

MANUAL DO ALUNO

DISCIPLINA SISTEMAS DIGITAIS

Módulos 4 e 5

República Democrática de Timor-Leste
Ministério da Educação



FICHA TÉCNICA

TÍTULO

MANUAL DO ALUNO - DISCIPLINA DE SISTEMAS DIGITAIS
Módulos 4 e 5

AUTOR

JORGE FLÁVIO

COLABORAÇÃO DAS EQUIPAS TÉCNICAS TIMORENSES DA DISCIPLINA
XXXXXXX

COLABORAÇÃO TÉCNICA NA REVISÃO
XXXXXXXXXX

DESIGN E PAGINAÇÃO

UNDESIGN - JOAO PAULO VILHENA
EVOLUA.PT

IMPRESSÃO E ACABAMENTO
XXXXXX

ISBN

XXX - XXX - X - XXXXX - X

TIRAGEM

XXXXXXX EXEMPLARES

COORDENAÇÃO GERAL DO PROJETO
MINISTÉRIO DA EDUCAÇÃO DE TIMOR-LESTE
2013



Índice

Circuitos Sequenciais	7
Apresentação.....	8
Introdução	8
Objetivos de aprendizagem	8
Âmbito de conteúdos	9
Circuitos Sequenciais	10
Latch com portas NAND.....	13
SET do Latch (FF)	14
RESET do Latch (FF).....	15
SET e RESET Ativos Simultaneamente.....	15
Representações Alternativas	16
Terminologia	17
Latch com portas NOR	19
Estado do flip-flop quando a alimentação é ligada.....	21
Estudo de casos em pesquisa de falhas.....	22
Sinais de CLOCK e Flip-Flops	23
Flip-Flops com CLOCK	24
Tempos de Setup (Preparação) e Hold (Manutenção).....	25
Flip-Flop S-C ou S-R com CLOCK	27
Circuito interno de um Flip-Flop S-C (S-R) disparado por transição	29
Flip-Flop J-K	31
Circuito interno de um Flip-Flop J-K disparado por transição.....	33
Flip-Flop tipo D.....	35
Implementação de um Flip-Flop D	36
Transferência de dados em Paralelo	37
Latch D (Latch transparente).....	38
Entradas Assíncronas.....	40
Designação para as entradas Assíncronas	41
Considerações sobre temporização em Flip-Flops	44
Tempos de Setup e Hold	44

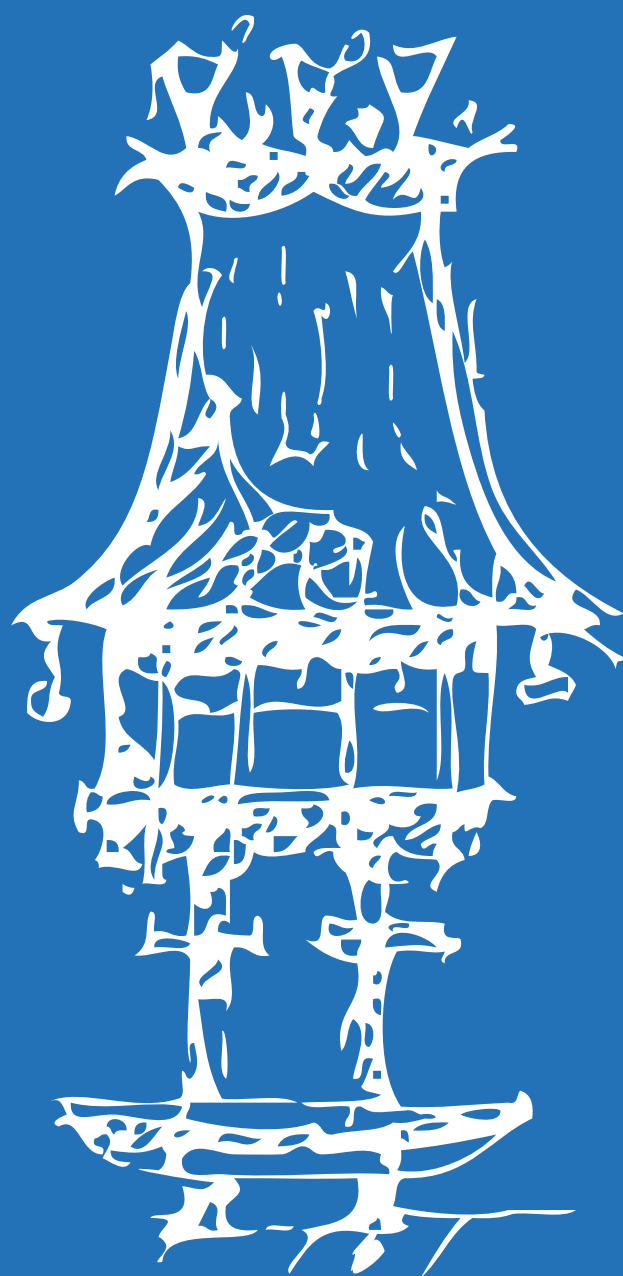



Atrasos de propagação	44
Frequência máxima de CLOCK, fMAX	45
Tempos de duração em ALTO e BAIXO do Sinal de CLOCK	45
Largura dos pulsos Assíncronos	46
Tempos de transição do CLOCK	46
Cl's Reais.....	46
Divisão de Frequência e Contagem	49
Operação de contagem	50
Diagrama de transição de estados.....	51
Módulo do Contador	52
Dispositivos Schmitt-Trigger.....	54
Bibliografia	57
Memórias.....	59
Apresentação.....	60
Introdução	60
Âmbito de conteúdos	61
Introdução.....	62
Lógica programável / Lógica tradicional	64
Vantagens	64
Evolução de Sistemas de Hardware	65
Memórias PROM	66
Os Dispositivos Lógicos Programáveis (PLD).....	68
Arranjos Lógicos Programáveis	70
PLA.....	70
Dispositivo PAL.....	72
Estrutura interna	72
Implementação de funções lógicas com PAL.....	73
PAL Comerciais.....	75
Arquitetura da PAL® combinatória 16L8.....	76
Arquitetura de PAL® sequencial PAL16R8.....	79
Arquitetura de PAL® sequencial ATF22V10.....	81
Arquitetura de PAL® sequencial ATF750C/CL.....	85



Saída sequencial	91
Programação e teste de um circuito utilizando uma PAL	94
Configuração da PAL® ATF22V10 usando CUPL	94
Configuração da PAL® ATF750C/CL usando CUPL	95
Estrutura de um programa em PALASM	101
Ficheiro de especificação.....	103
Segmento de declarações.....	104
Segmento de equações booleanas	105
Segmentos de máquina de estados e condições	105
Segmento de simulação.....	108
Exemplo de aplicação	109
Bibliografia	120







Circuitos Sequenciais

Módulo 4

Apresentação

Este módulo tem carácter teórico-prático, devendo decorrer essencialmente em ambiente laboratorial de modo que os alunos possam ensaiar e comprovar as características e funcionamento dos Circuitos Sequenciais estudados na teoria.

Introdução

A abordagem deste módulo de Circuitos Sequenciais leva-nos a uma melhor compreensão do funcionamento de vários tipos de aparelhos, que incorporam circuitos que utilizam estas características, existentes no mercado assim como a melhor escolha deste tipo de equipamentos para que se ajuste às crescentes evoluções disponíveis pelas diversas marcas.

Este módulo requer um conhecimento básico de matemática e análise de circuitos eletrónicos assim como a respetiva compreensão desses circuitos.

Objetivos de aprendizagem

- Flip-Flops (Biestáveis):
 - Distinguir circuito sequencial de circuito combinatório.
 - Compreender o funcionamento do FF com portas lógicas NAND e/ou NOR.
 - Representar o FF pela sua tabela da verdade e diagrama temporal.
 - Reconhecer biestáveis síncronos e assíncronos.
 - Identificar os biestáveis pelos seus símbolos.
 - Descrever o funcionamento de circuitos sequenciais através de diagramas de estado.
- Contadores e divisores de frequência:
 - Conhecer os vários tipos de contadores, as suas características e funcionamento.
 - Implementar um contador a partir da sua tabela da verdade.
 - Utilizar contadores como divisores de frequência.



