial é que a mudança de estado da máquina depende não somente das entradas presentes como também do estado atual e se um sinal de relógio determina o intervalo de tempo dessa mudança então é chamado de sistema sequencial síncrono e caso contrário de sistema sequencial assíncrono. Os sistemas que operam no modo sequencial com o relógio são os sistemas síncronos onde a resposta de saída só se modifica ao comando de um sincronismo ou os sistemas operam sem o relógio, no modo assíncrono, onde a resposta se modifica conforme a chegada dos sinais nas entradas do sistema e do tempo de propagação destes por cada bloco lógico. Um sistema sequencial é uma máquina de estados cuja saída depende do estado atual do sistema e das entradas externas. Os estados atuais definidos como estados internos são armazenados na memória do sistema e é conhecido como a memória de estado. O circuito evolui de estado lógico para o próximo estado ou estado futuro pela combinação da entrada externa com o estado lógico da memória. Este novo estado passa a ser o estado atual e é armazenado na memória do sistema, assim alterando o estado lógico da memória. Para uma sequência de eventos, o sistema executa uma sequência de estados. Como a memória é finita o número de diferentes estados que o circuito pode percorrer será um número finito de estados.

Definição Formal

Uma máquina de Moore pode ser definida como uma tupla-6 (S, S0, Σ, Λ, T, G) consistindo do seguinte:

- conjunto de estados finitos (S);
- estado de partida (também chamado de estado inicial) S0 o qual é um elemento de (S);

- conjunto finito chamado de alfabeto de entrada (Σ);
- conjunto finito chamado de alfabeto de saída (Λ);
- função transição (T : S × Σ → S) mapeando o estado e o alfabeto de entrada para o próximo estado;
- função de saída (G : S → Λ) mapeando cada estado para o alfabeto de saída.

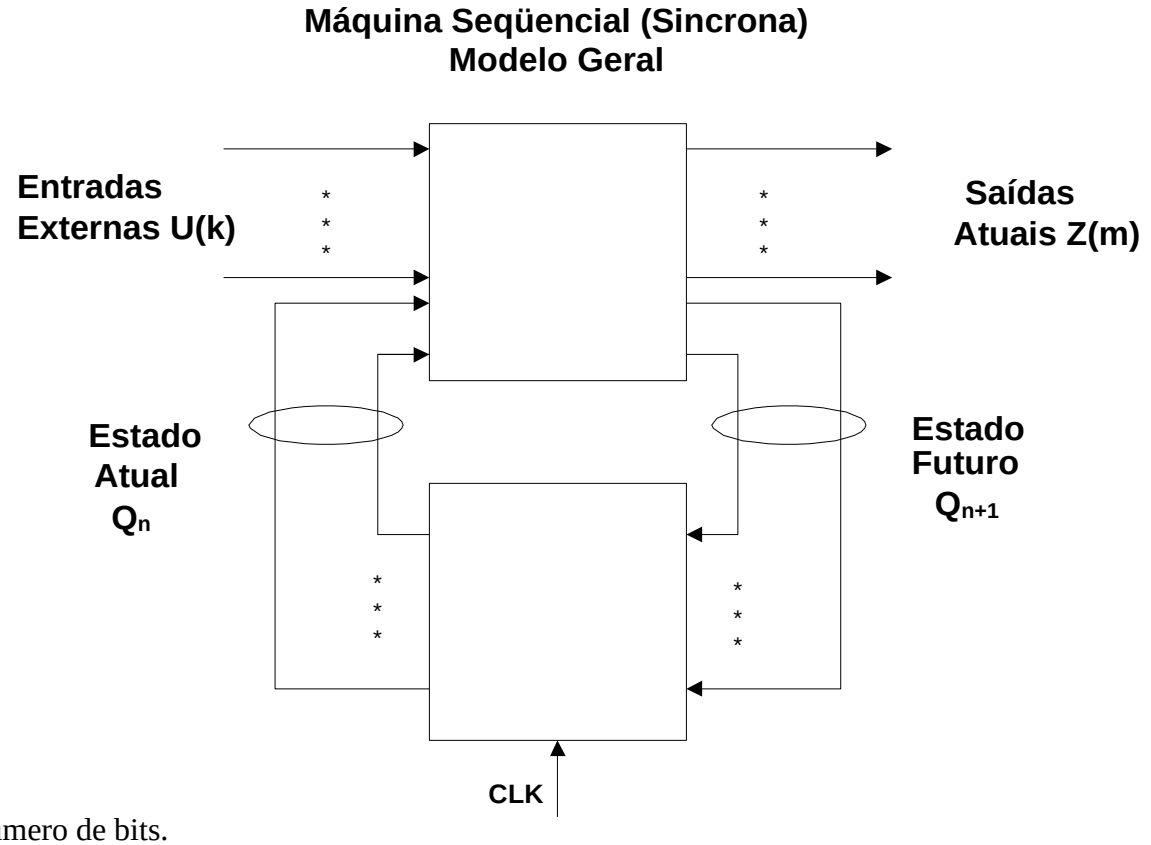
O número de estados na máquina de Moore será maior ou igual ao número de estados na correspondente máquina de Mealy. O fato é que cada transição na máquina de Mealy pode ser associada com um correspondente estado adicional mapeando a transição para uma saída única.

Definição Formal

Uma máquina de Mealy é uma 6-tupla, (S, S0, Σ, Λ, T, G), consistindo do seguinte:

- de um conjunto de estados finitos (S);
- de um estado de partida (também chamado de estado inicial) S0 o qual é um elemento de (S);
- de um conjunto finito chamado de alfabeto de entrada (Σ);
- de um conjunto finito chamado de alfabeto de saída (Λ);
- de uma função transição (T : S × Σ → S) mapeando o estado e o alfabeto de entrada para o próximo estado;
- de uma função de saída (G : S × Σ → Λ) mapeando cada estado e o alfabeto de entrada para o alfabeto de saída.

MODELO GERAL



n = número de bits.
k = número de entradas externas.
m = número de saídas.

A associação dos elementos de memórias chamados de flip-flops permite a implementação de outros subsistemas digitais como contador e registrador.

2.1 Exemplos de subsistemas digitais sequenciais.

- Contadores;
- Registradores;
- Células de memórias;
- Outros.

a) Classificação de alguns subsistemas digitais em sequenciais ou combinacionais.

Sistemas	Tipos
Portas Lógicas	combinatória
Codificador	combinatória
Demultiplex	combinatória
Registrador	sequencial
Flip-flop	Sequencial
Decodificador	combinatória
Somador/Subtrator	combinatória
Contador	sequencial
Memórias Estáticas	sequencial
Unidade Lógica Aritmética	combinatória

3. Modos de operações de um sistema sequencial.

Os modos de operações dos sistemas sequenciais são: do tipo síncrono ou assíncrono.

3.1 Modo Sequencial Assíncrono

São circuitos sequenciais na qual a evolução de estado não depende de um sinal de sincronismo, mas depende somente das entradas e dos estados internos.

3.2 Modo Sequencial Síncrono

São circuitos sequenciais na qual a evolução de estado depende de um sinal de sincronismo, conhecido como relógio (clock).

3.3 Relógio

É um sinal periódico de determinada amplitude e frequência. Pode ser uma onda quadrada ou retangular, mas um sinal que apresenta na amplitude somente tensões de nível zero e tensões de nível um.

4. MODELOS DE DESCRIÇÕES DE SISTEMAS

Para os modos de operação síncronos e assíncronos, dois modelos podem descrever o comportamento dinâmico dos sistemas. São eles: modelo de Mealy e de Moore. O presente estudo vai abordar somente os sistemas sequenciais síncronos. Os modelos de Mealy e de Moore são modelos muito parecidos definidos como:

a) Modelo de Moore.

O estado futuro de depende do estado atual e das entradas externas e a saída depende do estado atual.

b) Modelo de Mealy.

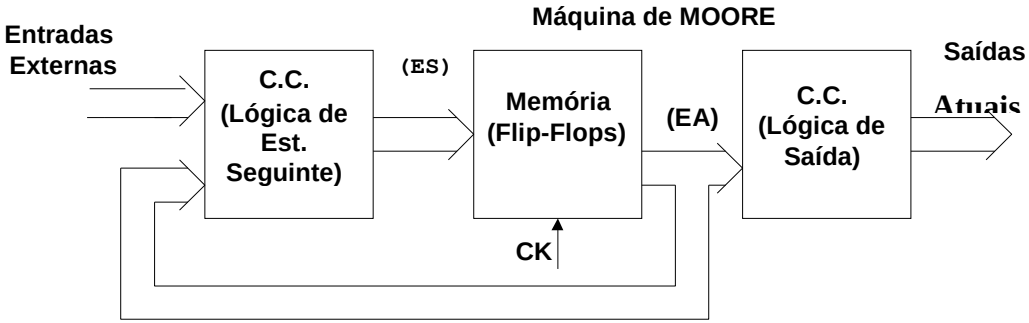
O estado futuro e a saída dependem do estado atual e das entradas externas.

A seguir são apresentados os dois modelos de descrições de sistemas sequenciais de Moore e de Mealy.

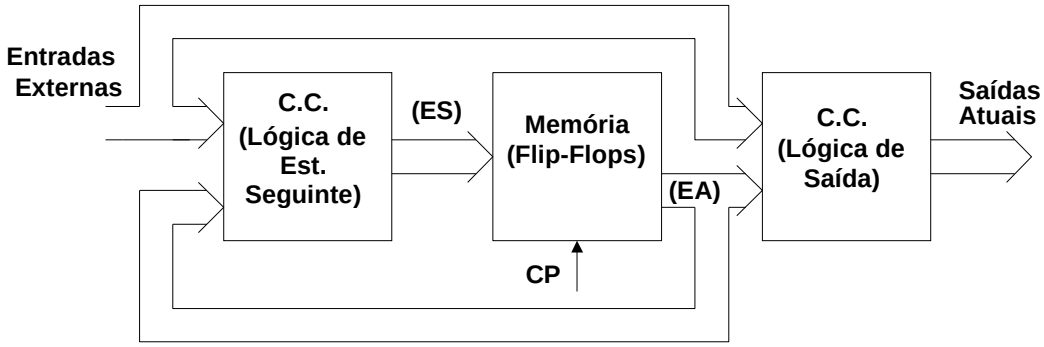
SISTEMAS SEQUENCIAIS SÍNCRONOS

MODELOS DE MOORE E MEALY (Caracterização)

Modelo de Moore: Circuito no qual as saídas são funções diretas dos estados.



Modelo de Mealy: Circuito no qual as saídas são funções dos estados e das entradas.



Nota: Em geral, os circuitos de Moore apresentam uma maior simplicidade na geração das saídas, enquanto os circuitos de Mealy conduzem a um menor número de estados e à eventual redução do número de FFs necessários.

DIAGRAMA DE ESTADOS

O diagrama de estados é uma ferramenta de análise de circuitos sequenciais que permite descrever numa forma gráfica, o comportamento dinâmico de uma máquina de estados. É, portanto um grafo em forma de diagrama, muito útil e prático no projeto de máquinas cuja descrição do funcionamento é produto de um conjunto de especificações bem elaboradas. Através de alguns símbolos conforme figura 1 o diagrama de estados é apresentado.

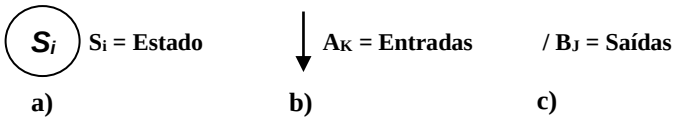
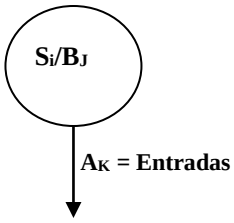


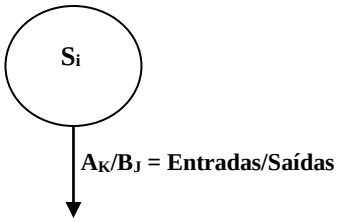
Figura: Símbolos do diagrama de estado.

O símbolo circular visto no item a) representa o estado, internamente ao círculo vem o nome do estado e a identificação do estado. Os arcos indicam o caminho de entrada e de saída. Podem existir múltiplos arcos e, portanto, muitos caminhos de entrada e de saída. O arco no item b) carrega as variáveis de entrada identificadas, mas pode representar uma expressão booleana. O símbolo no item c) representa a saída e internamente é inscrito o nome da variável de saída.

a) Modelo de Moore



b) Modelo de Mealy



Exemplo: Implementação da F.S.M. pelos modelos de Mealy e de Moore.

a) Diagrama de estados por Moore

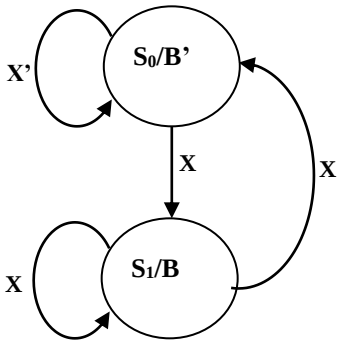


Figura: Modelo de estado Moore.

b) Diagrama de estados por Mealy

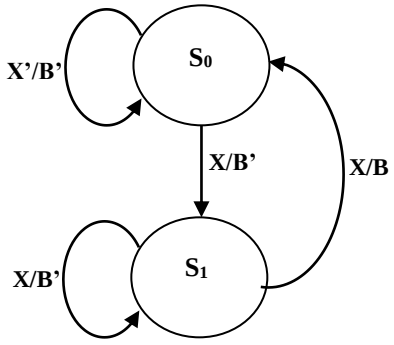
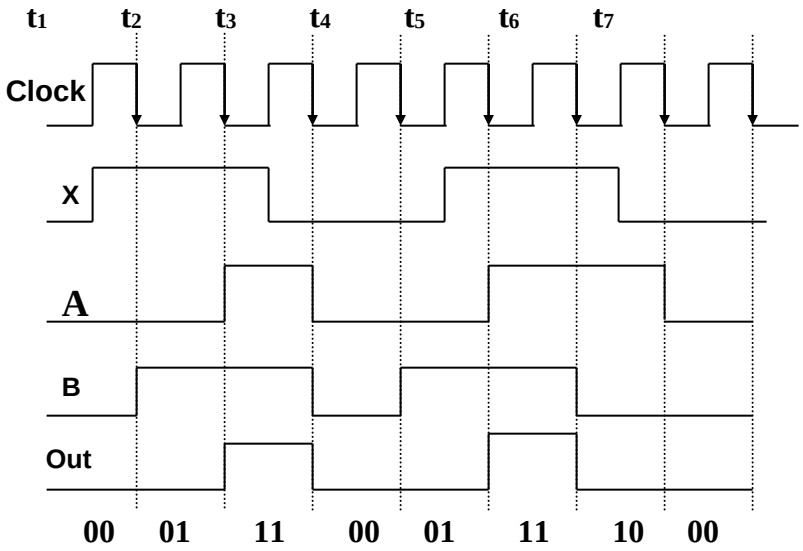


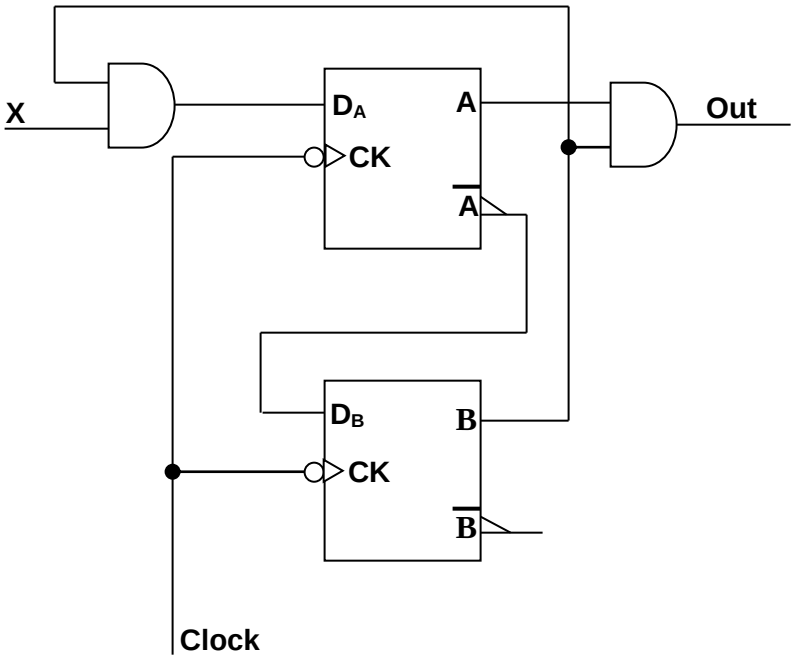
Figura: Modelo de estado Mealy.

A modelagem de um problema por Moore e por Mealy resulta num diagrama de estados, se descrito por Mealy, que pode ter um número de estados menor ou igual ao descrito por Moore.

Exemplo: As formas de ondas a seguir mostram o comportamento de um sistema digital, onde X é uma variável de entrada A e B representam a saída de cada estado e Out a saída do sistema digital. O sistema é síncrono e sensível à borda de descida do relógio.



O circuito a seguir gerou as formas de ondas apresentadas na figura. A seguir apresentamos a tabela de estados presentes e futuros, de descrição da máquina de estados.



Consiste de uma lógica de entrada das variáveis e as saídas dos F/Fs utilizados como variáveis internas e a lógica combinacional formada por portas lógicas. A lógica de entrada consiste de portas lógicas E de entrada e a lógica de saída são portas E saída, com os sinais lógicos X e A e B, temos a tabela de estados e de saída do circuito digital sequencial.

	Estado Atual		Entrada	Entradas F/Fs		Próximo Estado		Saída
Fila	A	B	X	D_A	D_B	A	B	Out
1	0	0	0	0	1	0	1	0
2	0	0	1	0	1	0	1	0
3	0	1	0	0	1	0	1	0
4	0	1	1	1	1	1	1	0
5	1	0	0	0	0	0	0	0
6	1	0	1	0	0	0	0	0
7	1	1	0	0	0	0	0	1
8	1	1	1	1	0	1	0	1

Para a implementação da máquina de estados um presente mapa de estados é utilizado para definir o nome dos estados. Como são 02 F/Fs somente 04 estados são possíveis. O mapa de estados presentes mostra estes estados possíveis em função das saídas dos F/Fs A e B. O mapa de estado define como ***a*** A = 0 e B = 0 ou 00, o estado ***b*** como 01, estado ***c*** como 10 e o estado ***d*** como 11. A designação de estado no mapa presente de estados será arbitrária, embora a designação de estados deve obedecer 02 princípios da adjacência para obter minimização do circuito. O asterisco no estado *d* identifica-o como estado de saída.

A / B		
	a	**c**
	b	**d***

A implementação com F/Fs do tipo D pode ser realizada, com a tabela de estados presentes A e B e a variável de entrada X. A saída Out = A.B.