- Registos de deslocamento:
 - Compreender o princípio de funcionamento de um registo de deslocamento, as suas características e aplicações.
 - Conhecer os diferentes modos de funcionamento de um registo de deslocamento quanto à entrada/saída de dados.
 - Identificar os registos de deslocamento quanto ao modo de deslocamento (à direita e à esquerda).

Âmbito de conteúdos

- Flip-Flops (Biestáveis).
- Registos de deslocamento.
- Contadores e divisores de frequência.



Circuitos Sequenciais

Os circuitos lógicos estudados até agora foram os circuitos combinatórios, nos quais as saídas, em qualquer instante de tempo, dependiam dos níveis presentes nas entradas no instante considerado. Quaisquer condições de entrada anteriores não tinham nenhum efeito nas saídas atuais, porque os circuitos lógicos combinatórios não tem memória. A maioria dos sistemas digitais é composta tanto de circuitos combinatórios como de elementos de memória.

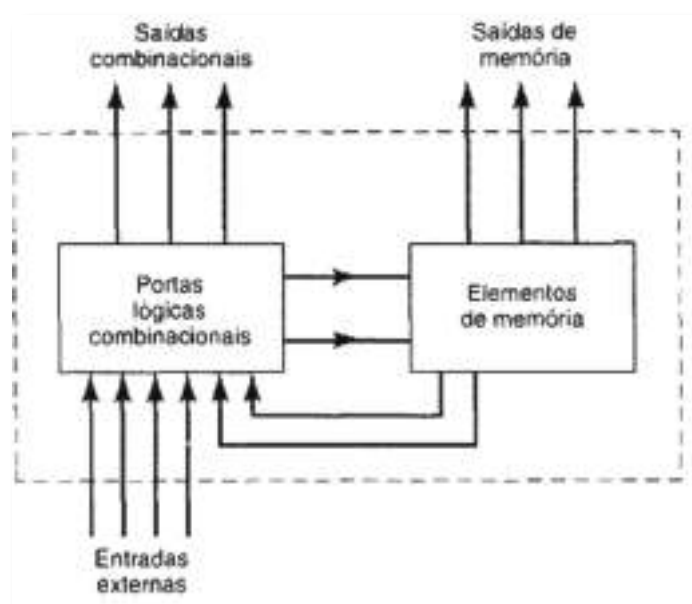


Fig. 1: Diagrama geral de um sistema digital.

A Fig.1 mostra um diagrama de blocos de um sistema digital geral que reúne portas lógicas combinatórias com dispositivos de memória. A parte combinatória aceita sinais lógicos de entradas externas e das saídas dos elementos de memória. O circuito combinatório opera sobre estas entradas para produzir várias saídas, algumas das quais são usadas para determinar os valores binários a serem armazenados nos elementos de memória. As saídas de alguns elementos de memória, por outro lado, vão para as entradas das portas lógicas dos circuitos combinatórios.

Este processo indica que as saídas externas de um sistema digital são uma função das entradas externas e das informações armazenadas nos seus elementos de memória.



O elemento de memória mais importante é o flip-flop, que é feito de uma configuração de portas lógicas. Embora uma porta lógica, por si só, não tenha capacidade de armazenamento, várias portas podem ser ligadas de modo que se permita que a informação seja armazenada. Muitas interconexões diferentes de portas são usadas para produzir flip-flops (abreviado como FFs).

A Fig2 (a) mostra o símbolo utilizado para um flip-flop genérico. Possui duas saídas, identificadas como Q e \bar{Q} que são opostas entre si. Q/\bar{Q} são as designações mais comuns usadas para as saídas dos FFs. Por vezes, utilizamos outras designações, tais como X/\bar{X} e A/\bar{A} , por conveniência, na identificação de FFs diferentes num circuito lógico.

A saída Q é chamada de saída normal do FF, e \bar{Q} é a saída invertida do FF. Sempre que nos referimos ao estado de um FF, estamos a referir ao estado da sua saída normal (Q). Fica subentendido que sua saída invertida (\bar{Q}) está no estado oposto. Por exemplo, se dissermos que um FF está no estado ALTO (1), significa que a saída $Q = 1$; se dissermos que um FF está no estado BAIXO (0), significa que a saída $Q = 0$. Obviamente, o estado de \bar{Q} é sempre o inverso de Q .

Os dois estados possíveis de operação para um FF estão resumidos na Fig.2 (b). Note que o estado ALTO ou 1 ($Q = 1 / \bar{Q} = 0$) também é chamado de estado SET. Sempre que as entradas de um FF fazem a saída passar para o estado $Q = 1$, denominamos isto de SET do FF. Analogamente, o estado BAIXO ou 0 ($Q = 0 / \bar{Q} = 1$) também é chamado de estado CLEAR ou RESET. Sempre que as entradas de um FF fazem a saída ir para o estado $Q = 0$, denominamos isto de RESET do FF. Mais à frente vamos ver que, muitos FFs tem uma entrada SET e/ou uma entrada RESET que são usadas para colocar o FF num determinado estado de saída.

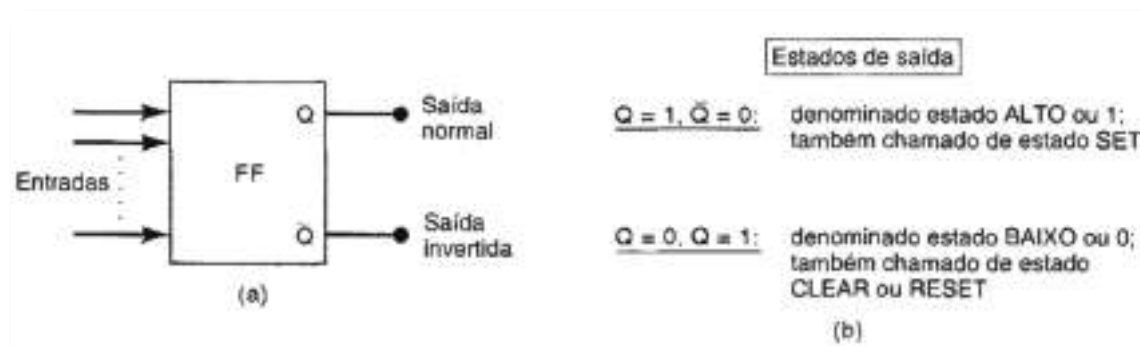


Fig. 2: Símbolo de um flip-flop genérico e definição dos dois estados de saída possíveis



De acordo com o símbolo na Fig.2 (a), um FF pode ter uma ou mais entradas. Estas entradas são utilizadas para causar a comutação do FF entre os seus possíveis estados de saída. Vamos descobrir que a maioria das entradas do FF precisam apenas de ser momentaneamente ativadas (pulsadas) de modo que provoquem uma mudança no estado da saída do FF, e a saída permanecerá neste novo estado mesmo após o pulso de entrada terminar. Esta é a propriedade de memória do FF.

O flip-flop é conhecido por outros nomes, incluindo «latch» e «multivibrador biestavel». O termo latch é usado para certos tipos de flip-flops que descreveremos. O termo multivibrador biestavel é o nome mais técnico para um flip-flop, mas é muito complexo para ser utilizado regularmente.



Latch com portas NAND

O circuito de FF mais básico pode ser construído com duas portas NAND ou com duas portas NOR. A versão com NAND, chamada de latch com portas NAND ou simplesmente latch, é mostrada na Fig.3 (a). As duas portas NAND são interligadas de modo cruzado, sendo que a saída da NAND-1 é ligada a uma das entradas da NAND-2, e vice-versa. As saídas das portas, identificadas como Q e \bar{Q} , respectivamente, são as saídas do latch. Sob condições normais, elas são sempre o inverso uma da outra. Existem duas entradas do latch: a entrada SET é a que faz um SET a Q para o estado 1; a entrada RESET (CLEAR) é a que limpa Q para o estado 0.