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

D_A = X.B

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

$D_B = \overline{A}$

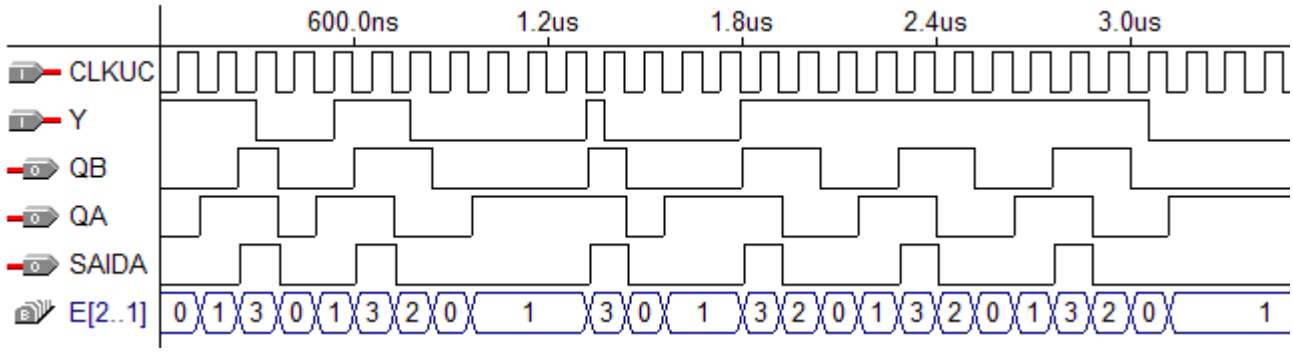
As equações de estado e saída implementadas no MAX-PLUS II, são:

Q_A = X.B e Q_B = A’

A implementação em AHDL é apresentada a seguir conforme o programa.

```
SUBDESIGN EXEMPLO_1_UC
(
CLKUC : INPUT;
Y : INPUT;
QA,QB,SAIDA : OUTPUT;
)
Variable E[2..1] : DFF;
BEGIN
E[].CLK=!CLKUC;
E[1] = !E2;
E[2] = (Y & E1);
QA = E1;
QB = E2;
SAIDA = E1 & E2;
END;
```

As formas de ondas apresentadas no MAX-PLUS II, são apresentadas, a seguir :

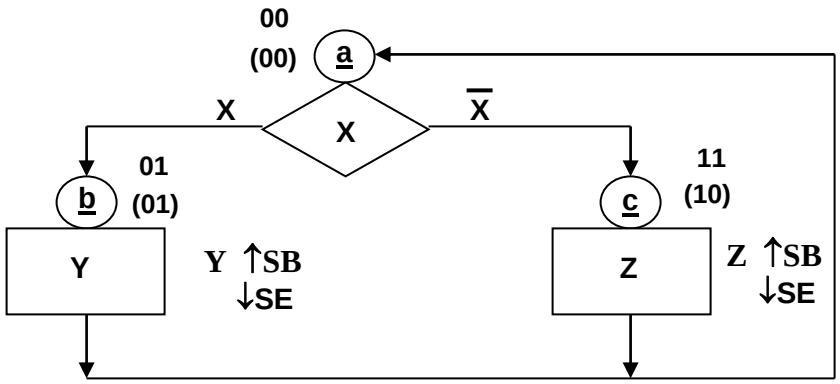


O próximo exemplo mostra um diagrama de estados com vários estados internos e várias saídas com muitos caminhos que resultam no seguinte diagrama de estados, apresentado na figura .

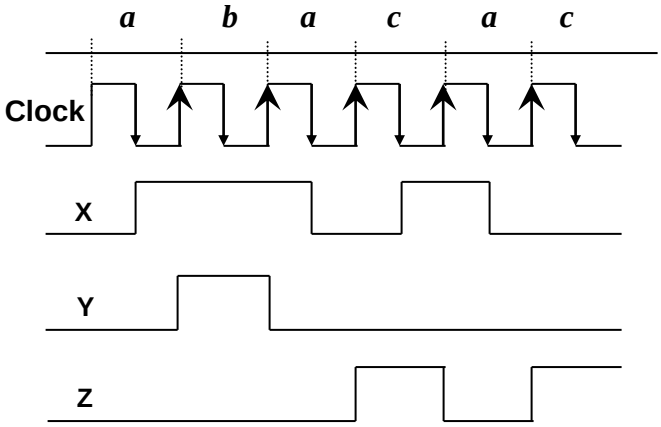
O próximo exemplo mostra um diagrama de estados com vários estados internos e várias saídas com muitos caminhos que resultam no seguinte diagrama de estados, apresentado na figura .

Exemplo: Construir um sistema sequencial para uma linha serial de dados X sincronizada com a borda de descida do clock. Uma máquina de estados será projetada para detectar o próximo valor de X. Se X =1, a saída Y deve ser Set. Se X = 0, a saída Z deve ser Set. O valor de saída será Set por um tempo de estado e então o próximo valor de X deve ser conferido.

SOLUÇÃO: O diagrama de estado a seguir reflete o comportamento especificado na figura . Como as trocas de estados do dado X ocorre com a borda de descida do clock, a troca de estado da variável X se dará somente com a transição negativa do clock. O sistema inicia no estado ***a*** e se X = 1 o sistema se move para o estado ***b***; se X = 0, o sistema se move para o estado ***c***, todas trocas de estados se darão com a transição positiva do clock. A saída em ambos estes estados existe por um tempo de clock após o qual o sistema retorna ao estado ***a*** para conferir o dado bit alternado. O diagrama de estado a seguir mostra a evolução do sistema sequencial.



A solução do problema é de um circuito sequencial pois a saída somente deve ser Set por um tempo de clock, mas poderia ser implementado com um circuito combinatório em conjunto com um sinal periódico do clock. As formas de ondas a seguir da figura mostram o comportamento do sistema com as variáveis X de entrada e as saídas Y e Z.

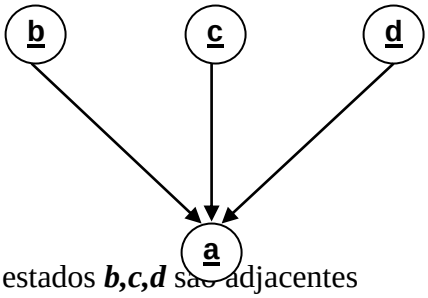


Os 03 estados que satisfazem as condições do problema, podem ser solucionados por 02 F/Fs A e B, onde A representa o MSB do código de estado. O projeto IFL, requer a designação de estado e será feito obedecendo os 02 princípios básicos para a minimização do circuito final. Usando métodos tradicionais para a implementação dos circuitos digitais combinacionais, o princípio da adjacência de estados leva a uma minimização final do circuito. Para comparar os resultados finais, o exemplo que estamos estudando será implementado obedecendo e não, os princípios das adjacências com designação correta e incorreta de estados.

PRINCÍPIO 1: Estados tendo o mesmo próximo estado para uma dada condição de entrada deve ter logicamente designação de estado adjacente.

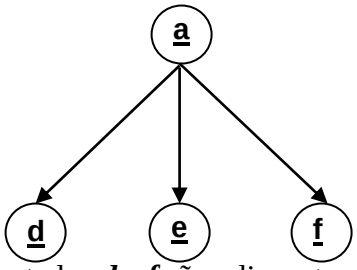
PRINCÍPIO 2: Estados que são os próximos estados de um único estado devem ter logicamente designações adjacentes.

Princípio 1



Os estados ***b,c,d*** são adjacentes .

Princípio 2



Os estados ***d,e,f*** são adjacentes.

Montamos 02 mapas de designações de estados, sendo uma correta e outra arbitrária para demonstração dos resultados. O mapa a) será corretamente designado obedecendo os princípios das adjacências conforme a seguir. Os estados ***b*** e ***c***, obedecem ao princípio 1, pois independente de qualquer condição estes levam para o mesmo estado ***a***. E também obedecem ao princípio 2 pois são os próximos estados ***b*** e ***c*** do produto de um único estado ***a.*** O mapa b) os estados serão montados arbitrariamente. Uma das designações corretas é apresentada no mapa a), mas existe outra solução corretamente designada.

A \ B	0	1
0	*a*	X
1	*b*	*c*

a) Estados designados corretamente.

A \ B	0	1
0	*a*	*c*
1	*b*	X

b) Estados designados arbitrariamente.

Onde X é um estado irrelevante.

O próximo passo é desenvolver os mapas para os F/Fs A e B. Os códigos designados para os estados ***a*** – 00, ***b*** – 01, ***c*** – 11. Os estados apresentados entre parênteses no diagrama de estados são incorretamente escolhidos e são ***a*** – 00, ***b*** – 01 e ***c*** – 10.

a) Estados designados corretamente.

A \ B	0	1
0	$\overline{X}$	X
1	0	0

F/F A

A \ B	0	1
0	1	X
1	0	0

F/F B

b) Estados arbitrariamente designados.

A \ B	0	1
0	$\overline{X}$	0
1	0	X

F/F A

A \ B	0	1
0	X	0
1	0	X

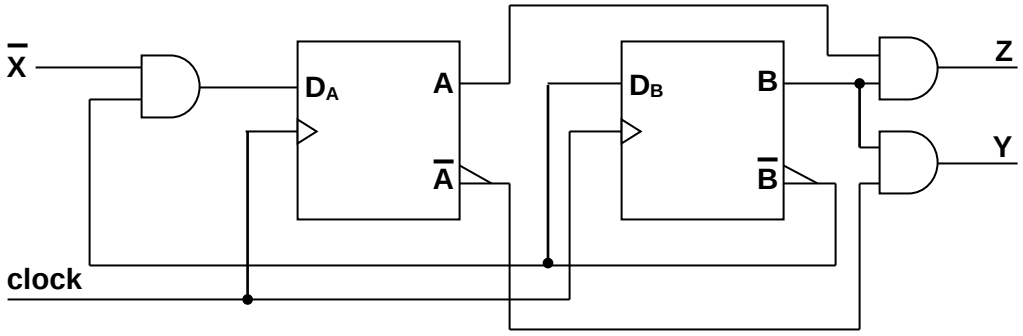
F/F B

O próximo passo é escolher o tipo de F/F a ser utilizado na implementação do circuito. Escolhemos inicialmente os F/Fs do tipo D, mais fácil de se implementar.

Para o item a) retiramos 02 funções D_A e D_B entradas de cada F/F A e B. Assim sendo:

$D_A = \overline{X} \cdot \overline{B}$ e $D_B = \overline{B}$.

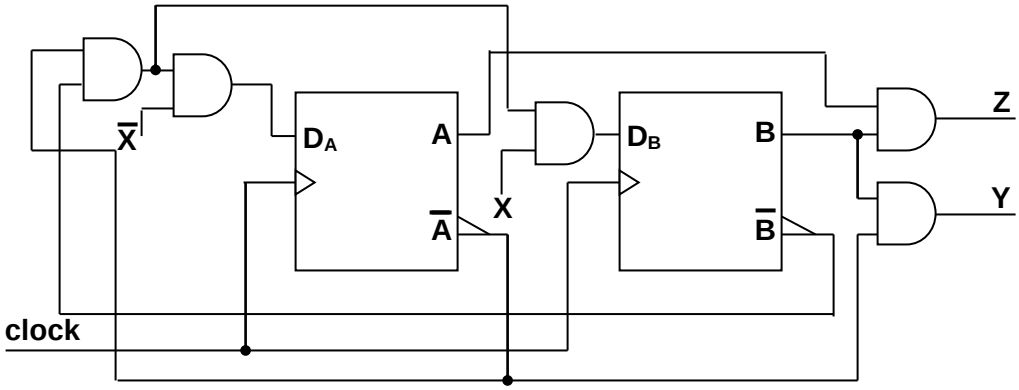
A implementação do circuito será:



Para o item b) retiramos 02 funções D_A e D_B entradas de cada F/F A e B. Assim sendo:

$D_A = \overline{X} \cdot \overline{A} \cdot \overline{B}$ e $D_B = X \cdot \overline{A} \cdot \overline{B}$.

Pela expressão pode-se notar que não são minimizadas as expressões de D_A e D_B pois há mais termos. O circuito a seguir mostra a implementação do item b).



Uma outra forma de implementação do problema anterior é mostrada a seguir, onde após a designação de estados, a tabela de estados é assim montada.

ATUAIS	**ENTRADA K**		**SAÍDAS**	
	0	**1**		
Q_BQ_A	**D_BD_A**	**D_BD_A**	**Y**	**Z**
00	11	01	0	0
01	00	00	1	1
10	XX	XX	X	X
11	00	00	0	1

Após a montagem da tabela, o circuito combinatório a ser projetado é tirado do mapa de Karnaugh.

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

$D_A = \overline{X}\,\overline{B}$

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

$D_B = \overline{B}$

As saídas Y e Z serão:

$Y = \overline{Q_B}\, Q_A$ e $Z = Q_A + Q_B$

A tabela de estados presentes e futuros será montada a seguir.

	Estado Atual		Entrada	Entradas F/Fs		Próximo Estado		Saída	
Fila	A	B	X	D_A	D_B	A	B	Y	Z
1	0	0	0	1	1	1	1	0	0
2	0	0	1	0	1	0	1	0	0
3	0	1	0	0	0	0	0	1	0
4	0	1	1	0	0	0	0	1	0
5	1	0	0	Θ	Θ	Θ	Θ	Θ	Θ
6	1	0	1	Θ	Θ	Θ	Θ	Θ	Θ
7	1	1	0	0	0	0	0	0	1
8	1	1	1	0	0	0	0	0	1

A implementação com outros tipos de F/Fs, como tipo JK é necessário da tabela de transição de estados ou tabela de excitação.

J	K	$Q_n \rightarrow Q_{n+1}$
0	X	$0 \rightarrow 0$
1	X	$0 \rightarrow 1$
X	1	$1 \rightarrow 0$
X	0	$1 \rightarrow 1$

A tabela de estados presentes e futuros será montada a seguir para F/F JK.

	Estado Atual		Entrada	Próximo Estado		Entradas dos F/Fs			
Fila	A	B	X	A	B	J_A	K_A	J_B	K_B
1	0	0	0	1	1	1	X	1	X
2	0	0	1	0	1	0	X	1	X
3	0	1	0	0	0	0	X	X	1
4	0	1	1	0	0	0	X	X	1
5	1	0	0	X	X	X	X	X	X
6	1	0	1	X	X	X	X	X	X
7	1	1	0	0	0	X	1	X	1
8	1	1	1	0	0	X	1	X	1

A próxima etapa é a implementação da lógica formadora de entrada dos F/Fs, construindo 04 mapas de Veitch Karnaugh.

AB / X	00	01	11	10
0	1	0	Θ	Θ
1	0	0	Θ	Θ

$J_A = \overline{B}\ \overline{X}$

AB / X	00	01	11	10
0	Θ	Θ	1	Θ
1	Θ	Θ	1	Θ

$K_A = 1$

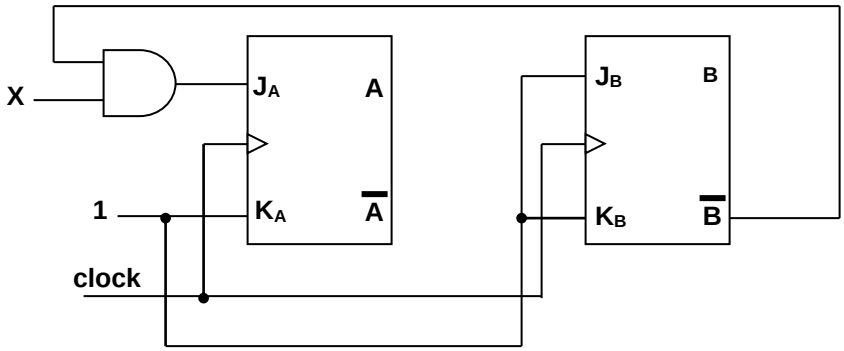
AB / X	00	01	11	10
0	1	Θ	Θ	Θ
1	1	Θ	Θ	Θ

$J_B = 1$

AB / X	00	01	11	10
0	Θ	1	Θ	Θ
1	Θ	1	Θ	Θ

$K_B = 1$

O circuito implementado com F/F do tipo JK é apresentado na figura.

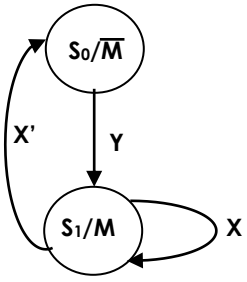


Módulo 03 - IMPLEMENTAÇÕES POR EQUAÇÃO DE ESTADOS E DE SAÍDA DOS MODELOS DE MEALY E DE MOORE 1 BIT POR ESTADO.

1. Introdução: Quando o número de variáveis em um problema quando passa de cinco variáveis, a implementação tradicional fica mais complexa devido ao número de combinações possíveis. O preenchimento de cada linha de uma tabela de estados e saída exige análises para condições não presentes e assim torna o processo por vezes bastante impraticável. O objetivo deste trabalho é apresentar uma estrutura de implementação simples, por equações de estados. Para cada estado é associado um bit (um F/F).

1.1 Equações de Estados retiradas de um diagrama de estados Modelo de Moore.

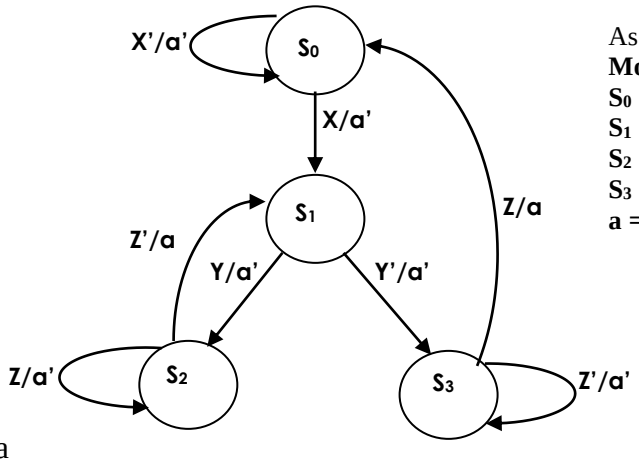
a) Caso I - Equações de Estados



Equação de estados e saída

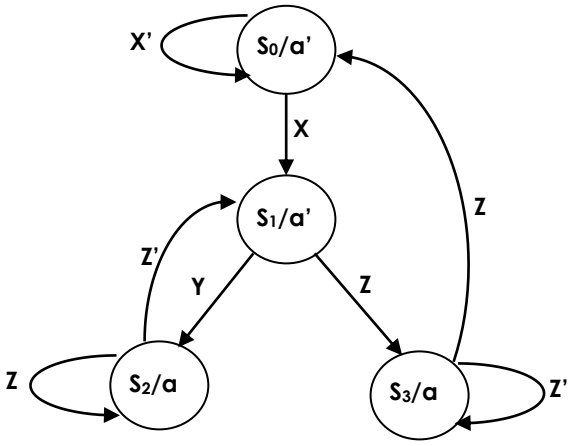
Modelo = Moore
$S_1 = S_0 Y + S_1 X$
$S_0 = S_1 X'$
$M = S_1$.

b) Caso II - Equações de Estados



As equações de estados e saída
Modelo = Mealy
$S_0 = S_3 Z + S_0 X'$
$S_1 = S_0 X + S_2 Z$
$S_2 = S_1 Y + S_2 Z$
$S_3 = S_1 Y' + S_3 Z'$
$a = S_2 Z' + S_3 Z$

2. Exemplos: Determinar as equações de estados e saída para os modelos sequenciais descritos.

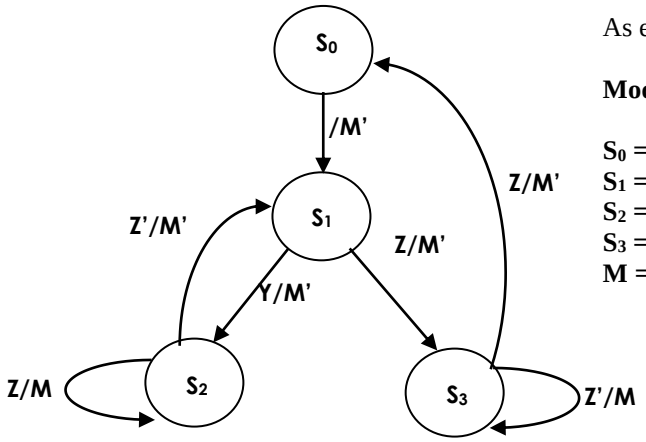


As equações de estados e saída.