Ambas as entradas SET e RESET estão normalmente em estado ALTO, e uma delas é pulsada em BAIXO sempre que se deseja alterar as saídas do latch. Vamos começar a nossa análise mostrando que existem dois estados de saídas similares quando SET = RESET = 1. Uma possibilidade é apresentada na Fig.3 (a), onde $Q = 0$ e $\bar{Q} = 1$. Com $Q = 0$, as entradas da NAND-2 são 0 e 1, o que causa $Q = 1$. O nível 1 de Q faz com que NAND-1 tenha nível ALTO em ambas as entradas, o que resulta em 0 na saída Q . Como efeito, o que temos é um nível BAIXO na saída de NAND-1 produzindo um nível ALTO na saída de NAND-2, que por sua vez mantém a saída de NAND-1 em BAIXO.

A segunda possibilidade é mostrada na Fig.3 (b), onde $Q = 1$ e $\bar{Q} = 0$. O nível ALTO da porta NAND-1 provoca um nível BAIXO na saída da NAND-2, o que por sua vez mantém a saída da NAND-1 em ALTO. Portanto, existem dois estados de saída possíveis quando SET = RESET = 1, e, como vamos ver em breve, o estado real depende do que ocorreu previamente com as entradas.



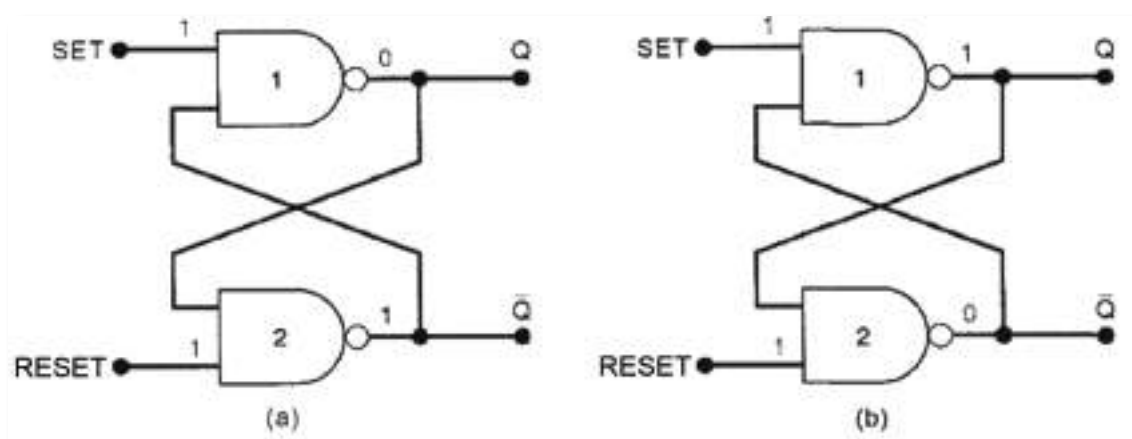


Fig. 3: Um latch NAND tem dois estados de repouso possíveis quando $SET = RESET = 1$

SET do Latch (FF)

Vamos investigar agora o que acontece quando a entrada SET é momentaneamente pulsada em BAIXO, enquanto o RESET é mantido em ALTO. A Fig.4 (a) mostra o que acontece quando $Q = 0$ antes da ocorrência do pulso. Quando SET é pulsado BAIXO em t_0 , Q vai para ALTO, e este nível ALTO força \bar{Q} a ir para BAIXO, de modo que agora a NAND-1 tem duas entradas em BAIXO. Logo, quando SET retorna para o estado 1 em t_1 a saída de NAND-1 permanece ALTO, o que por sua vez mantém a saída de NAND-2 em BAIXO. A Fig.4 (b) ilustra o que acontece quando $Q = 1$ e $\bar{Q} = 0$ antes da aplicação do pulso em SET. Visto que a saída $\bar{Q} = 0$ já está a manter a saída da NAND-1 em ALTO, o pulso BAIXO em SET não altera nada. Assim, quando SET retoma para ALTO, a saída do latch ainda esta com $Q=1$ e $\bar{Q}=0$.

Pode-se resumir a Fig.4 considerando-se que um pulso BAIXO na entrada SET leva sempre o latch para o estado onde $Q = 1$. Esta operação é denominada por SET do latch ou FF.

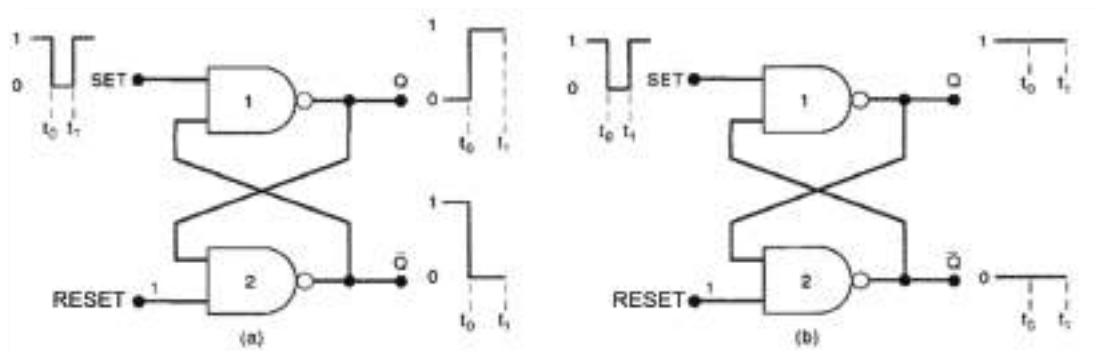


Fig. 4: Pulsando-se a entrada SET para o estado 0 quando (a) $Q = 0$ antes do pulso em SET; (b) $Q = 1$ antes do pulso em SET. Note que em ambos os casos Q acaba em ALTO



RESET do Latch (FF)

Vamos considerar agora o que acontece quando a entrada RESET é pulsada em BAIXO enquanto SET é mantido em ALTO. A Fig.5 (a) mostra o que acontece quando $Q = 0$ e $\bar{Q} = 1$ antes da aplicação do pulso. Como $Q = 0$ já está a manter a saída da NAND-2 em ALTO, o pulso em BAIXO no RESET não tem efeito nenhum. Quando RESET voltar para ALTO, as saídas do latch ainda serão $Q = 0$ e $\bar{Q} = 1$.

A Fig.5 (b) ilustra a situação onde $Q = 1$ antes da ocorrência do pulso no RESET. Assim que o RESET vai para BAIXO em t_0 , Q vai para ALTO, e este nível ALTO força Q a ir para BAIXO, de modo que NAND-2 agora tem as duas entradas em BAIXO. Portanto, quando o RESET retoma para ALTO em t_1 , a saída da NAND-2 permanece em ALTO, o que por sua vez mantém a saída da NAND-1 em BAIXO.

A Fig.5 pode ser resumida considerando-se que um pulso BAIXO na entrada RESET leva sempre o latch para o estado onde $Q = 0$. Esta operação é denominada de RESET do latch.

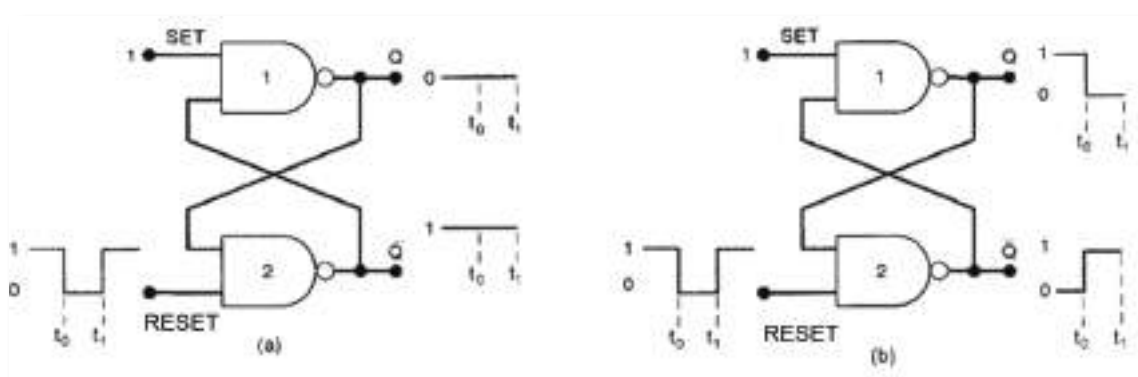


Fig. 5: Pulsando-se a entrada RESET para o estado BAIXO quando (a) $Q = 0$ antes do pulso em RESET; (b) $Q=1$ antes do pulso em RESET. Em ambos os casos, Q termina em baixo

SET e RESET Ativos Simultaneamente

O último caso a ser considerado é aquele em que as entradas SET e RESET são simultaneamente pulsadas em BAIXO. Isto produz um nível ALTO nas saídas das duas portas NAND, de modo que $Q = \bar{Q} = 1$. É claro que isto é uma condição indesejada, visto que as duas saídas supostamente são complementares entre si. Além disso, quando as entradas SET e RESET voltam para ALTO, o estado de saída resultante dependerá da



entrada que voltar a ALTO primeiro. Transições simultâneas de volta para 1 produzirão resultados imprevisíveis. Por estas razões, a condição $SET = RESET = 0$ normalmente não é utilizada para o latch NAND.