Modelo = Moore

$S_0 = S_3 Z + S_0 X'$
$S_1 = S_0 X + S_2 Z'$
$S_2 = S_1 Y + S_2 Z$
$S_3 = S_1 Z + S_3 Z'$
$a = S_2 + S_3$

3. Exemplo: Determinar as equações de estados e saída

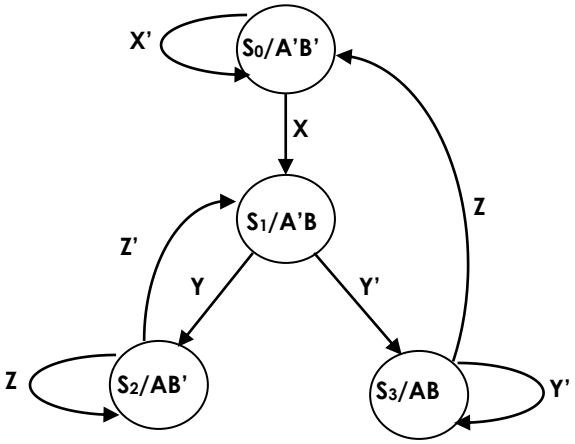


As equações de estados e saída

Modelo = Mealy

$S_0 = S_3\,Z$
$S_1 = S_0 + S_2\,Z'$
$S_2 = S_1\,Y + S_2\,Z$
$S_3 = S_1\,Z + S_3\,Z'$
$M = S_2\,Z + S_3\,Z'$

4. Exemplo: Determinar as equações de estados e saída**.**



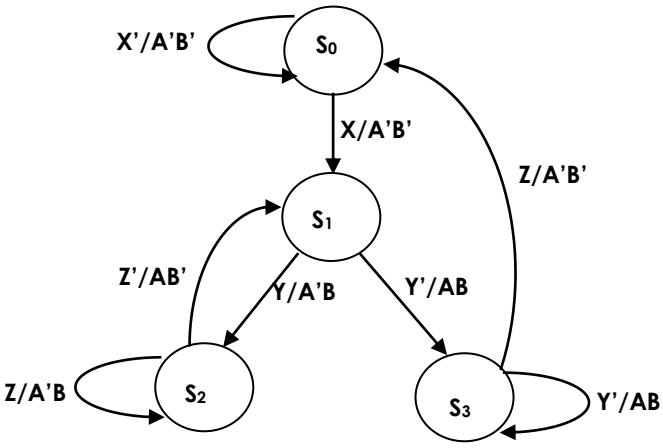
As equações de estados e saída

Modelo = Moore

$S_0 = S_0\,X' + S_3\,Z$
$S_1 = S_0\,X + S_2\,Z'$
$S_2 = S_1\,Y + S_2\,Z$
$S_3 = S_1\,Y' + S_3\,Y'$

$A = S_2 + S_3$
$B = S_1 + S_3$

Exemplo: Determinar as equações de estados e de saída.



As equações de estados e saída.

Modelo = Mealy

$S_0 = S_0\,X' + S_3\,Z$
$S_1 = S_0\,X + S_2\,Z'$
$S_2 = S_1\,Y + S_2\,Z$
$S_3 = S_1\,Y' + S_3Y'$

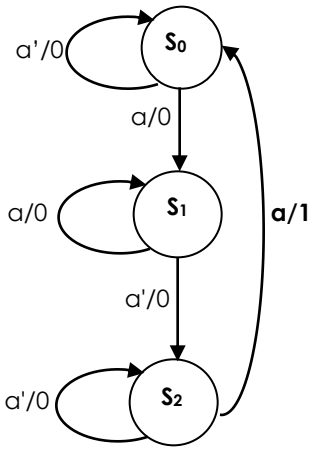
$A = S_1Y' + S_3Y' + S_2Z'$
$B = S_1 + S_2Z + S_3Y'$

IMPLEMENTAÇÃO DE MÁQUINAS DE ESTADOS POR EQUAÇÕES DE ESTADO

Dado o diagrama de estados pelo modelo de Mealy, pede-se:

a) Equações de estado e saída.
b) Implementação da FSM por ALP.
c) Simulação no MAX-PLUS.

SOLUÇÃO: Independente do número de variáveis de entrada e de estados as equações de estados e de saída, o diagrama de estados pelo modelo de Mealy, fica:



a) As equações de estados e de saída.

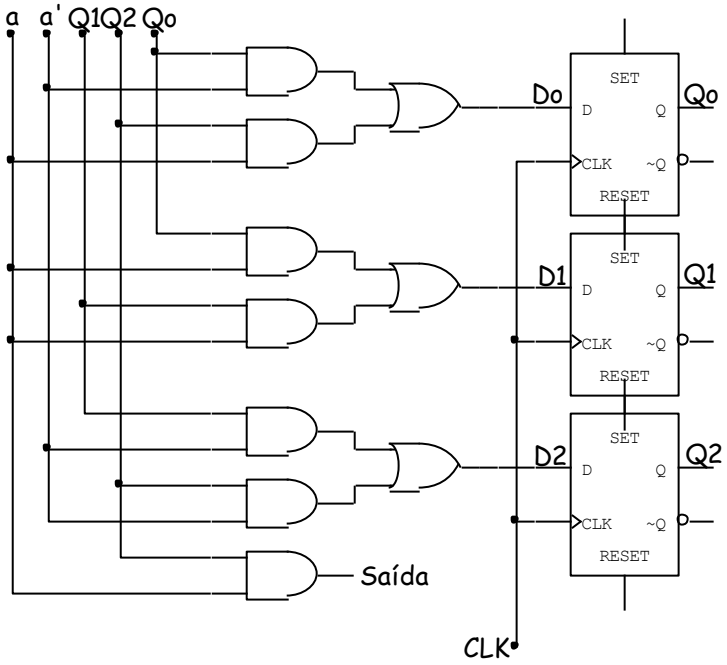
$S_0 = S_0a' + S_2a \Rightarrow Q_0 = Q_0a' + Q_2a$
$S_1 = S_0a + S_1a => Q_1 = Q_0a + Q_1a$
$S_2 = S_1a' + S_2a' \Rightarrow Q_2 = Q_1a' + Q_2a'$

$Saída = S_2a = Q_2a$

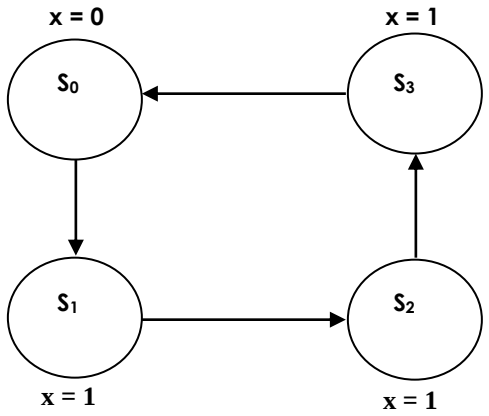
Utilizando-se de flip-flop tipo D, as equações booleanas para as suas entradas ficam:

$D_0 = Q_0a' + Q_2a$, $D_1 = Q_0a + Q_1a$ e $D_2 = Q_1a' + Q_2a'$

O circuito final fica:



Exemplo 6.15: Para a F.S.M. a seguir, a qual gera repetidamente uma sequência de saída 0,1,1 e 1. Pede-se:
a) As equações de estados e saída.
b) Tabela de estados da F.S.M. usando lógica de 1 bit por estado
c) Implementar a F.S.M. usando F/F tipo D.
d) Circuito final.



a) As equações de estados e saída.

$S_0 = S_3$ $S_1 = S_0$ $x = S_1 + S_2 + S_3$

$S_2 = S_1$ $S_3 = S_2$

b) A tabela de estados e saída.

Identificação dos estados

$S_0 = 0001$, $S_1 = 0010$, $S_2 = 0100$ $S_3 = 1000$

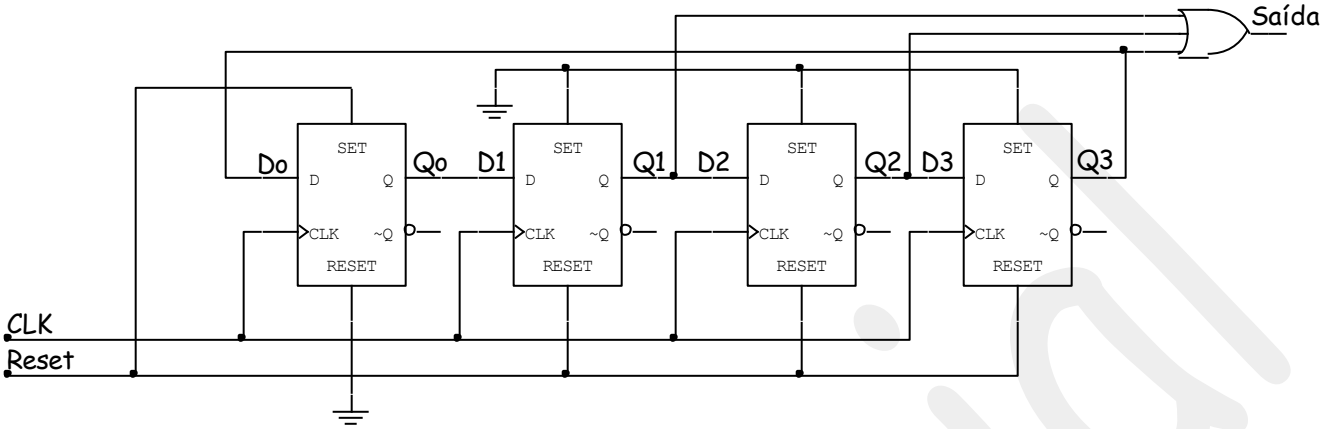
Estados	Atual S_3	Atual. S_2	Atual S_1	Atual S_0	Futuro D_3	Futuro D_2	Futuro D_1	Futuro D_0	Saída x
S_0	0	0	0	1	0	0	1	0	0
S_1	0	0	1	0	0	1	0	0	1
S_2	0	1	0	0	1	0	0	0	1
S_3	1	0	0	0	0	0	0	1	1

c) Implementação usando F/F tipo D.

$D_0 = S_3$ $D_1 = S_0$ $x = S_1 + S_2 + S_3$

$D_2 = S_1$ $D_3 = S_2$

O circuito final fica:



EXEMPLO: Deseja-se construir um comando automático de controle de um distribuidor pneumático. Dispõe-se do distribuidor com 02 saídas AV – Avanço do cilindro e AR – Recuo do cilindro. É um cilindro pneumático de dupla-ação, sendo monitorado por 02 sensores A cilindro avançado e R cilindro recuado. Para início de operação um botão M faz o cilindro partir e ficar oscilando entre os pontos A e R e para parar um botão Em faz o cilindro completar a operação toda e parar sobre o ponto R. Pede-se:
a) Definir as variáveis de entrada e saída;
b) Definir a lógica das variáveis de entrada e saída;
c) Diagrama de estados pelo modelo de Moore;
d) Implementação com a estrutura de equações de estados e saída.

Solução:

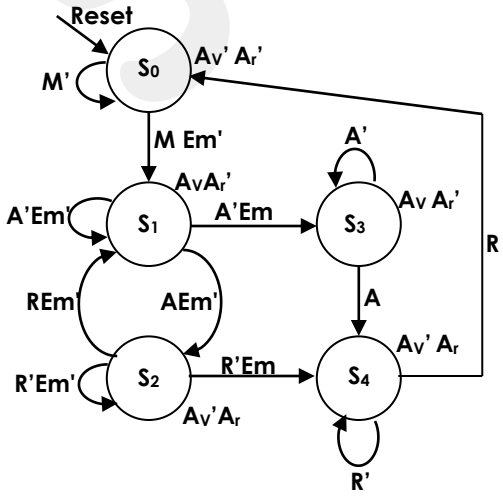
a) Variáveis de entrada e saída.

Entrada: M,A,R,Em
Saída: AV, AR.

b) Lógica das variáveis.

Entrada: variável = 0 ⇒ desativada e variável = 1 ⇒ ativa.
Saída: variável = 0 ⇒ desligada e variável =1 ⇒ ligada.

c) Diagrama de estados.



Designação de estados

S_0 = 000
S_1 = 001
S_2 = 010
S_3 = 011
S_4 = 100

Equações de Estados e Saída.
Modelo = Moore

$S_0 = S_0\ M' + S_4R$
$S_1 = S_0M\ Em' + S_1A'\ Em' + S_2R\ Em'$
$S_2 = S_1A\ Em' + S_2R\ Em'$
$S_3 = S_1A'\ Em + S_2A'$
$S_4 = S_2R'\ Em + S_3A + S_4R'$

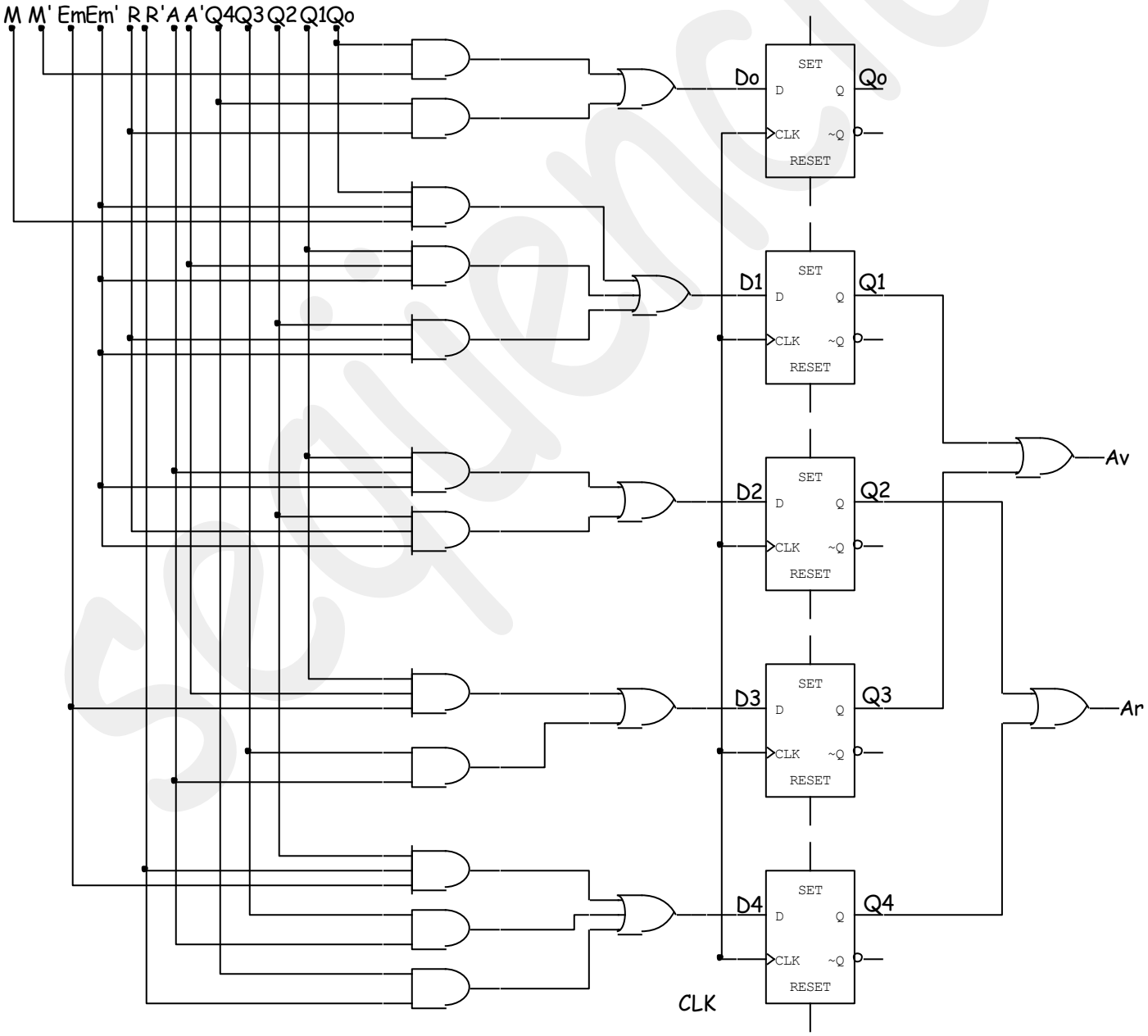
$AV = S_1 + S_3$
$AR = S_2 + S_4$

As equações de estados usando F/F do tipo D, temos:

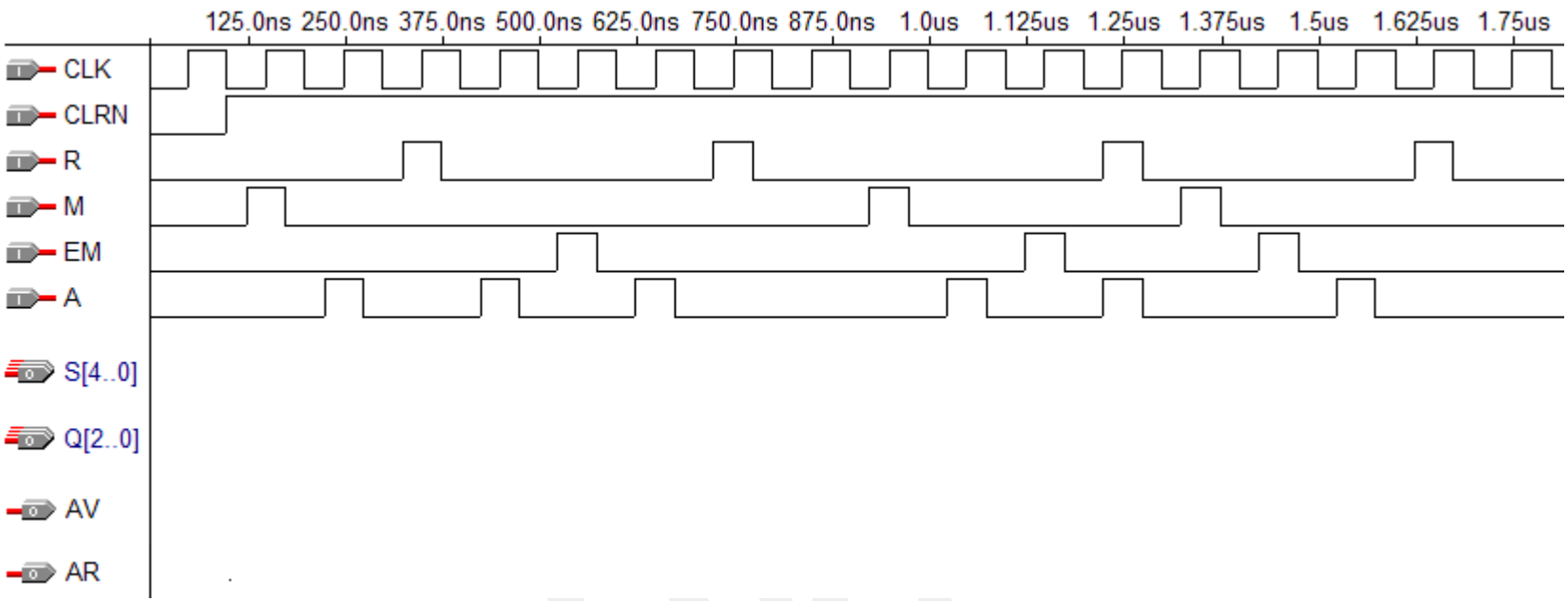
$D_0 = Q_0\ M' + Q_4R$
$D_1 = Q_0M\ Em' + Q_1A'\ Em' + Q_2R\ Em'$
$D_2 = Q_1A\ Em' + Q_2R\ Em'$
$D_3 = Q_1A'\ Em + Q_3A'$
$D_4 = Q_2R'\ Em + Q_3A + Q_4R'$

$AV = Q_1 + Q_3$
$AR = Q_2 + Q_4$

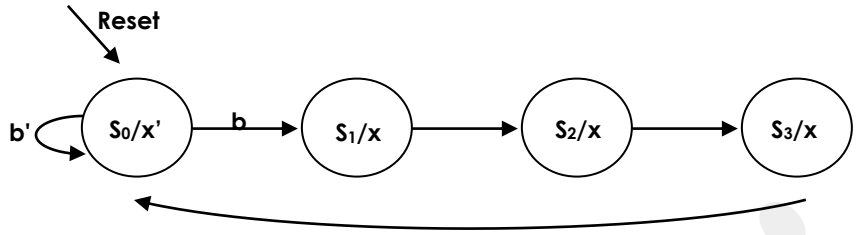
a) O circuito fica:



d) A simulação do circuito, fica:



Exemplo 6.16: Repetir o problema do laser usando a codificação de 1 bit por estado. Pede-se:
a) Diagrama de estados e saída com codificação de 1 bit por estado.
b) Equações de estados e saída.
c) Tabela de estados e saída com codificação de 1 bit por estado
d) Circuito final.



a) Equações de estados
a) As equações de estados e saída.