Resumo do Latch NAND

A operação descrita anteriormente pode ser convenientemente colocada numa tabela de verdade (Fig.6) que é resumida a seguir:

1. $SET = RESET = 1$. Esta condição é o estado normal de repouso e não tem nenhum efeito sobre o estado de saída.

As saídas Q e \bar{Q} permanecerão com os mesmos valores que estavam antes desta condição de entrada.

2. $SET = 0, RESET = 1$. Esta conjuntura faz a saída ir para o estado no qual $Q = 1$, onde permanecerá mesmo após o SET voltar para ALTO. Isto é denominado por SET do latch.

3. $SET = 1, RESET = 0$. Esta Situação provoca o estado $Q = 0$, onde a saída permanecerá mesmo após o RESET voltar para ALTO. Isto é denominado RESET do latch.

4. $SET = RESET = 0$. Esta condição tenta fazer um SET e um RESET do latch ao mesmo tempo e pode produzir resultados duvidosos. Não deve ser usada.

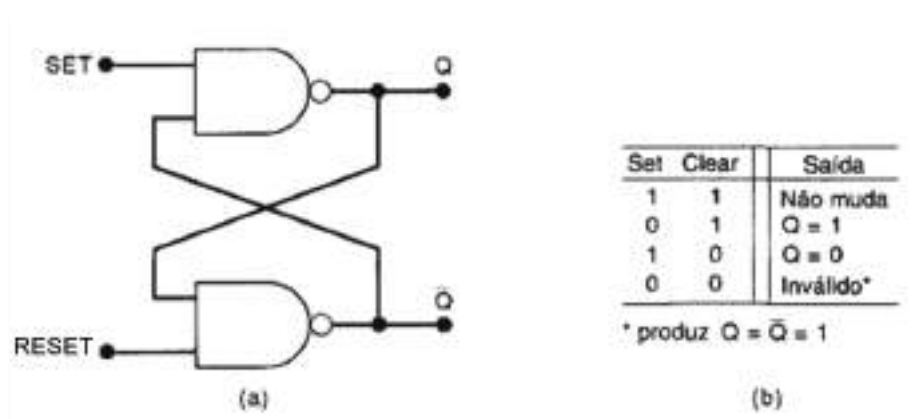


Fig. 6: (a) Latch NAND; (b) tabela de verdade.

Representações Alternativas

Considerando a descrição da operação do latch NAND, deve ficar claro que as entradas de SET e RESET são ativas em BAIXO. A entrada SET faz $Q = 1$ quando SET vai para



BAIXO, e a entrada RESET faz $Q = 0$ quando RESET vai para BAIXO. Por causa disto, o latch NAND frequentemente é desenhado usando-se a representação alternativa para cada porta NAND, conforme mostra a Fig.7 (a). As «bolhas» nas entradas, assim como a identificação dos sinais como *SET* e *RESET*, indicam o estado de acionamento BAIXO para estas entradas.

A Fig.7 (b) mostra uma representação simplificada que vamos usar algumas vezes. As letras *S* e *R* representam as entradas SET e RESET, enquanto as bolhas indicam que as entradas são ativas em nível BAIXO. Sempre que este símbolo for usado, ele representa um latch NAND.

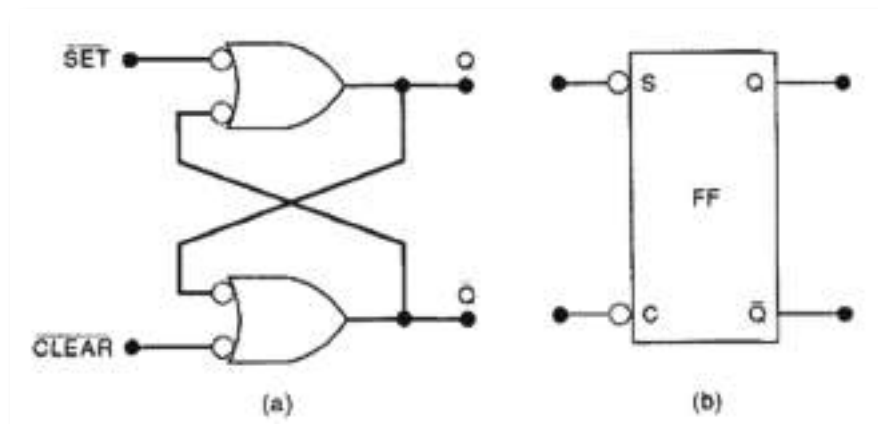


Fig. 7: (a) Representação equivalente para o latch NAND; (b) símbolo simplificado

Terminologia

A ação de RESET um FF ou latch também é chamada de limpar; e ambos os termos são usados indistintamente na área digital. Na verdade, a entrada RESET também pode ser chamada de CLEAR (C), e um latch SET-RESET pode ser denominado latch SET-CLEAR.

Exercício 1:

As formas de onda da Fig.8 são aplicadas nas entradas do latch da Fig.7. Considere que inicialmente $Q = 0$ e determine a forma de onda de Q .



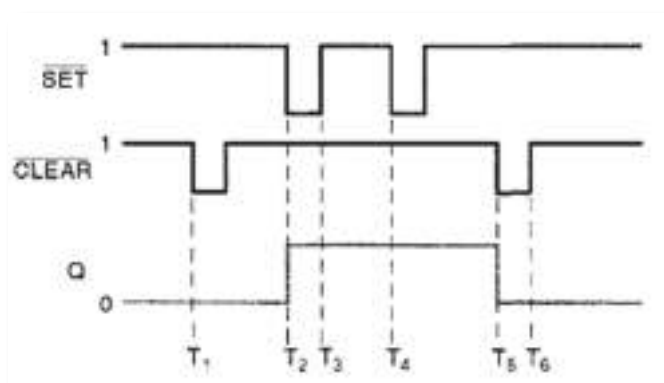


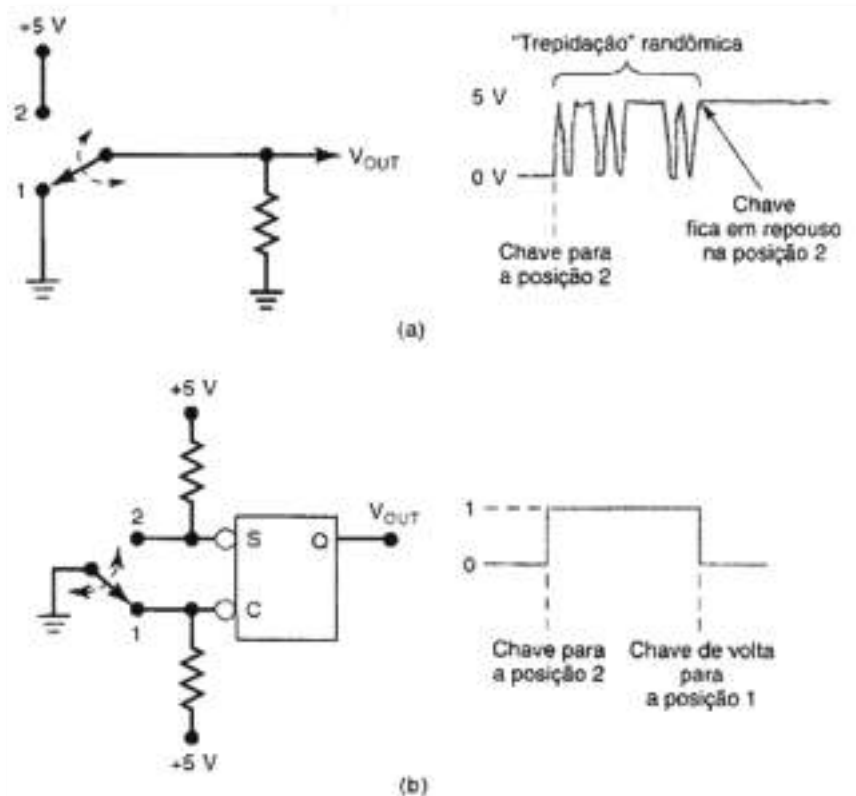
Fig. 8: Exemplo 1

Exercício 2:

É praticamente impossível obter uma transição de tensão «limpa» com um interruptor mecânico, por causa do fenómeno conhecido como trepidação de contacto (contact bounce). Este fenómeno é ilustrado na Fig.9 (a), onde a ação de mover o interruptor da posição de contacto 1 para a 2 produz muitas transições na tensão na saída, conforme o interruptor trepida (faz e interrompe o contacto com 2 muitas vezes) antes de parar sobre o contacto 2.