$S_0 = S_3 + S_0b'$ $S_1 = S_0b$ $x = S_1 + S_2 + S_3$

$S_2 = S_1$ $S_3 = S_2$

b) A tabela de estados e saída.

Identificação dos estados

$S_0 = 0001$, $S_1 = 0010$, $S_2 = 0100$ $S_3 = 1000$

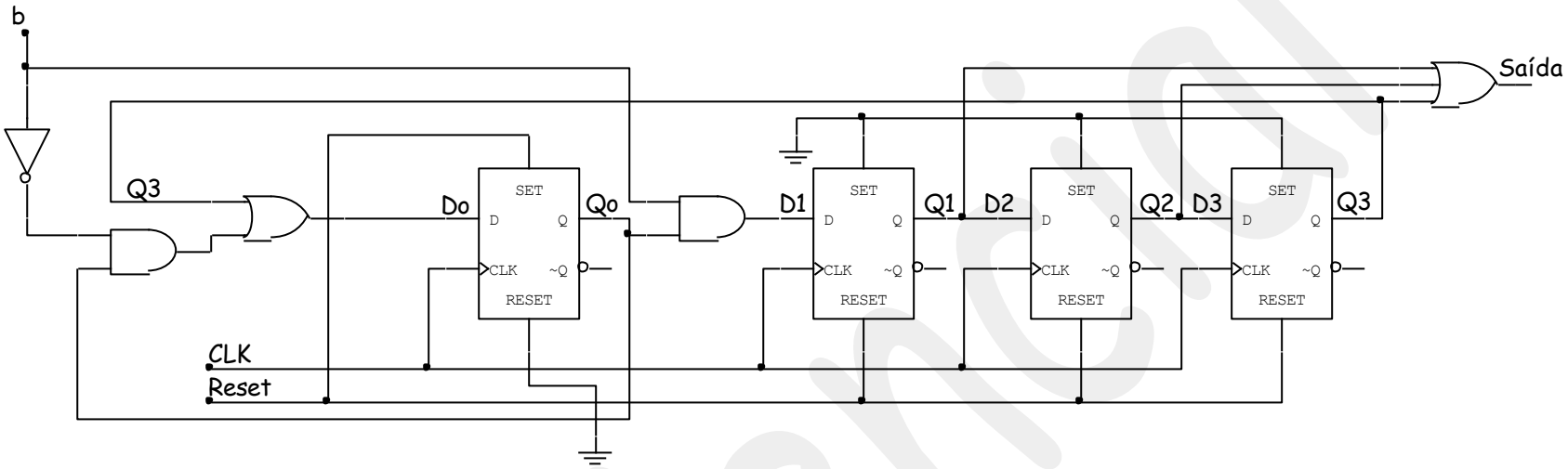
Estados	Atual S_3	Atual. S_2	Atual S_1	Atual S_0	Futuro D_3	Futuro D_2	Futuro D_1	Futuro D_0	Saída x
S_0	0	0	0	1	0	0	1	0	0
S_1	0	0	1	0	0	1	0	0	1
S_2	0	1	0	0	1	0	0	0	1
S_3	1	0	0	0	0	0	0	1	1

c) Implementação usando F/F tipo D.

$D_0 = Q_3 + Q_0b'$ $D_1 = Q_0b$ $x = Q_1 + Q_2 + Q_3$

$D_2 = Q_1$ $D_3 = Q_2$

O circuito final, fica:



EXEMPLO: Implementar a lógica de controle para um móvel o qual inicialmente se encontra em repouso no ponto A. O operador possui um botão P que acionado faz com que o móvel M se movimente em direção ao ponto B. Chegando em B o móvel inverte o sentido de direção e segue para o ponto A. Chegando em A o móvel pára. Pede-se:
a) Identificar as variáveis de entrada e de saída.
b) Definição da lógica das variáveis de entrada e saída.
c) O diagrama de estados pelo modelo de Moore e de Mealy.
d) As equações de estados e saída, pelos modelos de Moore e de Mealy.
e) Implementação por máquina de estados, usando a estrutura de implementação com a codificação de estados.

Adotar:

a) Variáveis de entrada: P,A,B
Variáveis de saída : A_V,A_r

b) Lógica das variáveis:

Entrada: Variável = 1 – Ativa e Variável = 0 – Não ativa.
Saída: Variável = 1 – Ligado e Variável = 0 – Desligado.

a) Implementação geração de sequência em VHDL.

```
ENTITY geraseq IS
   PORT(
      clk               : IN BIT;                          -- RELÓGIO
      q                 : OUT BIT_VECTOR(1 DOWNTO 0);  -- saída indica estado
      x                 : OUT BIT);                        -- sinal de saída
END geraseq;

ARCHITECTURE teste OF geraseq IS
   TYPE maq_estado IS (s0, s1, s2, s3);
   SIGNAL state: maq_estado;
BEGIN
   PROCESS (clk)
   BEGIN
      IF clk'EVENT AND clk = '1' THEN

            CASE state IS

                  WHEN S0 =>                                 -- q = 0
                        state <= s1;                         -- q = 1


                  WHEN s1 =>
                        state <= s2;                         -- q = 2

                  WHEN s2 =>
                        state <= s3;                         -- q = 3

                  WHEN s3 =>
                        state <= s0;                         -- q = 0

            END CASE;
            END IF;

            CASE state IS
                  WHEN s0  => x <= '0';                      -- saída = 0
                  WHEN s1  => x <= '1';                      -- saida = 0
                  WHEN s2  =>       x <= '1';                -- saida = 0
                  WHEN s3  =>       x <= '1';                -- saida = 0
                  END CASE;
            END PROCESS;
            WITH state SELECT
            q <="00" WHEN     s0,                            -- q = 0
                  "01" WHEN s1,                              -- q = 1
                  "10" WHEN s2,                              -- q = 2
                  "11" WHEN s3;                              -- q = 3
            END teste;
```

b) Implementação do cilindro pneumatico em VHDL

```
ENTITY cilindro IS
   PORT(
        clk                 : IN BIT;                          -- RELÓGIO
        M                   : IN BIT;                          -- sinal de inicio

‘1 A                        : IN BIT;                          -- sinal de entrada de
cilindro avançado
   R                        : IN BIT;                          -- sinal de entrada de
cilindro recuado
   Em                       : IN BIT;                          -- sinal de emergencia
   q                        : OUT BIT_VECTOR(2 DOWNTO 0);      -- saída indica estado
   Av                       : OUT BIT;                         -- sinal de saída avança
cilindro
   Ar                       : OUT BIT);                        -- sinal de saída recua
cilindro
END cilindro;

ARCHITECTURE teste OF cilindro IS
   TYPE maq_estado IS (espera, avança, recua, avança1, recua1);
   SIGNAL state: maq_estado;
BEGIN
   PROCESS (clk)
   BEGIN
        IF clk'EVENT AND clk = '1' THEN
                CASE state IS

                        WHEN espera =>                                  -- q = 0
                                IF      M = '1' THEN
                                        state <= avança;                -- q = 1
                                ELSIF M = '0' THEN
                                        state <= espera;                -- q = 0
                                END IF;

                        WHEN avança =>

                          IF    A = '0' AND Em = '0' THEN
                                state <= avança;                        -- q = 1
                            ELSIF A = '1' AND Em = '0' THEN
                                        state <= recua;                 -- q = 2
                                ELSIF A = '0' AND Em = '1' THEN
                                        STATE <= avança1;               -- q = 3
                                END IF;

                        WHEN recua =>
                                IF    R = '0' AND Em = '0' THEN
                                state <= recua;                         -- q = 2
                                ELSIF R = '0' AND Em = '1' THEN
                                        state <= recua1;                -- q = 4
                                ELSIF R = '1' AND Em = '0' THEN
```

```
                                    state <= avança;                    -- q = 1
                            END IF;

                        WHEN avança1 =>
                            IF    A = '0' THEN
                            state <= avança1;                           -- q = 3
                            ELSIF A = '1' THEN
                                    state <= recua1;                    -- q = 4
                            END IF;

                        WHEN recua1 =>
                            IF    R = '0' THEN
                            state <= recua1;                            -- q = 4
                        ELSIF R = '1' THEN
                                    state <= espera;                    -- q = 0
                            END IF;

                END CASE;
                            END IF;
    CASE state IS
        WHEN espera  => Av <= '0' ; Ar <= '0';-- saídas Avanço e Recua desligadas
        WHEN avança  => Av <= '1' ; Ar <= '0';-- saidas Avanço ligado e Recuo desligado
        WHEN recua   => AV <= '0' ; Ar <= '1';-- saidas Avanço desligado e Recua ligado
        WHEN avança1 =>   Av <= '1' ; Ar <= '0';-- saidas Avanço ligado e Recua desligado
        WHEN recua1  =>   Av <= '0' ; Ar <= '0';  -- saidas Avanço desligado e Recua ligado
                END CASE;
                END PROCESS;
                WITH state SELECT
                q <="000" WHEN     espera,
    -- q = 0
                        "001" WHEN avança,                                  -- q = 1
                        "010" WHEN recua,                                   -- q = 2
                        "011" WHEN avança1,                                 -- q = 3
                        "100" WHEN recua1;                                  -- q = 4
                END teste;
```

Implementação do móvel em VHDL.

```
ENTITY carro IS
    PORT(
        clk                 : IN BIT;               -- RELÓGIO
        P                   : IN BIT;               -- sinal de inicio
        A                   : IN BIT;               -- sinal de entrada de carro avançado
        B                   : IN BIT;       -- sinal de entrada de carro recuado
        q                   : OUT BIT_VECTOR(1 DOWNTO 0);-- saída indica estado
        Ma                  : OUT BIT;                  -- sinal de saída avança carro
        Mr                  : OUT BIT);                 -- sinal de saída recua carro
END carro;
ARCHITECTURE teste OF carro IS
    TYPE maq_estado IS (espera, avança, recua);
    SIGNAL state: maq_estado;
```

```
BEGIN
   PROCESS (clk)
   BEGIN

      IF clk'EVENT AND clk = '1' THEN

            CASE state IS

                  WHEN espera =>
                        IF   P = '0' THEN
                              state <= espera;          -- q = 0
                        ELSE
                        state <= avança;                -- q = 1
                        END IF;

                  WHEN avança =>
                        IF         A = '1' THEN
                              state <= recua;           -- q = 2
                        ELSE
                              state <= avança;          -- q = 1
                        END IF;

                  WHEN recua =>

                     IF    B = '0' THEN
                        state <= recua;                 -- q = 2
                     ELSE
                                    state <= espera;    -- q = 2
                        END IF;
            END CASE;

                  END IF;
            CASE state IS
      WHEN espera  => Ma <= '0' ; Mr <= '0';     -- saídas Avanço e Recua desligadas
      WHEN avança  => Ma <= '1' ; Mr <= '0';     -- saidas Avanço ligado e Recuo desligado
      WHEN recua   => Ma <= '0' ; Mr <= '1';     -- saidas Avanço desligado e Recua ligado
            END CASE;
            END PROCESS;
            WITH state SELECT
            q <="00" WHEN       espera,                                 -- q = 0
                  "01" WHEN     avança,                                 -- q = 1
                  "10" WHEN     recua;                                  -- q = 2

             END teste;
```

Modulo 04: EXERCÍCIOS DE MÁQUINAS SEQÜENCIAIS e F.S.M.

1. Introdução: O objetivo é resolver problemas sequenciais através de modelos de descrições de sistemas sequenciais e implementação por codificação de estados ou por equações de estados.

Exemplo: Construir uma máquina sequencial capaz de gerar uma saída Z ALTO sempre que uma sequência na entrada X de três bits consecutivos forem iguais a 000. Pede-se:

a) Diagrama de estados pelos modelos de Mealy e de Moore.
b) Equações de estados e saída do sistema.
c) Tabela de estados e saída usando a solução de estados codificados e 1 bit por estado.
d) Representação da máquina de estados para o item c).
e) Implementação da F.S.M. usando F/F tipo D.

Exemplo: Construir uma máquina sequencial capaz de gerar uma saída Z ALTO sempre que uma sequência na entrada X de três bits consecutivos forem iguais a 111. Pede-se:

a) Diagrama de estados pelos modelos de Mealy e de Moore.
b) Equações de estados e saída do sistema.
c) Tabela de estados e saída usando a solução de estados codificados e 1 bit por estado.
d) Representação da máquina de estados para o item c).
e) Implementação da F.S.M. usando F/F tipo D.

Exemplo: Construir uma máquina sequencial capaz de gerar uma saída Z ALTO sempre que uma sequência na entrada X de três bits consecutivos forem iguais a 101. Pede-se:

a) Diagrama de estados pelos modelos de Mealy e de Moore.
b) Equações de estados e saída do sistema.
c) Tabela de estados e saída usando a solução de estados codificados e 1 bit por estado.
d) Representação da máquina de estados para o item c).
e) Implementação da F.S.M. usando F/F tipo D.

Exemplo: Um móvel se encontra estacionado no ponto A (A = 1). Após pressionar um botão C (C = 1), O móvel sai do ponto A ligando o motor M sentido avante (AM = 1) segue até o ponto B (B = 1). No ponto B o acionamento do motor M é sentido invertido (IM = 1) deixa o ponto B e retorna ao ponto A e pára. Pede-se:

a) Diagrama de estados pelos modelos de Mealy e de Moore.
b) Equações de estados e saída do sistema.
c) Tabela de estados e saída usando a solução de estados codificados e 1 bit por estado.
d) Representação da máquina de estados para o item c).
e) Implementação da F.S.M. usando F/F tipo D.

Exemplo: Um disco D se encontra estacionado no ponto Z (Z = 1). Após pressionar o botão B (B = 1) o disco é acionado para girar (D = 1) deixa o ponto Z e gira uma volta e pára novamente no ponto Z. Pede-se:
a) Diagrama de estados pelos modelos de Mealy e de Moore.
b) Equações de estados e saída do sistema.
c) Tabela de estados e saída usando a solução de estados codificados e 1 bit por estado.
d) Representação da máquina de estados para o item c).

e) Implementação da F.S.M. usando F/F tipo D.

Exemplo: Construir um somador serial binário. Pede-se:
a) Diagrama de estados pelos modelos de Mealy e de Moore.
b) Equações de estados e saída do sistema.
c) Tabela de estados e saída usando a solução de estados codificados e 1 bit por estado.
d) Representação da máquina de estados para o item c).
e) Implementação da F.S.M. usando F/F tipo D.

Exemplo: Dois móveis se encontram estacionados respectivamente nos pontos A e B (A = B = 1). Após pressionador o botão M (M = 1), os 2 móveis partem acionando os respectivos motores no sentido avante (AMA = 1) e (AMB = 1) e permanecem acionados até chegar nos pontos C e D (C = 1 e D = 1). Como as velocidades não são exatamente as mesmas para AMA e AMB, um dos móveis chegará mais rápido que o outro. Antes de qualquer ação aquele móvel chegar primeiro nos pontos C ou D, deverá esperar o outro móvel chegar. Após os 2 móveis se encontrarem respectivamente nos pontos C e D, o sentido do acionamento dos motores são invertidos (IMA = 1) e (IMB = 1) e os motores deixam os pontos C e D e partem para os pontos A e B. Chegando aos pontos A e B os móveis devem parar. Pede-se:
a) Diagrama de estados pelos modelos de Mealy e de Moore.
b) Equações de estados e saída do sistema.
c) Tabela de estados e saída usando a solução de estados codificados e 1 bit por estado.
d) Representação da máquina de estados para o item c).
e) Implementação da F.S.M. usando F/F tipo D.

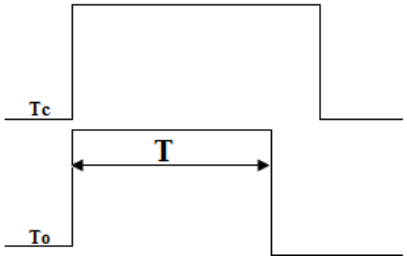
Exemplo: Construir uma máquina sequencial para uma máquina de refrigerante. A entrada da máquina pode receber moedas de $5 e de $10. O custo do refrigerante é de $15. Sempre que a somatória das moedas for igual ao custo do refrigerante a máquina libera o refrigerante. A máquina não fornece troco. Pede-se:
a) Diagrama de estados pelos modelos de Mealy e de Moore.
b) Equações de estados e saída do sistema.
c) Tabela de estados e saída usando a solução de estados codificados e 1 bit por estado.
d) Representação da máquina de estados para o item c).
e) Implementação da F.S.M. usando F/F tipo D.

Exemplo: Construir uma máquina sequencial para uma máquina de refrigerante. A entrada da máquina pode receber moedas de $5 e de $10. O custo do refrigerante é de $15. Sempre que a somatória das moedas for igual ao custo do refrigerante a máquina libera o refrigerante. A máquina fornece troco. Pede-se
a) Diagrama de estados pelos modelos de Mealy e de Moore.
b) Equações de estados e saída do sistema.
c) Tabela de estados e saída usando a solução de estados codificados e 1 bit por estado.
d) Representação da máquina de estados para o item c).
e) Implementação da F.S.M. usando F/F tipo D.

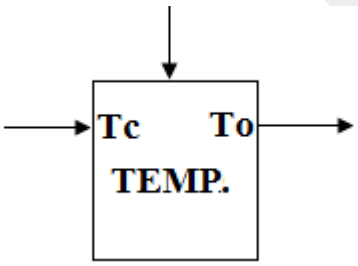
Exemplo: Um dispositivo muito utilizado nos projetos de automação e controle é chamado de temporizador. Existem vários tipos de temporizadores, assim como o descrito pelas formas de ondas a seguir. O temporizador é do tipo conhecido como "OFF-DELAY" onde é ligado (Set) quando a entrada é desligada (Reset). A constante de tempo do temporizador é realizada no bloco TEMP mostrado a seguir, onde T é a constante de tempo e é iniciada numa borda positiva de um sinal aplicado à entrada T_C e produzindo uma saída T_O a qual permanecerá ligada (Set) enquanto decorre

a constante de tempo e é desligada (Reset) no término dela. As formas de ondas do mecanismo do temporizador e saída descrevem como o temporizador opera o OFF-DELAY. Pede-se:
a) Realizar a modelagem do mecanismo do temporizador pelo modelo de Moore.
b) As equações de estados do modelo descrito.
c) A implementação do temporizador 1 bit por estado.