As múltiplas transições no sinal de saída geralmente não duram mais do que uns poucos milissegundos, mas podem ser inaceitáveis em muitas aplicações. Um latch NAND pode ser usado para evitar que a presença da trepidação de contacto afete a saída. Descreva a operação do circuito da Fig.9 (b) utilizado para eliminar os efeitos da trepidação de contacto.

Fig. 9: (a) Trepidação mecânica do contacto produz múltiplas transições; (b) latch NAND usado para eliminar as múltiplas transições.



Latch com portas NOR

Duas portas NOR interligadas de modo cruzado podem ser usadas como um latch com portas NOR. A configuração apresentada na Fig.10 (a) é similar ao latch NAND, exceto que as saídas Q e \bar{Q} estão em posições trocadas.

A análise da operação do latch NOR pode ser feita exatamente do mesmo modo que aquela feita para o latch NAND. Os resultados estão apresentados na tabela de verdade da Fig.10 (b) e podem ser resumidos como se segue:

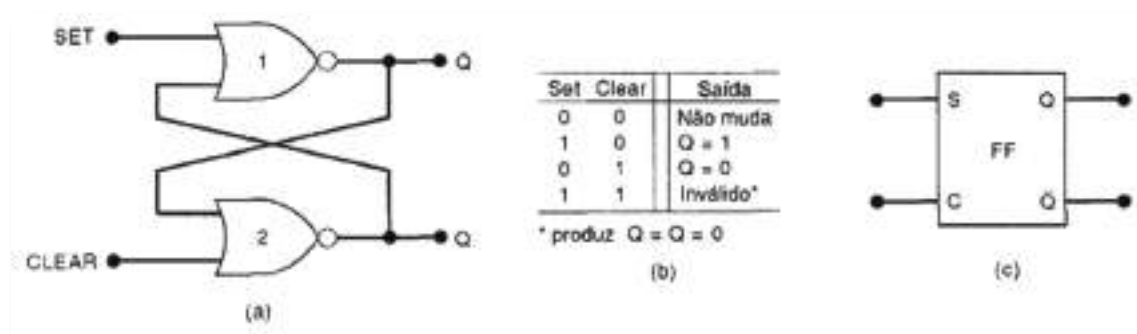


Fig. 10: (a) Latch com portas NOR; (b) tabela de verdade; (c) símbolo simplificado.

1. SET = CLEAR = 0. Este é o estado de repouso normal para o latch NOR, e não provoca nenhum efeito no estado de saída. Q e \bar{Q} permanecem com os mesmos valores que estavam antes da ocorrência desta condição de entrada.
2. SET = 1, CLEAR = 0. Esta condição leva a $Q = 1$, onde permanece mesmo após SET voltar a 0.
- 3- SET = 0, CLEAR = 1. Esta situação limpa $Q = 0$, onde permanece mesmo após CLEAR voltar a 0.
4. SET = CLEAR = 1. Esta condição tenta fazer um SET e um CLEAR (RESET) do latch ao mesmo tempo, e produz $Q = \bar{Q} = 0$. Se as entradas retornam a 0 simultaneamente, o estado de saída resultante é imprevisível. Esta condição de entrada não deve ser usada. O latch com portas NOR opera exatamente como um latch com portas NAND, exceto que as entradas SET e CLEAR são ativas em ALTO em vez de ativas em BAIXO e que o estado normal de repouso é SET = CLEAR = 0. Q é levado ao nível ALTO por um pulso ALTO na entrada SET e é colocado em BAIXO por um pulso ALTO na entrada CLEAR. O símbolo



simplificado para o latch NOR na Fig.10 (c) é apresentado sem as «bolhas» nas entradas S e Q e isto indica que estas entradas são ativas em ALTO.

Exercício 1:

Suponha que inicialmente $Q = 0$, e determine a forma de onda de Q para as entradas da latch NOR da Fig.11.

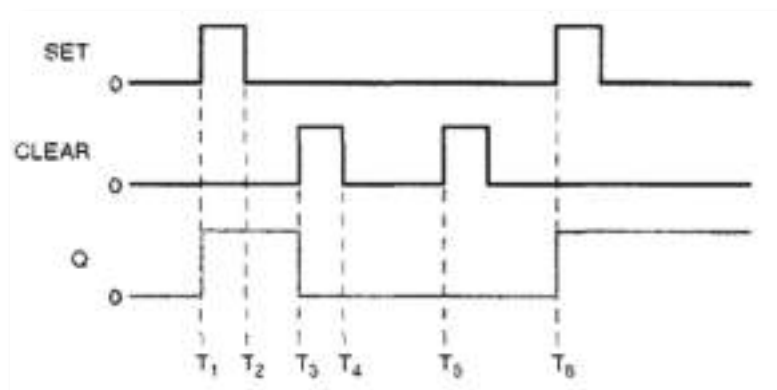


Fig. 11: Exemplo 1

Exercício 2:

A Fig.12 mostra um circuito simples que pode ser usado para detetar a interrupção de um feixe de luz. A luz é focada num foto transístor, que está ligado na configuração emissor comum para operar como um interruptor.

Suponha que o latch foi previamente limpo para o estado 0 abrindo-se o interruptor SW1 momentaneamente, descreva o que acontece se o feixe de luz for momentaneamente interrompido.

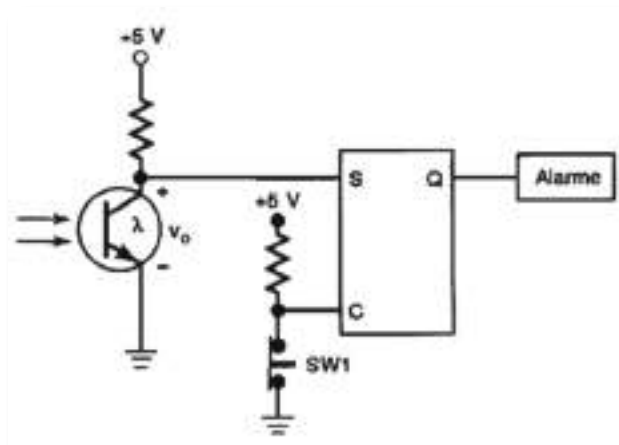


Fig. 12: Exemplo 2



Estado do flip-flop quando a alimentação é ligada

Quando ligamos a fonte de alimentação de um circuito, não é possível prever o estado inicial de uma saída de um flipflop se as entradas SET e CLEAR estiverem nos seus estados inativos (por exemplo, $S = C = 1$ para um latch NAND, $S = C = 0$ para um latch NOR). Existem hipóteses iguais de o estado inicial ser $Q = 0$ ou $Q = 1$. O estado inicial vai depender de factores tais como: atrasos de propagação internos, capacitâncias parasitas e carregamento externo. Se um latch, ou um FF, deve começar num determinado estado para garantir a correta operação de um circuito, então ele deve ser colocado neste estado ativando-se momentaneamente a entrada SET ou CLEAR no início da operação do circuito.

Frequentemente isto é conseguido pela aplicação de um impulso na entrada apropriada.



Estudo de casos em pesquisa de falhas

Os dois exercícios a seguir mostrarão o tipo de raciocínio que é empregue na depuração de um circuito que contém um latch.

Exemplo 1:

Analise e descreva a operação do circuito da Fig.13.

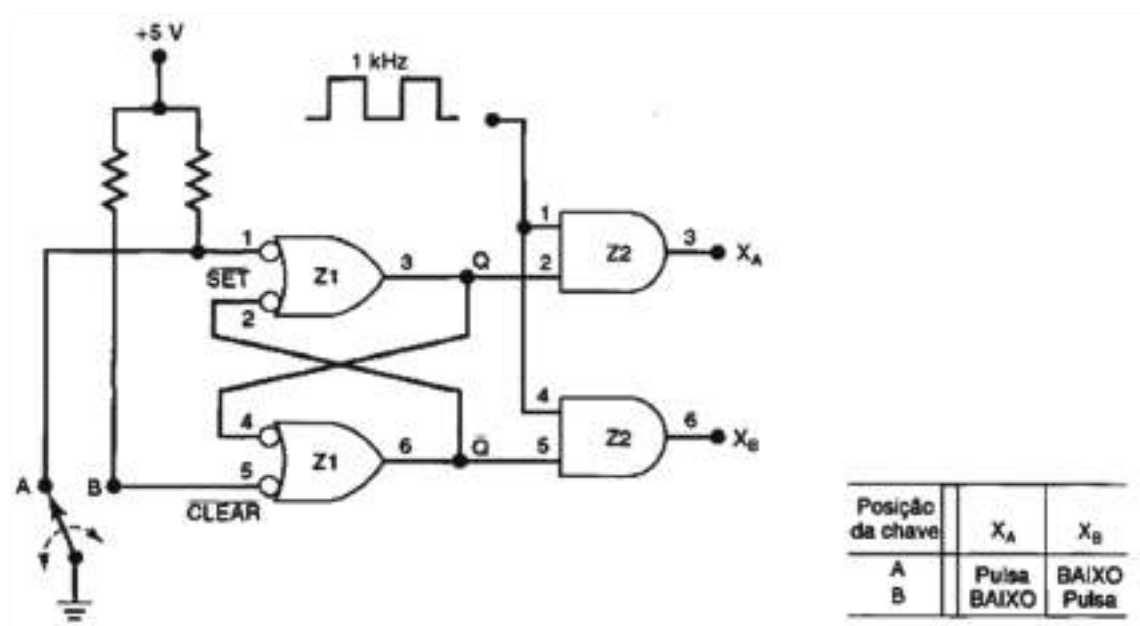


Fig. 13: Exemplo 1 e 2

Exemplo 2:

Um estudante testa o circuito da Fig.13 e anota as suas observações sobre o estado de vários pontos do circuito, conforme é apresentado na Tabela 1. Ele observa que, quando o interruptor está na posição *B*, o circuito funciona corretamente; entretanto, quando está na posição *A*, o latch não vai para o estado onde $Q = 1$. Quais são as possíveis causas deste mau funcionamento?

Posição da chave	SET (Z1-1)	CLEAR (Z1-2)	Q (Z1-3)	\bar{Q} (Z1-6)	X_A (Z2-3)	X_B (Z2-6)
A	BAIXO	ALTO	BAIXO	ALTO	BAIXO	Pulsa
B	ALTO	BAIXO	BAIXO	ALTO	BAIXO	Pulsa



Sinais de CLOCK e Flip-Flops

Os sistemas digitais podem operar em modo assíncrono ou em modo síncrono. Nos sistemas assíncronos, as saídas dos circuitos lógicos podem mudar de estado a qualquer momento em que uma ou mais entradas mudem de estado. Um sistema assíncrono é, geralmente, mais difícil de se projetar e depurar do que um sistema síncrono.

Nos sistemas síncronos, um sinal, comumente chamado CLOCK (relógio), determina os momentos nos quais qualquer uma das saídas pode mudar de estado. Este sinal de CLOCK é geralmente um trem de pulsos retangulares, ou uma onda quadrada, como pode ser visto na Fig.14. O sinal de CLOCK é distribuído para todas as partes do sistema, e a maioria das saídas (senão todas) do sistema pode mudar de estado somente quando o CLOCK faz uma transição. As transições estão indicadas na Fig.14. Quando o CLOCK faz uma transição de 0 para 1, esta é chamada de transição positiva (subida). Quando o CLOCK faz uma transição de 1 para 0, esta é chamada de transição negativa (descida).

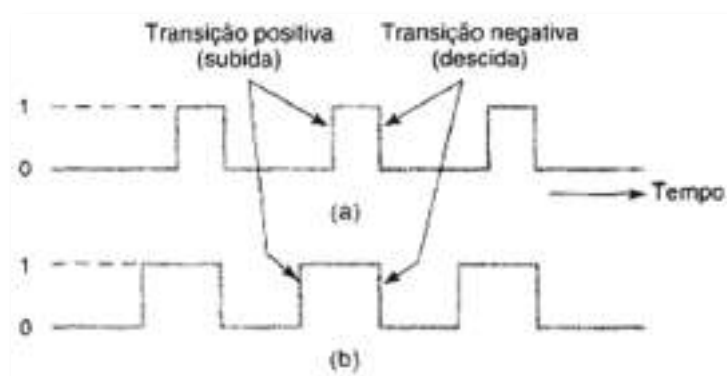


Fig. 14: Sinais de CLOCK

A maioria dos sistemas é síncrona, embora existam sempre algumas partes assíncronas, porque circuitos síncronos são mais fáceis de projetar e depurar. E são mais fáceis de depurar porque as saídas dos circuitos podem mudar de estado apenas em instantes de tempo bem determinados. Por outras palavras, quase tudo está sincronizado com as transições do sinal CLOCK.

A sincronização feita pelos sinais de CLOCK é obtida através do uso de flip-flops com CLOCK que são projetados para mudar de estado numa das transições do CLOCK.



Flip-Flops com CLOCK

Vários tipos de FFs com CLOCK são usados num grande número de aplicações. Antes de iniciarmos nosso estudo dos diferentes tipos de FFs, descreveremos os conceitos fundamentais que são comuns a todos eles.

1. Flip-Flops com CLOCK têm uma entrada de CLOCK que é geralmente chamada de *CLK*, *CK* ou *CP*. Por norma vamos usar o termo *CLK*, como é apresentado na Fig.15. Na maioria dos FFs com CLOCK, a entrada *CLK* é disparada por transição, o que significa que ela é ativada pela transição do sinal presente nesta entrada. Isto é indicado por um pequeno triângulo na entrada *CLK*. Isto diferencia os flip-flops dos latches que são disparados por nível.

Na Fig.15 (a), a entrada *CLK* é ativada apenas quando uma transição positiva ocorre, não sendo ativada em nenhum outro momento. Na Fig.15 (b), a entrada *CLK* é ativa apenas por uma transição negativa, que é simbolizada pela presença de uma pequena bolha.

2. FFs com CLOCK também possuem uma ou mais entradas de controlo que podem ter vários nomes, dependendo do seu funcionamento. As entradas de controlo não têm nenhum efeito sobre *Q* até que ocorra uma transição de disparo na entrada *CLK*. Por outras palavras, o seu efeito sobre *Q* é sincronizado com o sinal aplicado a *CLK*. Por esta razão, são chamadas de entradas de controlo síncronas.

Por exemplo, as entradas de controlo do FF vistas na Fig.15 (a) não afetam *Q* até que uma transição positiva do sinal de CLOCK ocorra. Do mesmo modo, as entradas de controlo da Fig.15 (b) não afetarão *Q* enquanto não ocorrer uma transição negativa do sinal CLOCK.

3. Resumindo, podemos dizer que as entradas de controlo deixam as saídas dos flip-flops prontas para mudar de estado, enquanto a transição ativa na entrada *CLK*, de facto, dispara esta mudança. As entradas de controlo são responsáveis pelo que deve mudar (isto é, para que estado a saída deve ir), enquanto a entrada *CLK* determina quando isto deve acontecer.