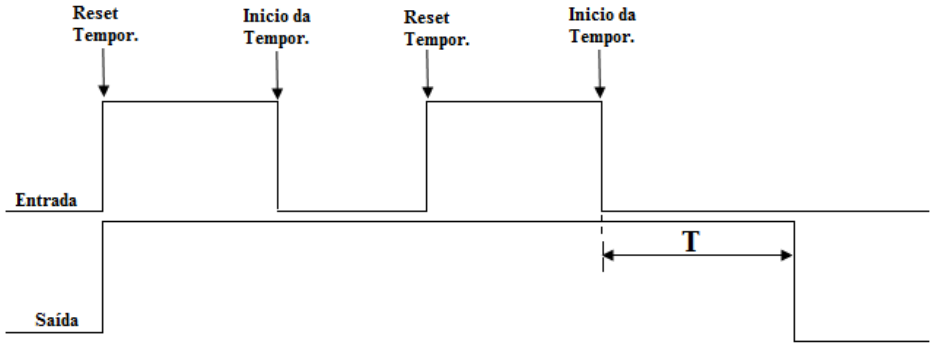
Formas de ondas do TEMP.



Bloco temporizador

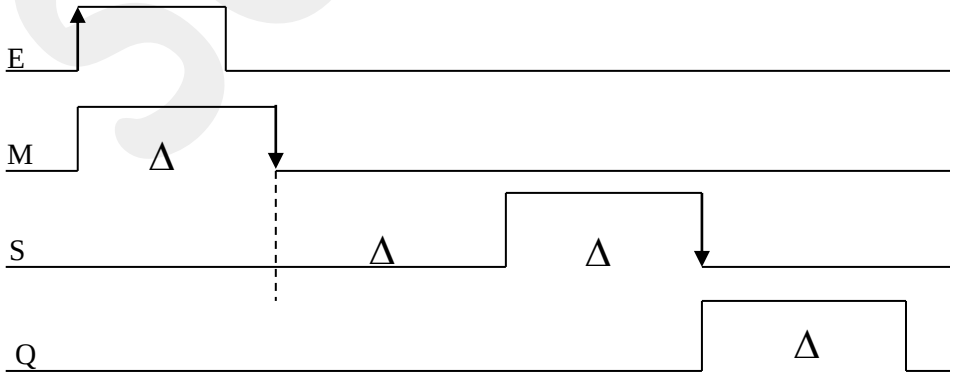


Mecanismo do temporizador



Tempor. -> Temporização

Exemplo: Um temporizador é descrito pelos sinais a seguir. Sendo E a entrada do temporizador e Q a saída do temporizador. As saídas M e S são saídas intermediárias à saída Q. Dispõe-se de um timer disparado por uma transição na entrada positiva ou negativa na sua entrada T e uma saída do timer QT = 1 durante a constante de tempo igual a Δ e QT = 0 no final da temporização. Pede-se:
a) Diagrama de estados pelo modelo de Moore.
b) Equações de estados e saída do sistema.
c) Tabela de estados e saída usando a solução de estados codificados e 1 bit por estado.
d) Representação da máquina de estados para o item c).
e) Implementação da F.S.M. usando F/F tipo D.



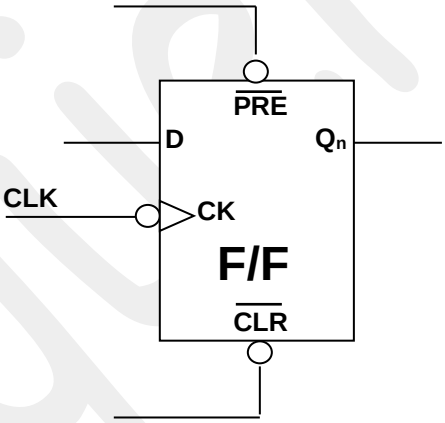
Módulo 05: Entradas Assíncronas, preset e clear, borda de subida e descida, set-up e hold, associação de F/Flop tipo T, contador assíncrono crescente e decrescente. Livro Texto – pág. 149 a 152

Introdução: As entradas assíncronas Preset e Clear quando aplicadas a um dispositivo sequencial como um contador ou um flip-flop são independentes do relógio e quando ativas separadamente fazem mudanças de estados imediatamente. São utilizadas para carregar um estado inicial no sistema ou zeramento imediato. A tabela da verdade a seguir mostra o funcionamento das entradas. Os subsistemas contador síncrono e assíncrono, registrador utilizam as entradas assíncronas no zeramento, na carga paralela. Normalmente as entradas assíncronas vêm associadas a um controle de carga.

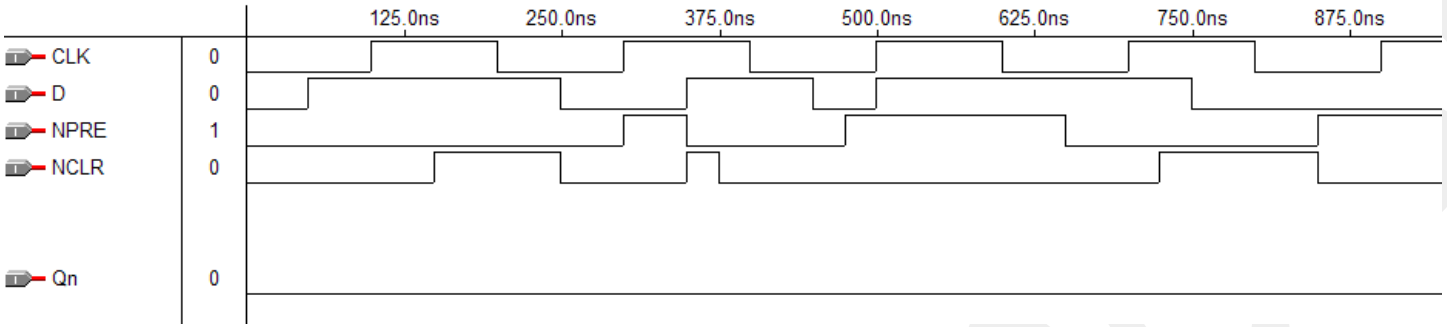
1. Entradas Assíncronas Preset e Clear.

PRE'	CLR'	D	CLK	Q_{n+1}
0	0	X	P	P
0	1	X	X	1
1	0	X	X	0
1	1	a	↓	a

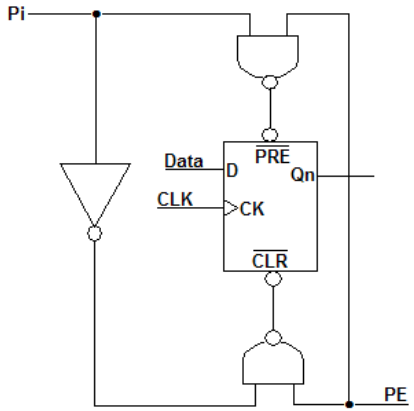
2. Flip– Flop tipo D síncrono borda de descida e com entradas assíncronas preset e clear.



3. Formas de Ondas no F/F tipo D acima.



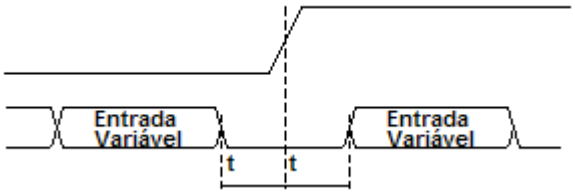
4. Implementação de uma célula única para um F/F tipo D borda de subida com única entrada assíncrona.



A tabela da verdade do circuito acima, fica:

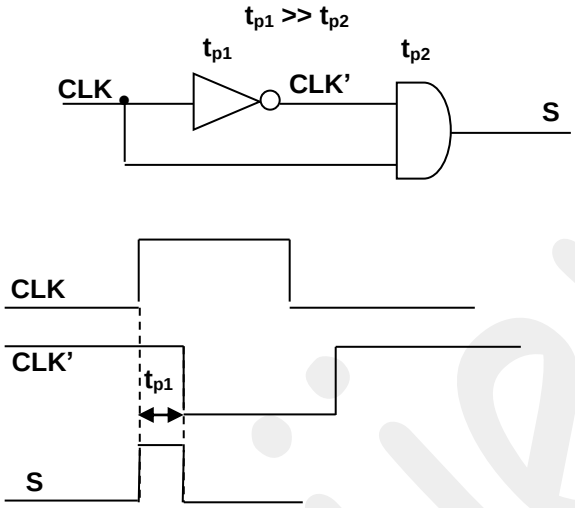
PE	P_i	D	CLK	Q_{n+1}
1	0	X	X	0
1	1	X	X	1
0	X	0	↑	0
0	X	1	↑	1

5. Tempos de set-up(t_{SET-UP}) e de holding(t_{hold}).

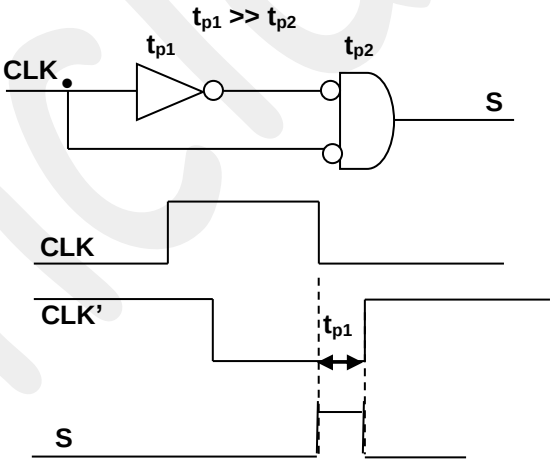


6. Borda de Subida ou positiva ou borda de Descida ou negativa.

a) Borda de Subida.

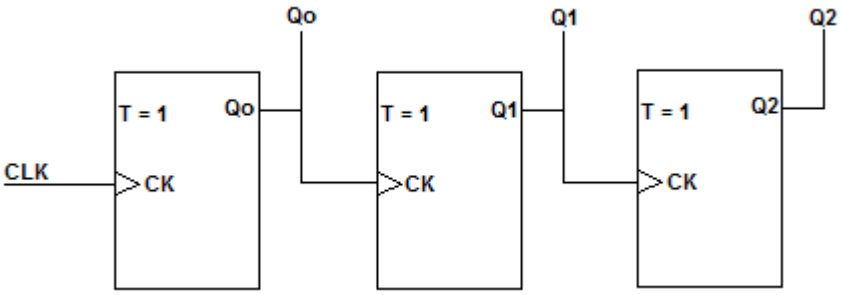


b) Borda de Descida

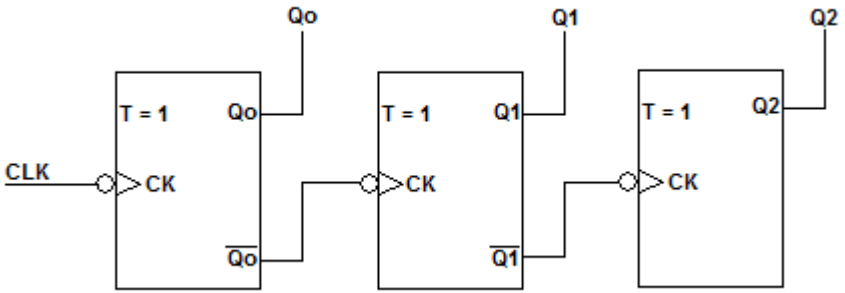


7. Associação de 3 F/Fs do tipo T com T = 1 em cada um dos F/Fs. pág. 197 a 200

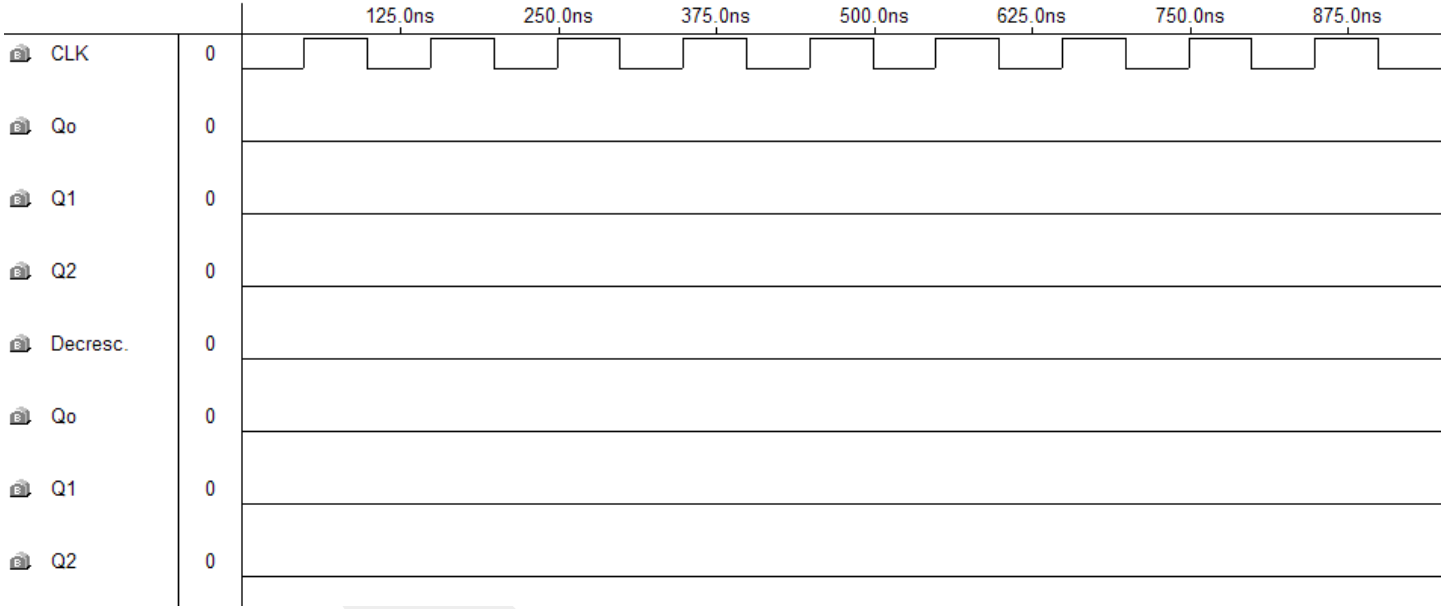
7.1 Modo Crescente



7.2 Modo Decrescente

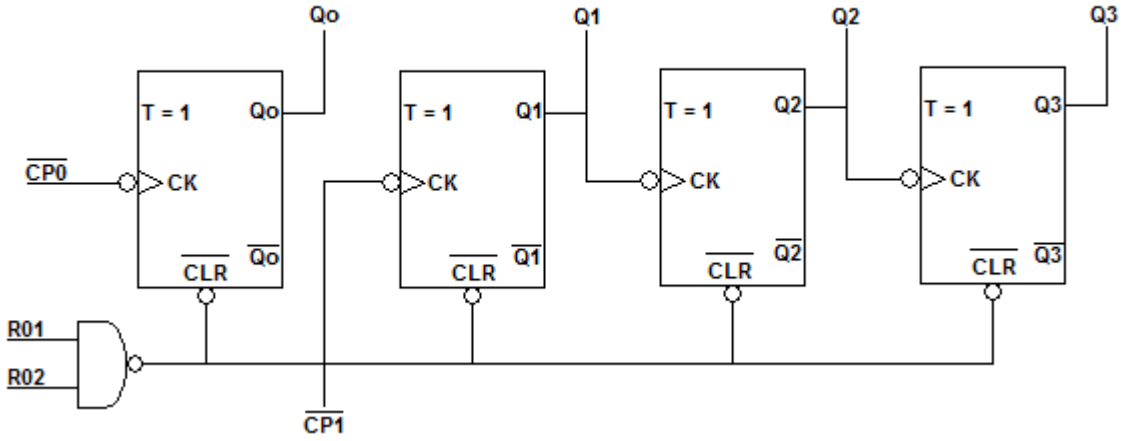


8. As formas de ondas em Q_2,Q_1 e Q_0, modos crescente e decrescente serão:

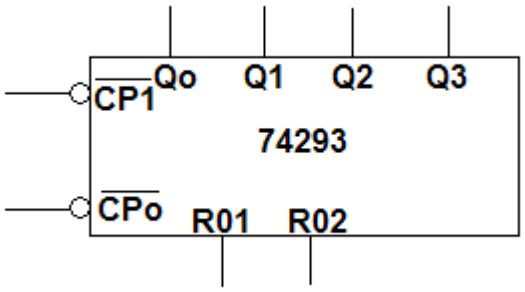


9. Contadores Assíncronos comerciais – CI 74293.

9.1 Configuração Interna.



9.2 Bloco contador assíncrono ou modulante, tabela da verdade e configuração – CI 74293



R01	R02	CP0	CP1	Estado Q's
1	1	x	x	Zera Q's
0	x	1	1	Mantém Q's
x	0	1	1	Mantém Q's
0	x	↓	1	Q_0'
0	X	1	↓	Q_1'

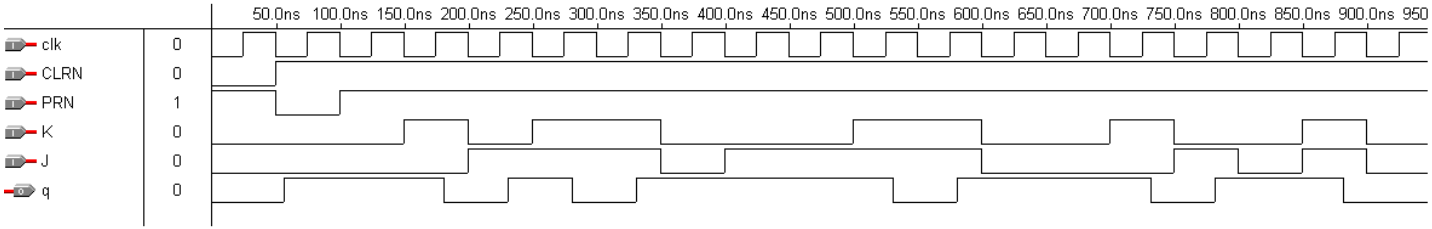
9.3 Resposta em frequência.