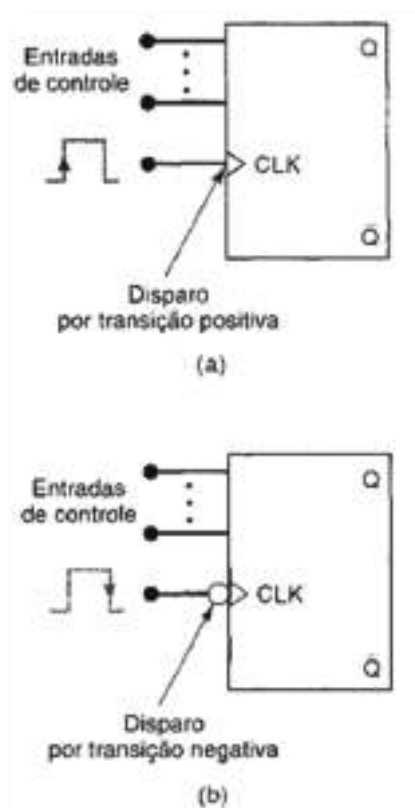


Fig. 15: Flip-flops com CLOCK têm uma entrada de CLOCK que pode ser disparada por (a) uma transição positiva. As entradas de controle determinam o efeito da transição

Tempos de Setup (Preparação) e Hold (Manutenção)

Dois parâmetros de temporização devem ser observados para que o FF responda de modo confiável às suas entradas de controle quando ocorrer uma transição de disparo na entrada *CLK*. Estes parâmetros são ilustrados na Fig.16 para um FF disparado por transição positiva.

O tempo de Setup, t_s , é o intervalo de tempo que precede imediatamente uma transição ativa do sinal de *CLK*, durante o qual cada entrada de controle deve permanecer num nível estável. Os fabricantes de CIs geralmente especificam o tempo mínimo de Setup permitido t_s (min). Se este parâmetro não for respeitado, o FF pode não responder de modo confiável quando houver uma transição do CLOCK.

O tempo de Hold, t_H , é o intervalo de tempo que se segue imediatamente após uma transição de disparo do sinal de *CLK*, durante o qual as entradas de controle síncronas devem ser mantidas num nível estável. Os fabricantes de CIs geralmente especificam um valor mínimo aceitável para o tempo de Hold, t_H (min). Se este parâmetro não for



respeitado, o FF pode não responder de modo confiável quando houver uma transição do CLOCK.

Assim, para garantir que um FF com CLOCK responda de modo correto quando ocorrer uma transição de disparo do CLOCK, as entradas de controle devem estar estáveis, isto é, não devem mudar de estado, pelo menos durante um intervalo de tempo igual a t_s (min) antes da transição do CLOCK, e pelo menos por um intervalo igual a t_H (min) depois da transição do CLOCK.

Os tempos de setup de flip-flops tem valores mínimos permitidos para t_s e t_H na ordem de nanos segundos. Os tempos de Setup são geralmente na ordem de 5 a 50 ns, enquanto os tempos de hold são geralmente na ordem de 0 a 10 ns. Observe que estes intervalos são medidos nos instantes em que as transições estão em 50%.

Esses parâmetros são muito importantes em sistemas síncronos porque, como veremos, existirão muitas situações em que as entradas de controle síncronas vão mudar de estado aproximadamente ao mesmo tempo que a entrada CLK.

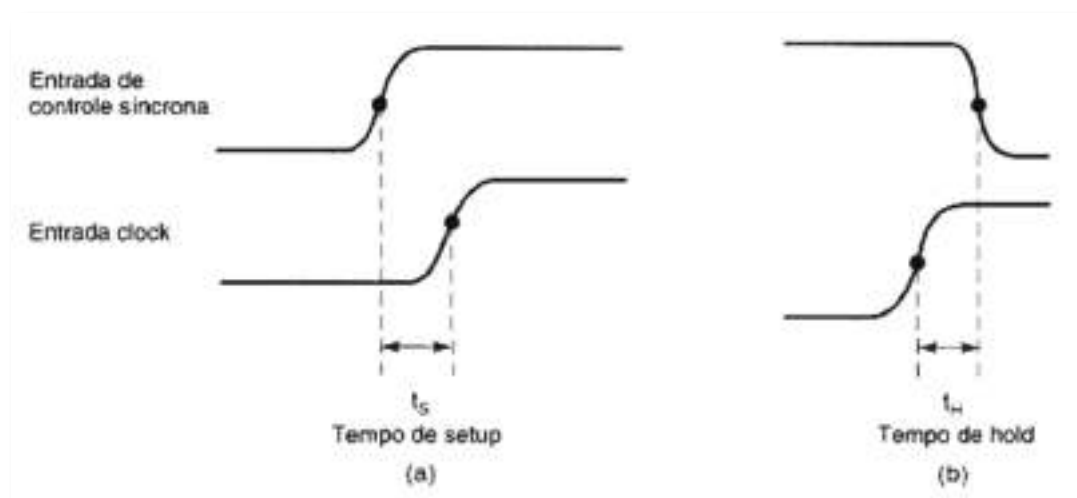


Fig. 16: Entradas de controle devem ser mantidas estáveis por (a) um tempo t_s antes da transição de disparo e por (b) um tempo t_H



Flip-Flop S-C ou S-R com CLOCK

A Fig.17 (a) mostra o símbolo lógico de um flip-flop S-C com CLOCK que é disparado pela transição positiva do sinal de CLOCK. Isto significa que o flip-flop pode mudar de estado somente quando o sinal aplicado na sua entrada CLK faz uma transição de 0 para 1. As entradas S e C controlam o estado do FF, do mesmo modo que foi descrito anteriormente para o caso do latch com portas NOR; entretanto, a saída do flip-flop não responderá a estas entradas até à ocorrência de uma transição positiva do sinal de CLOCK.

A tabela de verdade na Fig.17 (b) mostra como a saída do flip-flop responde a uma transição positiva na entrada CLK para várias combinações possíveis das entradas S e C. Esta tabela de verdade utiliza uma nova nomenclatura. A seta para cima (\uparrow) indica que uma transição positiva é necessária na entrada CLK. Q_0 representa o nível lógico existente antes da transição positiva. Esta nomenclatura é amplamente utilizada pelos fabricantes nos manuais de circuitos integrados.

As formas de onda na Fig.17 (c) mostram a operação de um flip-flop S-C. Se admitirmos que os tempos de Setup e hold são respeitados em todos os casos, podemos analisar as formas de onda como segue:

1. Inicialmente todas as entradas estão em 0 e vamos supor que a saída Q está em 0, ou seja, $Q_0 = 0$.
2. Quando ocorre a primeira transição positiva do sinal de CLOCK (ponto a), as entradas S e C estão ambas em 0, e portanto o estado do flip-flop não é alterado e permanece no estado $Q = 0$ (isto é, $Q = Q_0$).
3. Quando a segunda transição positiva do sinal de CLOCK ocorre (ponto c), a entrada S está agora em alto, enquanto C permanece em baixo, portanto o FF vai para o estado $Q = 1$ na subida do pulso de CLOCK.
4. Quando o terceiro pulso de CLOCK faz a sua transição positiva (ponto e), S está em 0 e C está em 1, o que faz com que o flip-flop vá para o estado 0.
5. No quarto pulso de CLOCK, o flip-flop vai para o estado $Q = 1$ (ponto g) porque $S = 1$ e $C = 0$ quando a transição positiva ocorre.
6. O quinto pulso de CLOCK também encontra $S = 1$ e $C = 0$, quando a transição positiva ocorre. Entretanto, como Q já está em alto, permanecerá neste estado.