Considerando-se n flip-flops em cascata e o tempo de propagação de um F/F igual a T_P e da porta lógica T_{PORTA} a frequência máxima aplicada à associação será:

$$f_{MAX} = \frac{1}{n\,T_p + T_{PORTA}}$$

1. VHDL do flip-flop tipo JK com entradas assíncronas Preset e Clear.

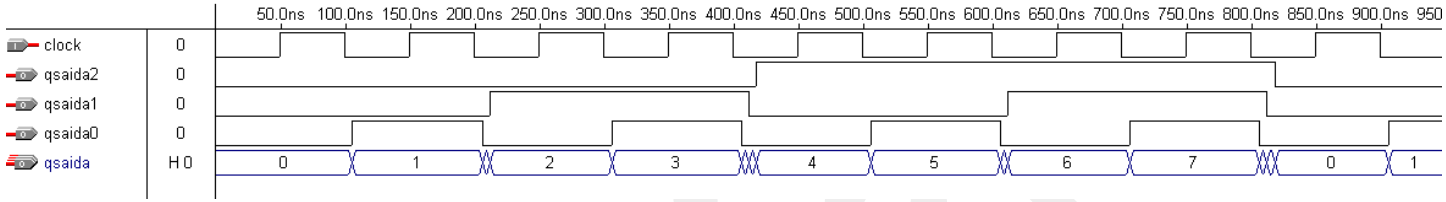
```
-- Flip-flop JK com entradas Preset e Clear
-- Assíncronas e JK síncrona
ENTITY jk_completo is
PORT(
clk,J,K,PRN, CLRN: IN BIT;-- ENTRADAS SÍNCRONAS E ASSINCRONAS
q                          :OUT BIT); -- Saída Q
END jk_completo;

ARCHITECTURE ff OF jk_completo IS
SIGNAL qestado      :BIT;
BEGIN
        PROCESS (clk, PRN, CLRN)        -- Sinais de entrada
                BEGIN
                        IF PRN =  '0' THEN qestado <= '1'; -- Preset Q
                        ELSIF CLRN = '0' THEN qestado <=  '0'; -- Limpa Q
                        ELSIF clk =  '1' AND clk'EVENT THEN -- Na borda de subida do clock
                                IF J =  '1' AND K =  '1' THEN qestado <= NOT qestado; -- Set Q
                                ELSIF J =  '1' AND K =  '0' THEN qestado <=  '1';
                                ELSIF J =  '0' AND K =  '1' THEN qestado <=  '0';
                                END IF;
                        END IF;
        END PROCESS;
        q <= qestado;
END   ff;
```



2. VHDL para um contador assíncronos de 3 bits, borda de descida.

```
ARCHITECTURE contador OF contador_tres_bits IS
        SIGNAL ALTO         :BIT;
        COMPONENT neg_jk
        PORT (clk,j,k :IN BIT;
        q         :OUT BIT);
        END COMPONENT;
BEGIN
        ALTO <='1';
ff0:    neg_jk PORT MAP (j => ALTO,
                                        K => ALTO,
                                        clk => clock,
                                        q => qsaida(0));
ff1:    neg_jk PORT MAP (j => ALTO,
                                        K => ALTO,
                                        clk => qsaida(0),
                                        q => qsaida(1));
ff2:    neg_jk PORT MAP (j => ALTO,
                                        K => ALTO,
                                        clk => qsaida(1),
                                        q => qsaida(2));
 END contador;

ENTITY neg_jk IS
PORT (clk,j,k :IN BIT;
        q                       :OUT BIT);
END neg_jk;

ARCHITECTURE ff OF neg_jk IS
SIGNAL qestado      :BIT;
BEGIN
        PROCESS (clk)       -- Sinais de entrada
```

```
        BEGIN
            IF (clk'EVENT AND clk ='0') THEN -- Na borda de subida do clock
                IF J ='1' AND K ='1' THEN qestado <= NOT qestado; -- Set Q
                ELSIF J ='1' AND K ='0' THEN qestado <= '1';
                ELSIF J ='0' AND K ='1' THEN qestado <= '0';
                END IF;
            END IF;
    END PROCESS;
    q <= qestado;
END   ff;
```

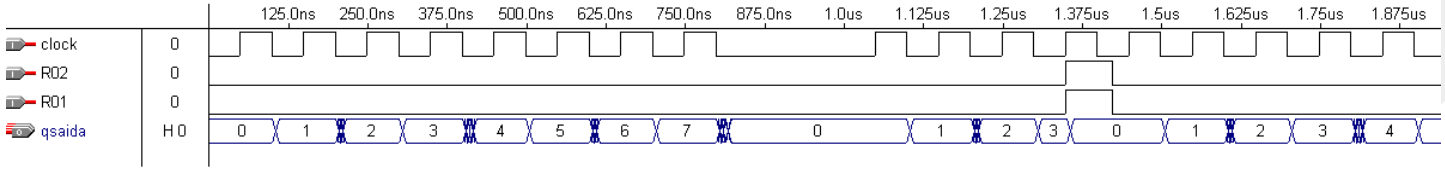


3. VHDL contador assíncrono com reset (R01 e R02)

```
--Contador de 3 bits com Flip-flop JK e entrada Reset R01  e R02
ENTITY contador_tres_bits_reset IS
PORT ( clock,R01,R02       :IN BIT;
         qsaida  : BUFFER BIT_VECTOR ( 2 DOWNTO 0));
END contador_tres_bits_reset;

ARCHITECTURE contador OF contador_tres_bits_reset IS
    SIGNAL ALTO,m   :BIT;
    COMPONENT neg_jk
    PORT (clk,j,k,clr     :IN BIT;
    q       :OUT BIT);
    END COMPONENT;
BEGIN
    ALTO <='1';
    m <= NOT (R01 AND R02);
```

```
ff0:    neg_jk PORT MAP (j => ALTO,
                         K => ALTO,
                        clk => clock,
                        clr => m,
                        q => qsaida(0));
ff1:    neg_jk PORT MAP (j => ALTO,
                         K => ALTO,
                        clk => qsaida(0),
                        clr => m,
                        q => qsaida(1));
ff2:    neg_jk PORT MAP (j => ALTO,
                         K => ALTO,
                        clk => qsaida(1),
                        clr => m,
                        q => qsaida(2));
 END contador;

ENTITY neg_jk IS
PORT (clk,j,k,clr     :IN BIT;
      q               :OUT BIT);
END neg_jk;


ARCHITECTURE ff OF neg_jk IS
SIGNAL qestado     :BIT;
BEGIN
      PROCESS (clk)      -- Sinais de entrada
            BEGIN
                  IF clr = '0' THEN qestado <=  '0'; -- Limpa Q
                  ELSIF (clk'EVENT AND clk ='0') THEN -- Na borda de descida do clock
                        IF J ='1' AND K ='1' THEN qestado <= NOT qestado; -- Set Q
                        ELSIF J ='1' AND K ='0' THEN qestado <= '1';
                        ELSIF J ='0' AND K ='1' THEN qestado <= '0';
                        END IF;
```

```
                END IF;
        END PROCESS;
        q <= qestado;
END     ff;
```

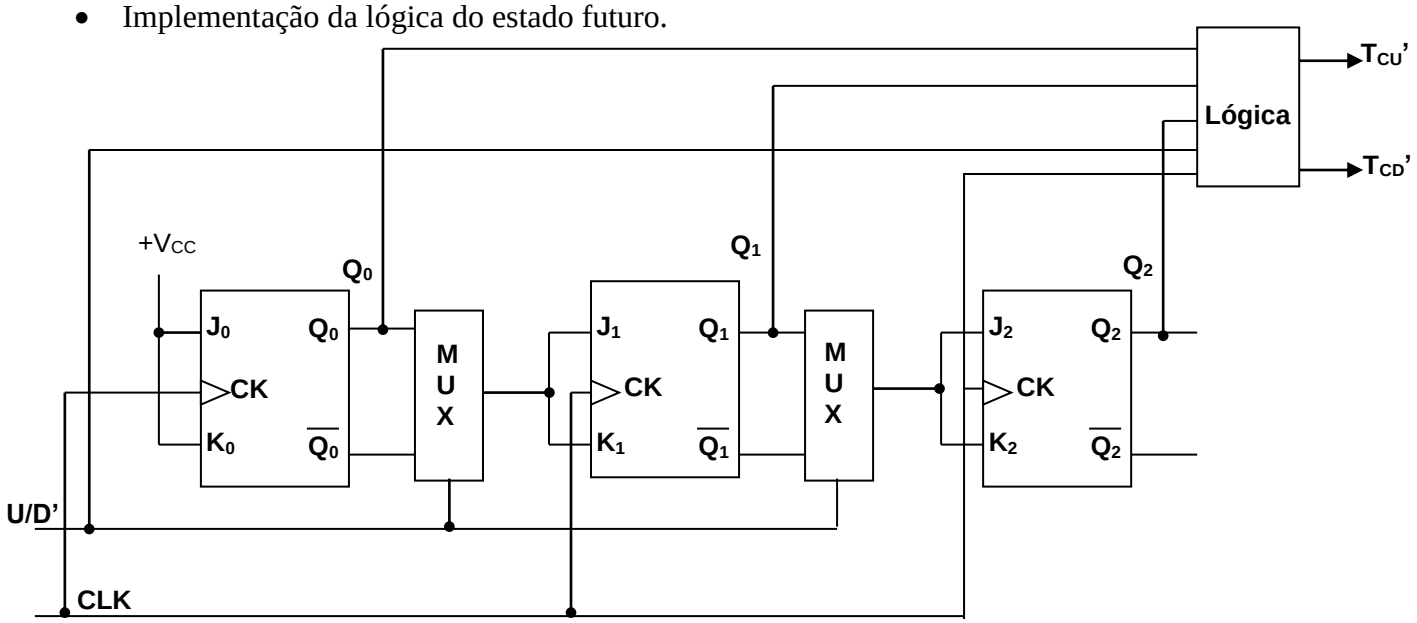


Módulo 06: Estudo do contador síncrono

1. Introdução: O estudo do contador síncrono e suas propriedades, implementação, resposta em frequência e formas de ondas. São contadores cujas associações dos F/Fs são interligadas por um único sinal de relógio. O resultado é a comutação simultânea dos flip-flops. Essa característica faz com que construir contadores mais rápidos do que os contadores assíncronos. Possui carga paralela que pode ser síncrona ou assíncrona através de um sinal de controle. A seguir apresentamos algumas das suas propriedades.

1.2 PROPRIEDADES

- Mudança de estado de cada F/F síncrono;
- Carga Paralela, no caso de contadores paralelos comerciais;
- O tempo de propagação de cada F/F e do MUX;
- Implementação da lógica do estado futuro.



Considerando-se n flip-flops, o tempo de propagação do F/F T_P e o tempo de propagação do MUX T_{PPORTA}, então a frequência máxima de operação será:

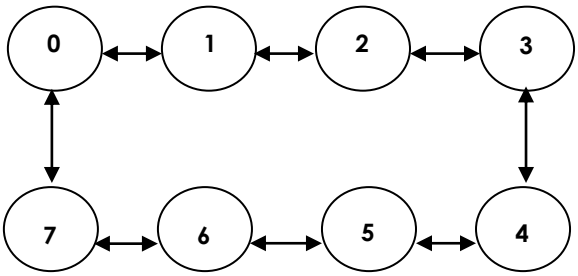
$$f_{MAX} = \frac{1}{T_P + (n-1)T_{PORTA}}$$

2. Projeto com contadores síncronos.

Exemplo: Implementar um contador síncrono módulo oito nos modos crescente e decrescente, selecionados por uma entrada U/D'. Para U/D' = 0 o modo é decrescente e U/D' = 1 modo crescente. A lógica de saída é de borda positiva e na passagem do estado sete para zero para T_{CU}' e de 0 para 7 para T_{CD}'. Pede-se:

a) A representação por diagrama de estados
b) A tabela de estados completa, sabendo-se que a saída do contador para o modo crescente deve ocorrer na passagem do estado sete para o estado zero borda de subida e no modo decrescente deve ocorrer na passagem do estado 0 para o estado sete e borda de subida.
c) Implementação da lógica do contador usando F/F JK, com as expressões booleanas simplificadas de cada entrada do F/F JK e da saída do contador.
d) O circuito final.
e) Formas de Ondas

a) Diagrama de estados



b) A tabela de transição do F/F JK.

J	K	$Q_n \rightarrow Q_{n+1}$
0	x	0 → 0
1	x	0 → 1
x	1	1 → 0
x	0	1 → 1

c) A tabela completa de estados e saída.

U/D'	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0	T_{CU}'	T_{CD}'
0	0	0	0	1	1	1	1	X	1	X	1	X	1	0
0	0	0	1	0	0	0	0	X	0	X	X	1	1	1
0	0	1	0	0	0	1	0	X	x	1	1	X	1	1
0	0	1	1	0	1	0	0	X	x	0	X	1	1	1
0	1	0	0	0	1	1	x	1	1	X	1	X	1	1
0	1	0	1	1	0	0	x	0	0	X	X	1	1	1
0	1	1	0	1	0	1	x	0	x	1	1	X	1	1
0	1	1	1	1	1	0	x	0	x	0	X	1	1	1
1	0	0	0	0	0	1	0	X	0	X	1	X	1	1
1	0	0	1	0	1	0	0	X	1	X	X	1	1	1
1	0	1	0	0	1	1	0	X	x	0	1	X	1	1
1	0	1	1	1	0	0	1	X	x	1	X	1	1	1
1	1	0	0	1	0	1	x	0	0	X	1	X	1	1
1	1	0	1	1	1	0	x	0	1	X	X	1	1	1
1	1	1	0	1	1	1	x	0	x	0	1	X	1	1
1	1	1	1	0	0	0	x	1	x	1	X	1	0	1

d) Mapas de Karnaugh para implementação da lógica do estado futuro.

$U/D'Q_2$	00	01	11	10
$Q_1Q_0$00	1	X	X	0
01	0	X	X	0
11	0	X	X	1
10	0	X	X	0

$J_2 = (U/D')'\ Q_1'Q_0' + U/D'Q_1Q_0$

$U/D'Q_2$	00	01	11	10
$Q_1Q_0$00	X	1	X	0
01	X	0	X	0
11	X	0	X	1
10	X	0	X	0

$K_2 = J_2$

$U/D'Q_2$	00	01	11	10
$Q_1Q_0$00	1	1	0	0
01	0	0	1	1
11	X	X	X	X
10	X	X	X	X

$J_1 = (U/D')'Q_0' + U/D'Q_0$

$U/D'Q_2$	00	01	11	10
$Q_1Q_0$00	X	X	X	X
01	X	X	X	X
11	0	0	1	1
10	1	1	0	0

$K_1 = J_1$

$U/D'Q_2$	00	01	11	10
Q_1Q_0 00	1	1	1	1
01	X	X	X	X
11	X	X	X	X
10	1	1	1	1

$U/D'Q_2$	00	01	11	10
Q_1Q_0 00	X	X	X	X
01	1	1	1	1
11	1	1	1	1
10	X	X	X	X

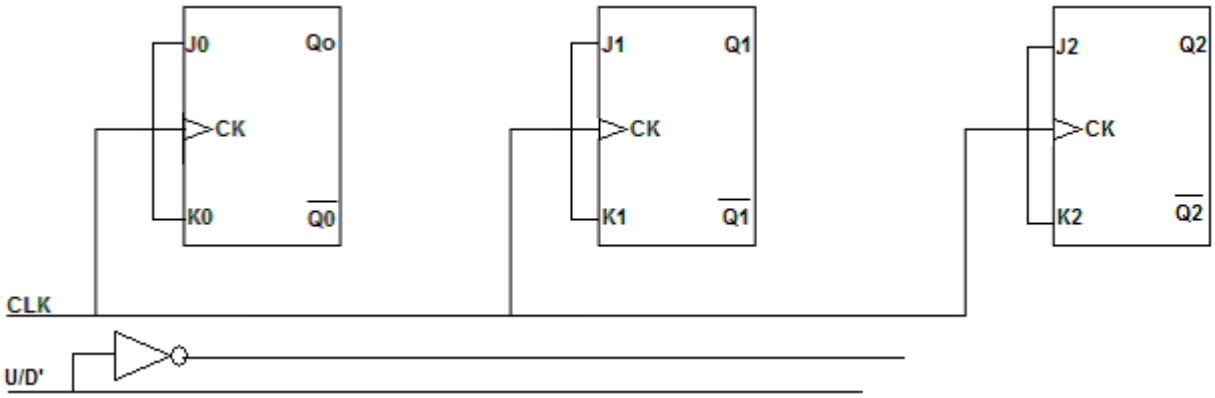
$J_0 = 1$

$K_0 = 1$

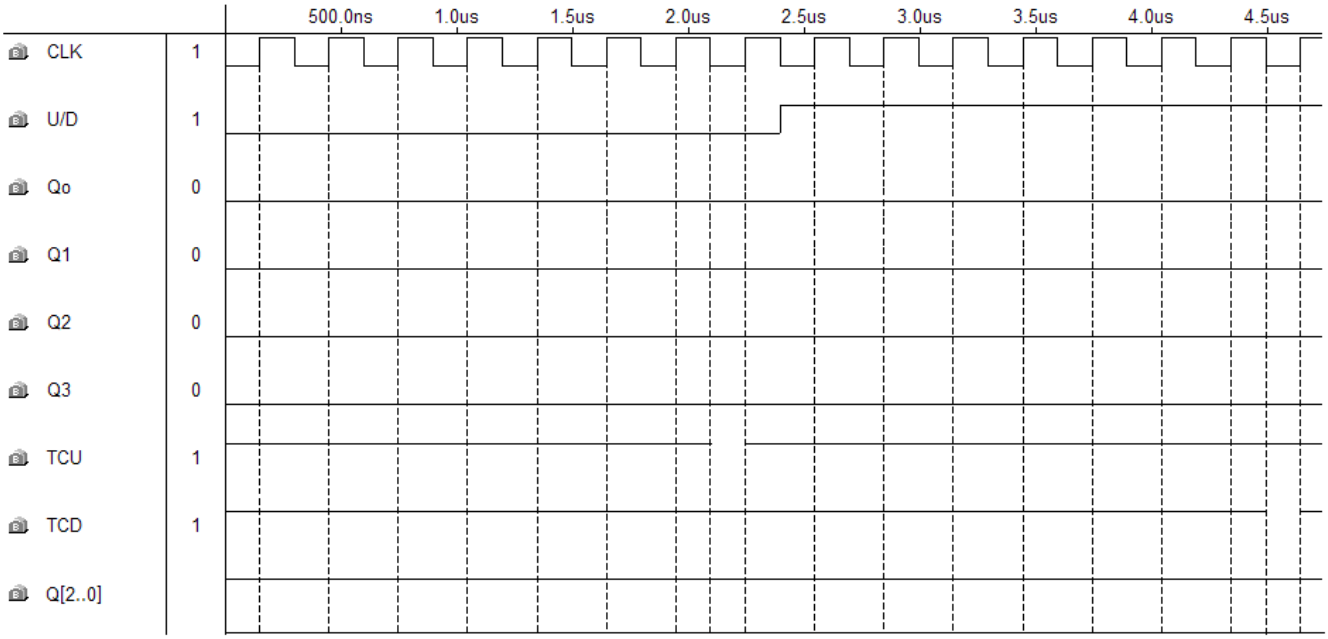
$T_{CU}' = [U/D' Q_2Q_1Q_0CLK']'$

$T_{CD}' = [(U/D')' Q_2'Q_1'Q_0'CLK']'$

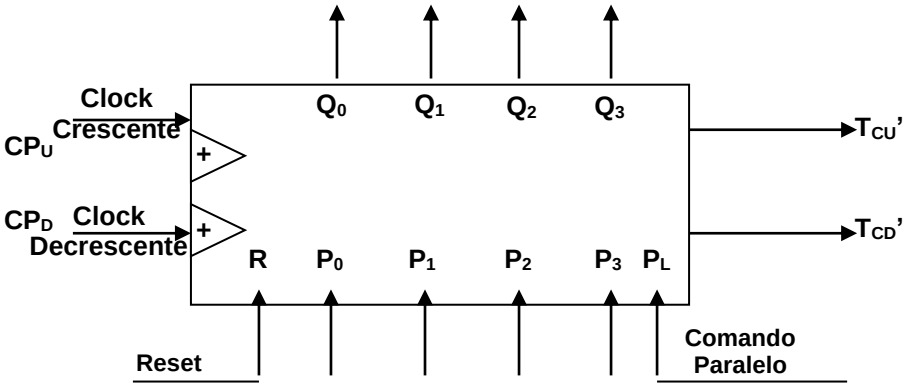
e) Circuito final.



f) Formas de Ondas



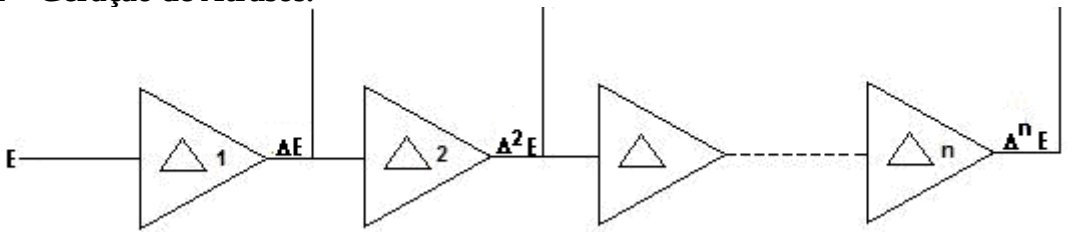
3. Contadores Síncronos comerciais – 74193.



Módulo 07: Registradores de Deslocamentos.

1. Introdução: Implementação de um registrador de deslocamento, conversão paralelo-série e série-paralelo, geração de atrasos, contador e implementação de um registrador de deslocamento com entradas externas. Esses dispositivos são fundamentais no nível RTL, sendo um bloco operacional.

I – Geração de Atrasos.

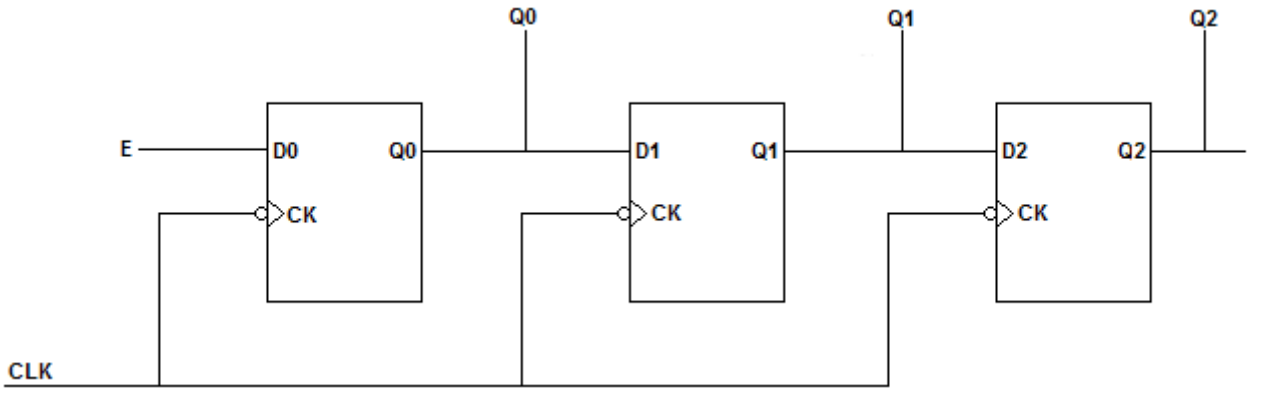


1 = ΔE, 2 = Δ^2E, 3 = Δ^3E e n = Δ^nE.

Onde **Δ** é o atraso.

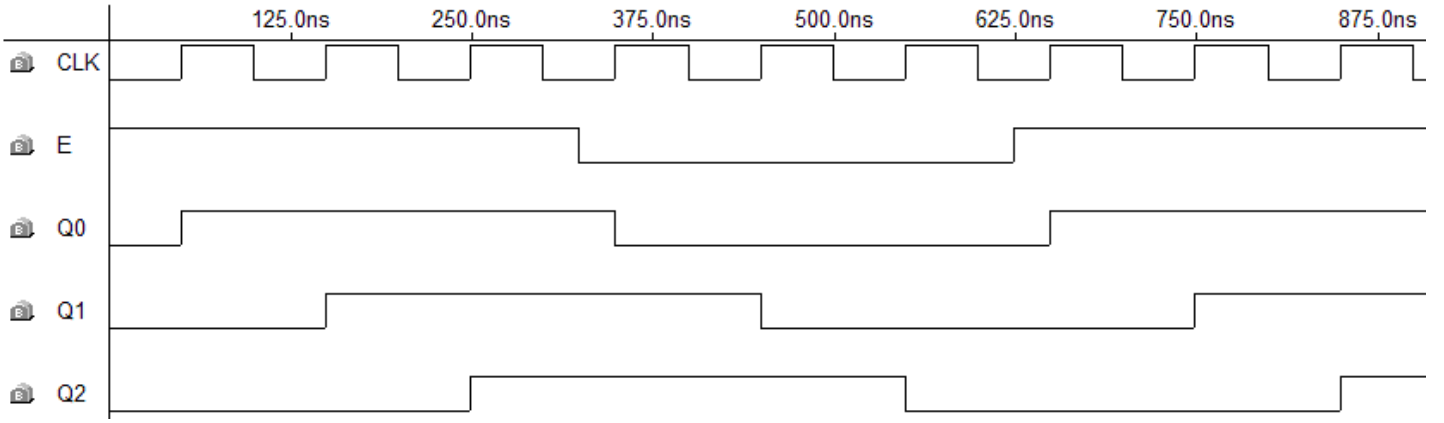
Para os circuitos sequenciais síncronos o atraso é de um período de relógio e pode ser demonstrado como a seguir.

2. Gerador de Atraso realizado com F/F do tipo D por não alterar o sinal de entrada.

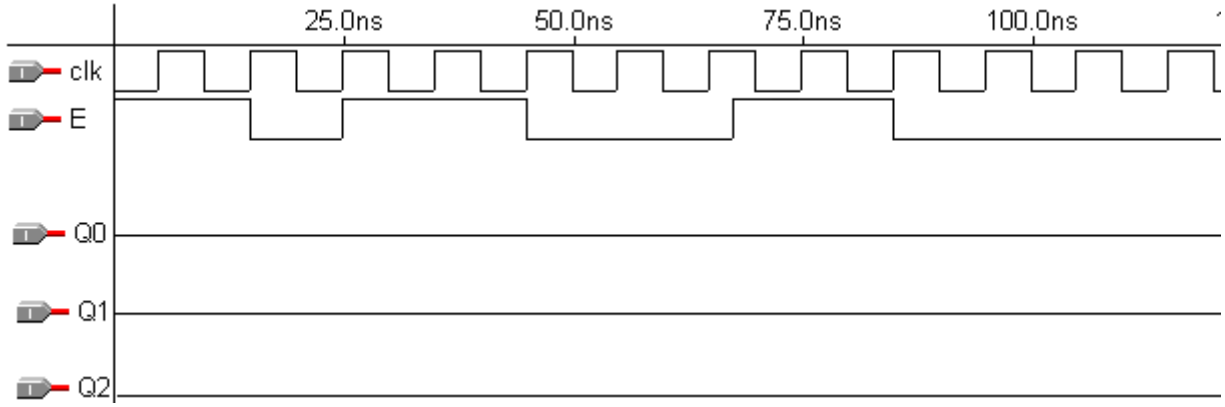


Q_0 = ΔE, Q_1 = Δ^2E e Q_3 = Δ^3E.

2.1 Formas de Ondas para o circuito acima.



Preencher as formas de ondas a seguir.



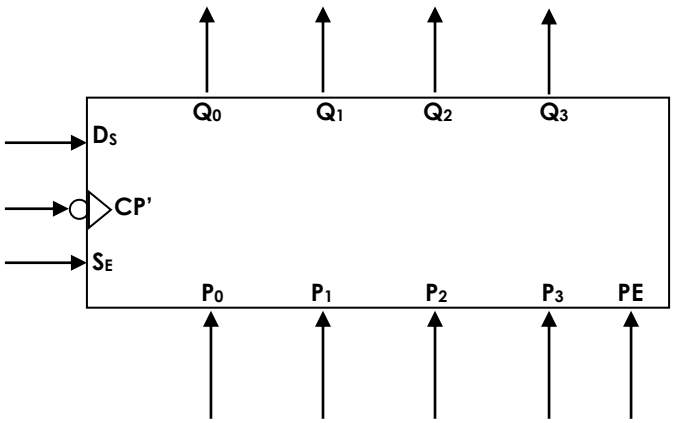
4. Para a entrada E no circuito e após 3 pulsos de relógio a entrada E com valor seqüencial 101, fica:

CLK	E	Q_0	Q_1	Q_2	Saída
-	1	0	0	0	0
↓	0	1	0	0	0
↓	1	0	1	0	0
↓	-	1	0	1	1

Saída = Q_2

5. Registrador de Deslocamentos com entrada serial e paralela, projeto de uma célula.

5.1 Configuração do registrador



S_E = Comando de entrada serial.
P_E = Comando de entrada paralela.
D_S = Entrada serial.
P_0 ... P_3 = Entradas paralelas.
Q_0 ...Q_3 = Saídas paralelas.
Q_3 = Saída serial.
CP = Clock, borda de descida.
5.2 Tabela da Verdade do registrador de deslocamento.

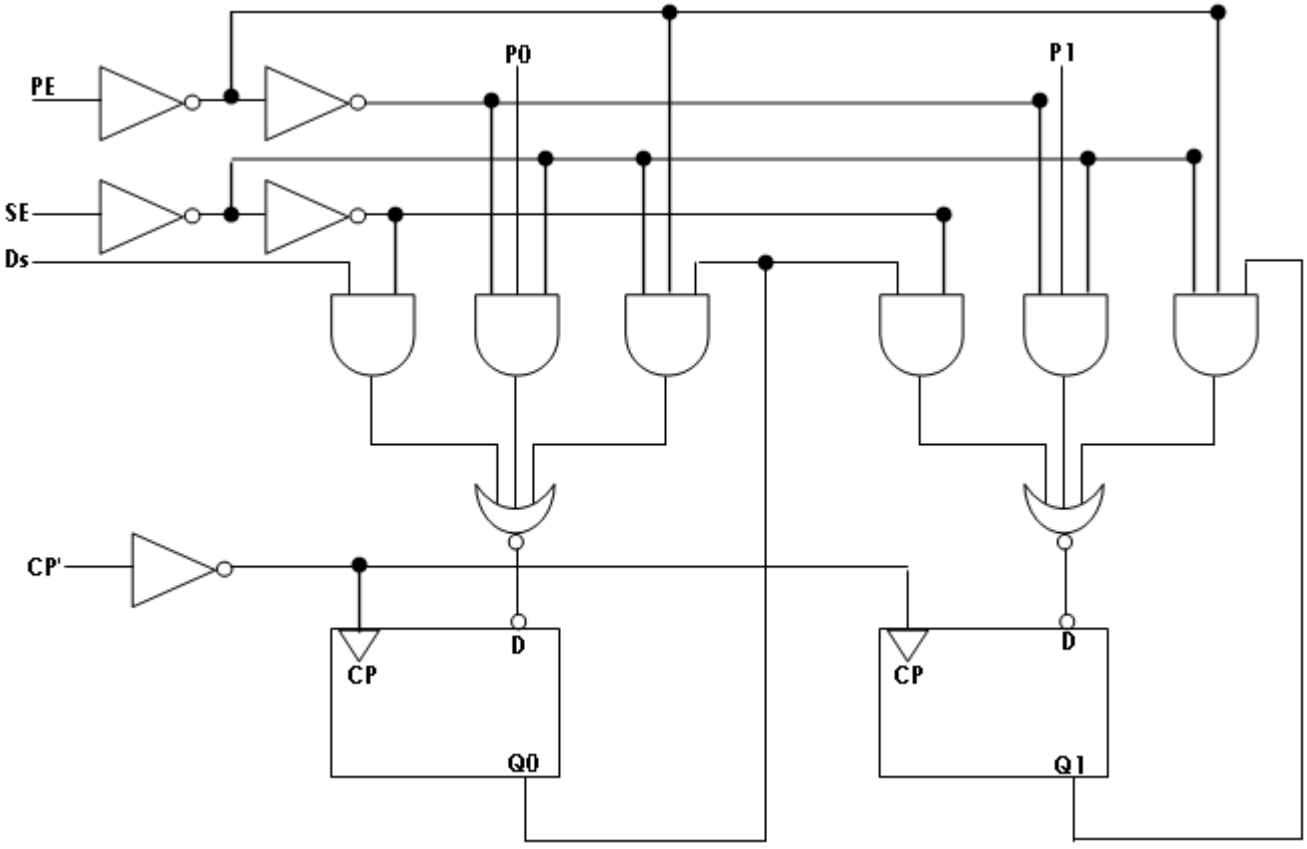
S_E	P_E	CLK	STATUS
1	X	↓	D →Q_0, Q_0 → Q_1,....,Q_2 → Q_3
0	1	↓	$Q_0 = P_0$, $Q_1 = P_1$,....., $Q_3 = P_3$
0	0	X	Q's não mudam

5.3 Implementação de uma célula de registrador.

$S_EP_EQ_0$	000	001	011	010	110	111	101	100
D_SP_0 00	0	1	0	0	0	0	0	0
01	0	1	1	1	0	0	0	0
11	0	1	1	1	1	1	1	1
10	0	1	0	0	1	1	1	1

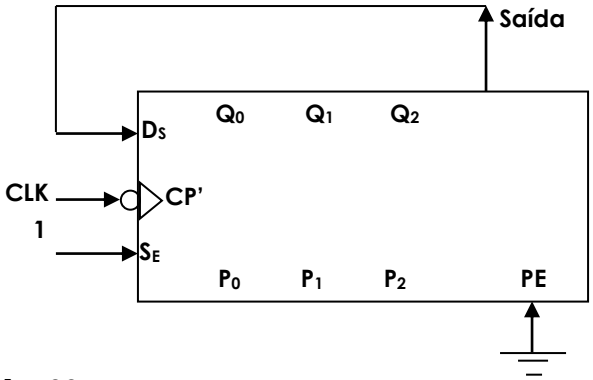
$D_0 = S_E'P_E'Q_0 + S_E'P_EP_0 + S_ED_S$

5.4 Circuito da célula do registrador



5.5 Aplicação com registrador para a confecção de contadores.

a) Em anel



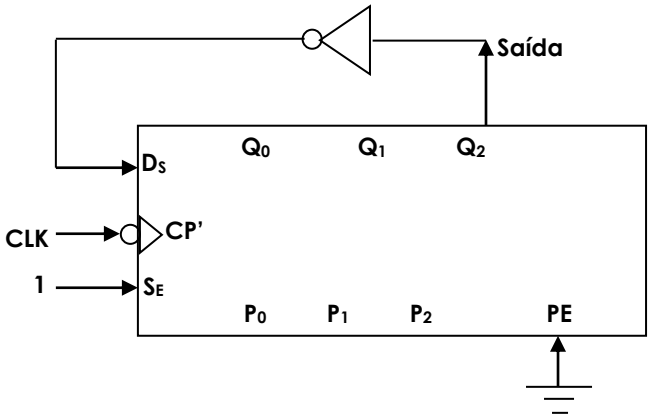
Estado Inicial = 001

A malha de estados percorrida pelo registrador de 3 bits, será:

001 - 010 - 100

Podemos concluir que a divisão de frequência desse contador para N bits é por: N.

b) Johnson



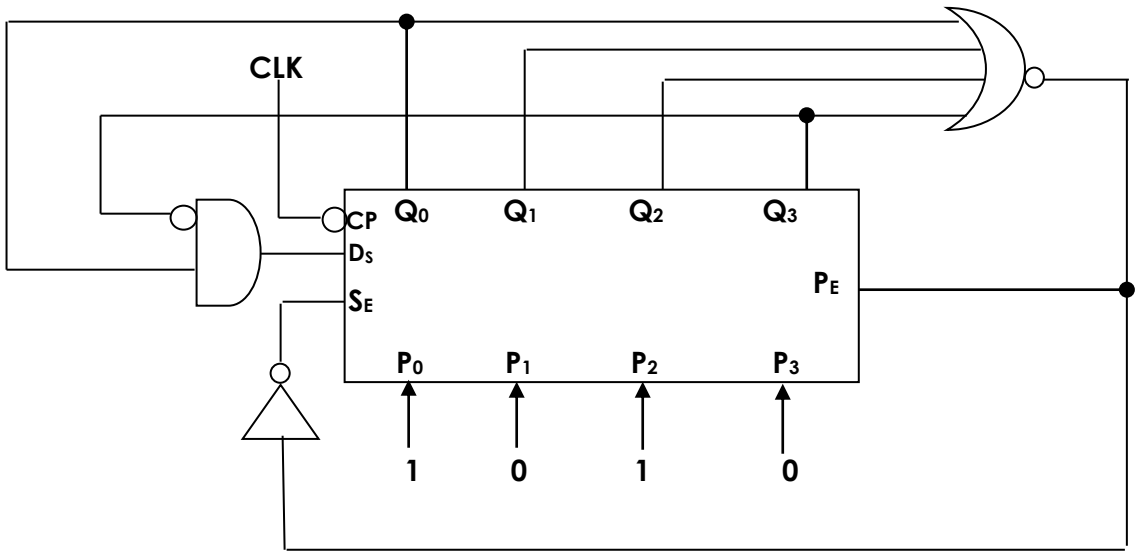
Estado Inicial = 000

A malha de estados percorrida pelo registrador de 3 bits, será:

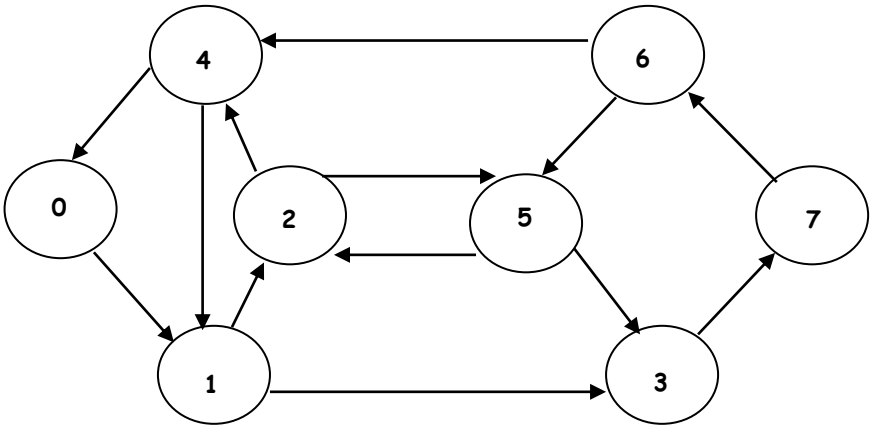
000 – 001 – 011 – 111 – 110 – 100

Podemos concluir que a divisão de frequência desse contador para N bits é por: 2N

Exemplo: Determinar os estados percorridos pela associação e a frequência de saída.



6. Diagrama de estados geral para o registrador de deslocamentos de 3 bits pela entrada serial D_S.



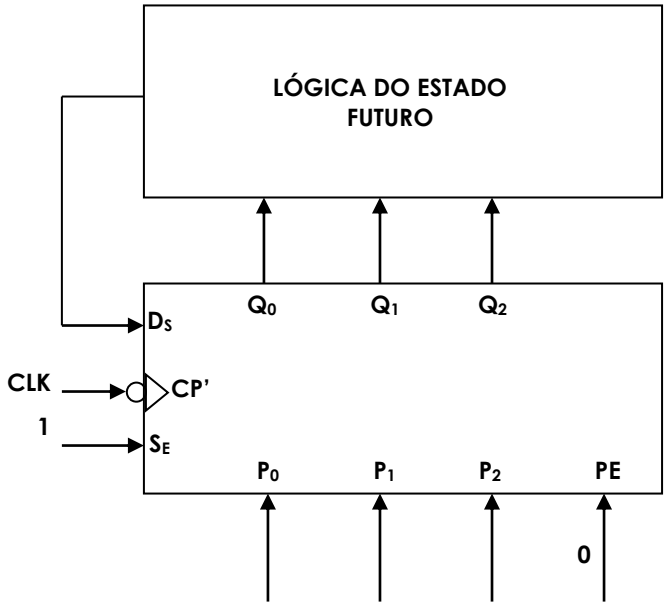
Regra:

1) $D_S = 0$, $Q_{n+1} = 2 \times Q_n$;

2) $D_S = 1$, $Q_{n+1} = 2 \times (Q_n + 1)$.

Exercício: Construir um contador módulo 5, cuja malha de estados é 0 – 1 – 3 – 6 – 4, usando o registrador de deslocamento entrada serial D_S, cuja saída será borda de descida e na passagem do estado 4 para o estado 0. Pede-se:
a) Representação esquemática do contador.
b) Implementação D_S e S (saída) lógica do estado futuro e saída.

a) Representação esquemática do sistema sequencial contador.



b) Mapa de Karnaugh para implementação da lógica do estado futuro.

Q_2Q_1	00	01	11	10
$Q_0$0	1	0	0	0
01	1	0	0	0

$D_S = Q_2'Q_1'$
c) $S = (Q_2Q_1'Q_0'clk)$

VHDL – Contador serial mod. 5, contador Johnson, em anel e registrador universal

1. VHDL – Cobtaodr serial modulo 5

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.NUMERIC_BIT.ALL;

ENTITY contador_serial IS
 PORT(clk, il : IN BIT;
 --d0: out BIT;
 sh: IN BIT_VECTOR(1 DOWNTO 0);
 P : IN BIT_VECTOR(2 DOWNTO 0);
 Q : OUT BIT_VECTOR(2 DOWNTO 0));
END contador_serial;

ARCHITECTURE beh OF contador_serial IS
 SIGNAL qtmp : BIT_VECTOR(2 DOWNTO 0);
 SIGNAL d0,ir: bit;
BEGIN
 PROCESS(clk)
 BEGIN
 d0 <= (not qtmp(2)) and (not qtmp(1));
 ir <= d0;
 IF (clk = '1' AND clk'EVENT) THEN
 CASE sh IS
 WHEN "00" => qtmp <= qtmp;
 WHEN "01" => qtmp <= P;
 WHEN "10" => qtmp<=qtmp(1 DOWNTO 0) & ir;
 WHEN "11" => qtmp<=il & qtmp(2 DOWNTO 1);
 WHEN OTHERS => NULL;
 END CASE;
 END IF;
 END PROCESS;
 Q <= qtmp;
 End beh;
```

2. VHDL - Contador Johnson de 4 bits.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE  IEEE.NUMERIC_STD.ALL;


ENTITY johnson_counter IS
PORT (
 clk     : IN STD_LOGIC;
 reset_n : IN STD_LOGIC;
 counter : OUT STD_LOGIC_VECTOR( 3 DOWNTO 0) );
END johnson_counter;

ARCHITECTURE Arch OF johnson_counter IS
```

```
SIGNAL s_counter : UNSIGNED(3 downto 0):=(others => '0');
BEGIN

counter <= STD_LOGIC_VECTOR(s_counter);

P_johnson_ctr : PROCESS(clk)
BEGIN
   IF (reset_n = '0') THEN
         s_counter <= (OTHERS => '0');
   ELSIF( rising_edge(clk) ) THEN
         s_counter(1) <= s_counter(0);
         s_counter(2) <= s_counter(1);
         s_counter(3) <= s_counter(2);
         s_counter(0) <= not s_counter(3);
   END IF;
END PROCESS P_johnson_ctr;
END Arch;
```

3. VHDL - Contador em anel

```
--Ring counter
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
entity ring_counter is
   Port ( clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          output : out  STD_LOGIC_VECTOR (3 downto 0));
end ring_counter;
architecture Behavioral of ring_counter is
signal temp : STD_LOGIC_VECTOR(3 downto 0):=(others => '0');
begin
process(clk)
begin
   if( clk'event and clk='1' ) then
      if (reset = '1') then
         temp <= (0=> '1', others => '0');
      else
         temp(1) <= temp(0);
         temp(2) <= temp(1);
         temp(3) <= temp(2);
         temp(0) <= temp(3);
      end if;
   end if;
end process;
output<=temp;
end Behavioral;
```

4. VHDL – Registrador Universal de 4 bits síncrono carga paralela e deslocamento esquerda/direita e vice-versa.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
```

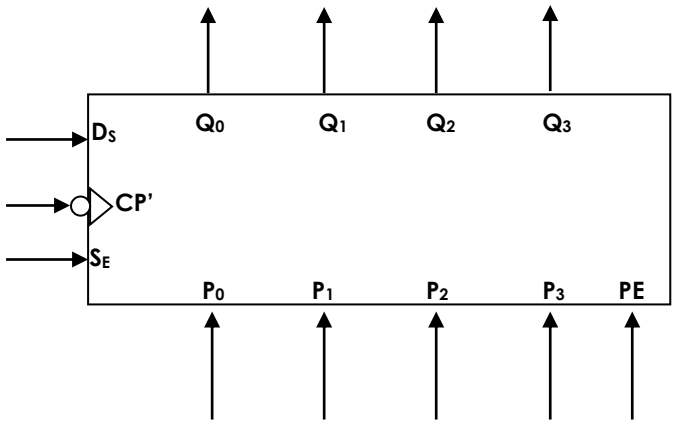
```
USE IEEE.NUMERIC_BIT.ALL;

ENTITY registrador_universal IS
PORT(clk, il, ir : IN BIT;
sh: IN BIT_VECTOR(1 DOWNTO 0);
P : IN BIT_VECTOR(3 DOWNTO 0);
Q : OUT BIT_VECTOR(3 DOWNTO 0));
END registrador_universal;

ARCHITECTURE beh OF registrador_universal IS
SIGNAL qtmp : BIT_VECTOR(3 DOWNTO 0);
        BEGIN
        PROCESS(clk)

BEGIN

        IF (clk = '1' AND clk'EVENT) THEN
 CASE sh IS
        WHEN "00" => qtmp <= qtmp;
        WHEN "01" => qtmp <= P;
        WHEN "10" => qtmp<=qtmp(2 DOWNTO 0) & ir;
        WHEN "11" => qtmp<=il & qtmp(3 DOWNTO 1);
        WHEN OTHERS => NULL;
 END CASE;
        END IF;
        END PROCESS;
 Q <= qtmp;
 End beh;
```

Para casa: Construir um contador módulo 8 usando o registrador de deslocamento no modo serial por D_S.

Módulo 08: Registradores de Deslocamentos.

1. Introdução: Implementação de um registrador de deslocamento com entradas paralelas. O subsistema digital registrador é o elemento mais importante dos dispositivos digitais. Sua flexibilidade na implementação de funções digitais credencia como universal, pois realiza funções de contagem, armazenamento temporário, deslocamentos a esquerda e a direita, conversões serie- paralelo e paralelo série entre outras.

a) Configuração do registrador



S_E = Comando de entrada serial.
P_E = Comando de entrada paralela.
D_S = Entrada serial.
P_0 ... P_3 = Entradas paralelas.
Q_0 ...Q_3 = Saídas paralelas.
Q_3 = Saída serial.
CP = Clock, borda de descida.

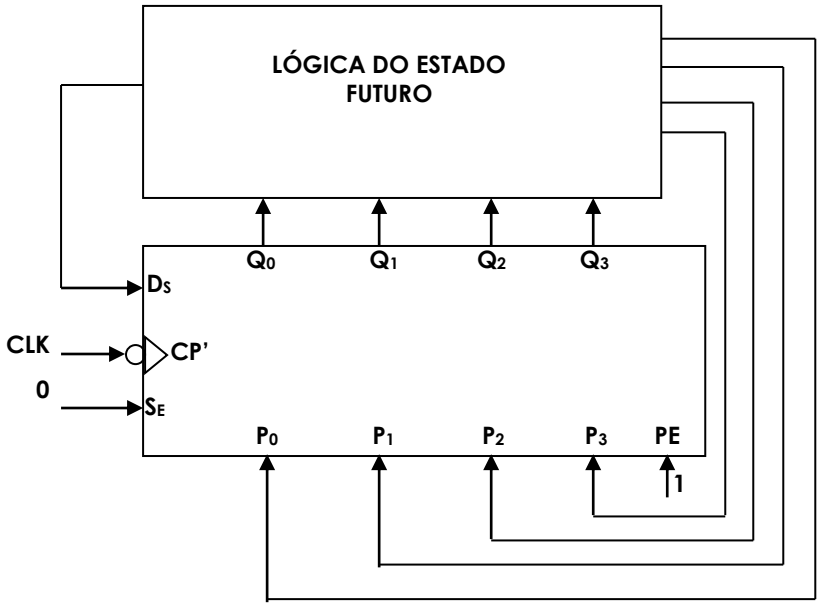
b) Tabela da Verdade do registrador de deslocamento.

S_E	P_E	CLK	STATUS
1	X	↓	$D \rightarrow Q_0$, $Q_0 \rightarrow Q_1$,....,$Q_2 \rightarrow Q_3$
0	1	↓	$Q_0 = P_0$, $Q_1 = P_1$,....., $Q_3 = P_3$
0	0	X	Q's não mudam

Exercício: Construir um contador módulo 10, cuja malha de estados é 0 – 1 – 2 - 3 – 4 – 5 – 6 – 7 – 8 - 9, usando o registrador de deslocamento entradas paralelas P_0 ... P_3, cuja saída será borda de descida e na passagem do estado 9 para o estado 0. Pede-se :

a) Representação esquemática do contador.
b) Implementação ds entradas paralelas e S (saída) lógica do estado futuro e saída.

a) Representação esquemática do sistema sequencial contador.



b) Mapa de Karnaugh para implementação da lógica do estado futuro.

Q_3Q_2 / Q_1Q_0	00	01	11	10
00	0	0	0	1
01	0	0	0	0
11	0	1	0	0
10	0	0	0	0

P_3 = Q3Q2'Q1'Q0' + Q3'Q2Q1Q0

Q_3Q_2 / Q_1Q_0	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	1	0	0	0
10	0	1	0	0

P_2 =Q3'Q2Q1' + Q3'Q2Q0' + Q3'Q2'Q1Q0

Q_3Q_2 / Q_1Q_0	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	0	0	0	0
10	1	1	0	0

P_1 = Q3'Q1'Q0 + Q3' Q1Q0'

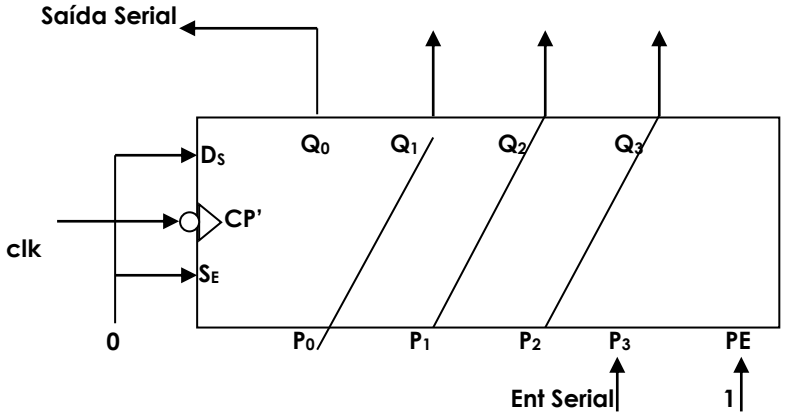
Q_3Q_2 / Q_1Q_0	00	01	11	10
00	1	1	0	1
01	0	0	0	0
11	0	0	0	0
10	1	1	0	0

P_0 =Q3'Q0' + Q2'Q1'Q0'

Exercício: Construir um contador módulo 10 usando o registrador de deslocamento no modo paralelo código Excesso 3.

Implementação de um registrador de deslocamento com entradas paralelas modo deslocamento da direita para a esquerda.

a) Configuração do registrador

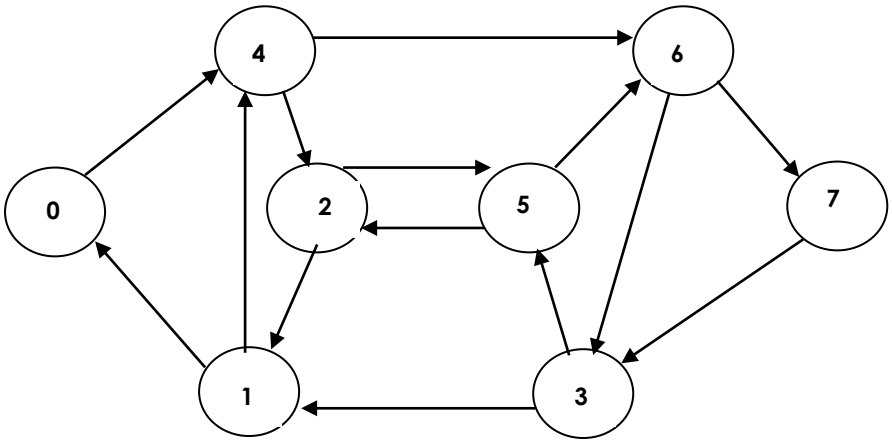


S_E = Comando de entrada serial.
P_E = Comando de entrada paralela.
D_S = Entrada serial.
P_0 ... P_3 = Entradas paralelas.
Q_0 ...Q_3 = Saídas paralelas.
Q_3 = Saída serial.
CP = Clock, borda de descida.

b) Tabela da Verdade do registrador de deslocamento.

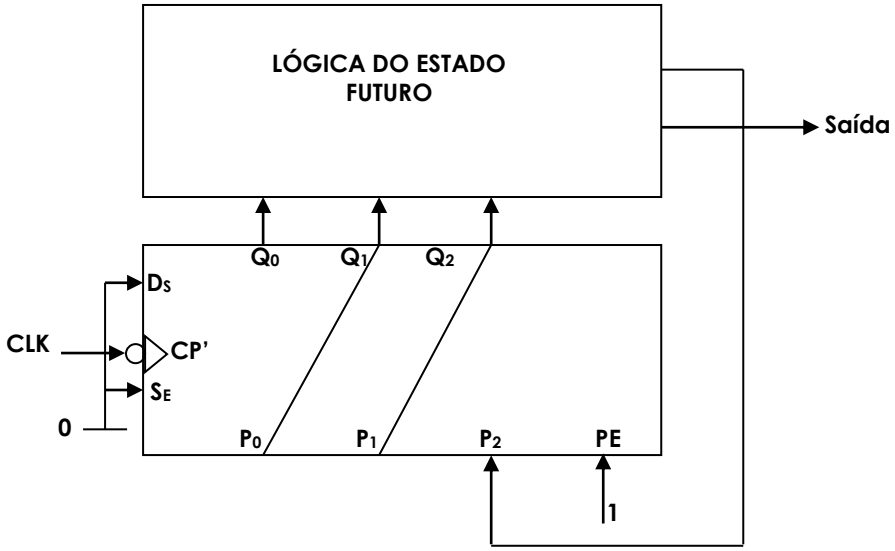
S_E	P_E	CLK	STATUS
1	X	↓	$D_S \rightarrow Q_0$, $Q_0 \rightarrow Q_1$,....,$Q_2 \rightarrow Q_3$
0	1	↓	$Q_0 = P_0$, $Q_1 = P_1$,....., $Q_3 = P_3$
0	0	X	Q's não mudam

c) Diagrama de estados geral do modo inverso.



Exercício: Construir um contador módulo 8, usando o registrador de deslocamento modo inverso, cuja saída será borda de descida e na passagem do estado 1 para o estado 0. Pede-se:
a) Representação esquemática do contador.
b) Implementação ds entradas paralelas e S (saída) lógica do estado futuro e saída.

a) Representação esquemática do sistema sequencial contador.



b) Mapa de Karnaugh para implementação da lógica do estado futuro.

Q_2Q_1	00	01	11	10
Q_0 0	1	0	0	1
1	0	0	0	0

$D_S = Q_1'Q_0'$

c) $S = Q_2'Q_1'Q_0clk$

Exercício: Construir um contador módulo 5 usando o registrador de deslocamento no modo serial por P_2.

VHDL – Contador serial modo inverso e contador paralelo.

1. VHDL – Contador serial invertido módulo 5.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
USE IEEE.NUMERIC_BIT.ALL;

ENTITY contador_reverso IS
 PORT(clk,ir : IN BIT;
 --d0: out BIT;
 sh: IN BIT_VECTOR(1 DOWNTO 0);
 P : IN BIT_VECTOR(2 DOWNTO 0);
 Q : OUT BIT_VECTOR(2 DOWNTO 0));
END contador_reverso;

ARCHITECTURE beh OF contador_reverso IS
 SIGNAL qtmp : BIT_VECTOR(2 DOWNTO 0);
 SIGNAL d0,il: bit;
BEGIN
 PROCESS(clk)
 BEGIN
 d0 <= (not qtmp(1)) and (not qtmp(0));
 il <= d0;
 IF (clk = '1' AND clk'EVENT) THEN
 CASE sh IS
 WHEN "00" => qtmp <= qtmp;
 WHEN "01" => qtmp <= P;
 WHEN "10" => qtmp<=qtmp(1 DOWNTO 0) & ir;
 WHEN "11" => qtmp<=il & qtmp(2 DOWNTO 1);
 WHEN OTHERS => NULL;
 END CASE;
 END IF;
 END PROCESS;
 Q <= qtmp;
 End beh;
```

2. Contador decimal BCD-8421 modo paralelo.

-- 4-bit Não assinalado contador paralelo.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity contador_paralelo is
 port(CLK, reset : in bit;
 load: in bit;
 --P : in std_logic_vector(3 downto 0);
 Q : out bit_vector(3 downto 0));
end contador_paralelo;
architecture archi of contador_paralelo is
```

```
 signal tmp: bit_vector(3 downto 0);
 signal p0,p1,p2,p3: bit;
begin
process (CLK)
 begin
        if (CLK'event and CLK='1') then
                if (reset = '1') then
-- tmp <= "0000";
                        elsif load = '0' then
                        p3 <= (tmp(3) and not tmp(2) and not tmp(1) and not tmp(0))or (not tmp(3) and tmp(2)
and tmp(1) and tmp(0));
                        p2 <= ((not tmp(3)and tmp(2) and not tmp(1)) or (not tmp(3) and tmp(2) and not tmp(0))
or (not tmp(3) and not tmp(2) and tmp(1) and tmp(0)));
                        p1 <= (not tmp(3) and not tmp(1) and tmp(0)) or (not tmp(3) and tmp(1) and not tmp(0));
                        p0 <= (not tmp(3) and not tmp(0)) or (not tmp(2) and not tmp(1) and not tmp(0));
                        tmp(0) <= p0;
                        tmp(1) <= p1;
                        tmp(2) <= p2;
                        tmp(3) <= p3;
                        end if;
                end if;
end process;
Q <= tmp;
end archi;
```