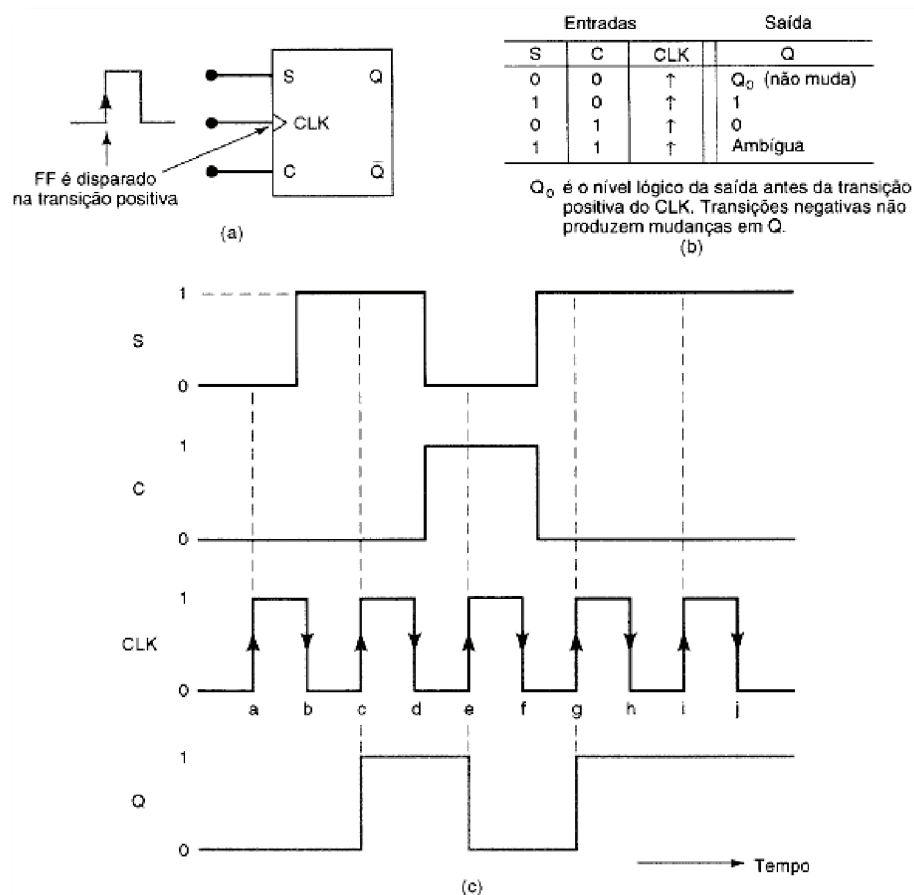
7. A combinação $S = C = 1$ não deve ser usada, pois resulta numa condição duvidosa.

A partir da análise dessas formas de onda, podemos observar que o FF não é afetado pelas transições negativas dos pulsos de CLOCK. Também devemos notar que os níveis lógicos nas entradas S e C não têm efeito sobre o FF, a não ser que ocorra uma transição positiva do sinal de CLOCK. As entradas S e C são entradas de controlo síncronas, pois indicam qual o estado que o FF deve tomar quando ocorrer um pulso de CLOCK. A entrada CLK é a entrada de disparo, isto é, a entrada que faz com que o FF mude de estado de acordo com os níveis das entradas S e C , quando uma transição de disparo no sinal de CLOCK ocorrer.

A Fig.18 mostra o símbolo e a tabela de verdade para um flip-flop S-C com CLOCK que é disparado por uma transição negativa na entrada CLK . A pequena bolha junto com o triângulo na entrada CLK indica que este FF vai ser disparado quando houver uma transição de 1 para 0 na entrada CLK .

O FF opera do mesmo modo que aquele disparado por transição positiva, exceto pelo facto de que a saída muda de estado somente nas transições de descida dos pulsos de CLOCK (pontos b, d, f e j na Fig.17). Tanto os flip-flops disparados por transição negativa quanto aqueles disparados por transição positiva são usados em sistemas digitais.

Fig. 17: Flip-flop S-C com CLOCK que responde somente às transições positivas dos pulsos de CLOCK; (b) tabela de verdade; (c) formas de onda típicas.



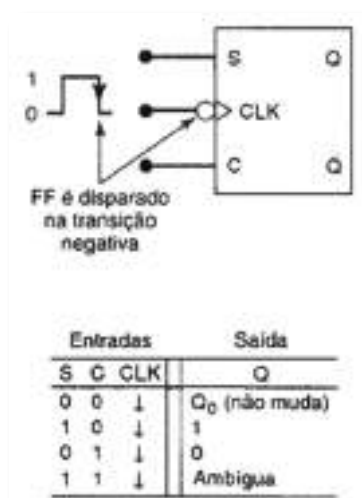


Fig. 18: Flip-flop S-C com CLOCK que é disparado apenas nas transições negativas.

Circuito interno de um Flip-Flop S-C (S-R) disparado por transição

Uma análise detalhada do circuito interno de um FF com CLOCK não é necessária, uma vez que todos os tipos estão disponíveis como circuitos integrados. Apesar do nosso interesse estar no funcionamento externo dos flip-flops, podemos compreendê-lo melhor estudando uma versão simplificada dos circuitos internos dos flip-flops. Esta pode ser vista na Fig. 19.

O circuito pode ser dividido em três partes principais:

1. Um latch NAND formado pelas portas NAND-1 e NAND-2.
2. Um circuito controlador de pulsos formado pelas portas NAND-1 e NAND-2.
3. Um circuito detetor de transição.

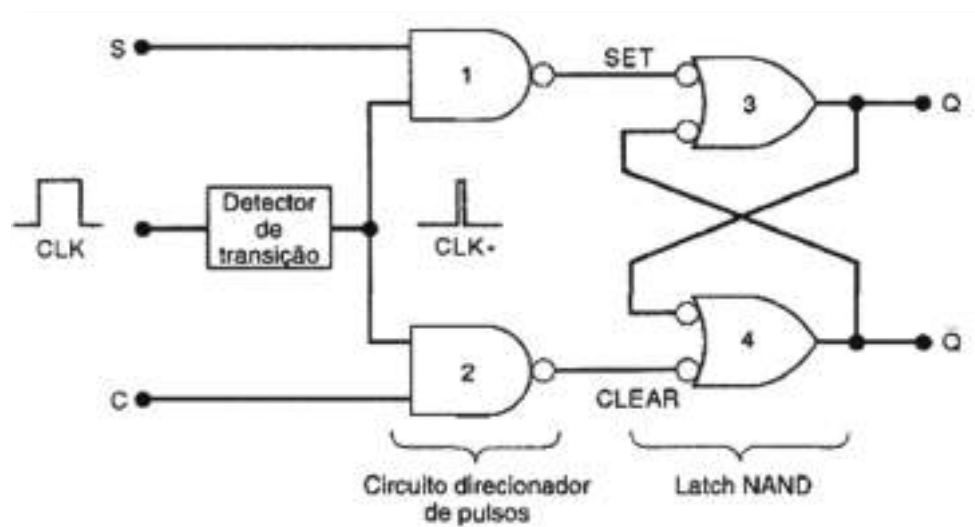


Fig. 19: Versão simplificada do circuito interno de um flip-flop disparado por transição.



Como pode ser visto na Fig. 19, o detetor de transição produz um pulso estreito e positivo (CLK) que coincide com a transição de disparo na entrada CLK . O circuito controlador de pulsos «direciona» este pulso estreito para a entrada SET ou para a entrada CLEAR, de acordo com os níveis presentes em S e C . Por exemplo, para $S = 1$ e $C = 0$, o sinal CLK passa através da porta NAND-1, sendo invertido e produzindo um pulso em nível BAIXO na entrada SET do latch, o que resulta em $Q = 1$. Com $S = 0$ e $C = 1$, o sinal CLK passa através da porta NAND-2, sendo invertido e produzindo um pulso em BAIXO na entrada CLEAR do latch, o que resulta em $Q = 0$.

A Fig.20 (a) mostra como o sinal de CLK é gerado para flip-flops disparados por transição positiva. O INVERSOR produz um atraso de alguns nanos segundos, de modo que as transições de CLK ocorram um pouco depois daquelas de CLK . Uma porta AND produz na saída um pulso em ALTO apenas durante os poucos nanos segundos nos quais CLK e CLK estão ambos em ALTO. Isto resulta num pulso estreito em CLK que ocorre na subida do sinal de CLOCK. A Fig.20 (b) mostra o arranjo necessário para produzir CLK na transição negativa para flip-flops que são disparados na descida.

Uma vez que o sinal CLK está em ALTO apenas por alguns nanos segundos, a saída Q é afetada por S e C apenas por um curto período, após a ocorrência da transição de disparo de CLK . É isto que dá aos flip-flops a característica de disparo por transição.

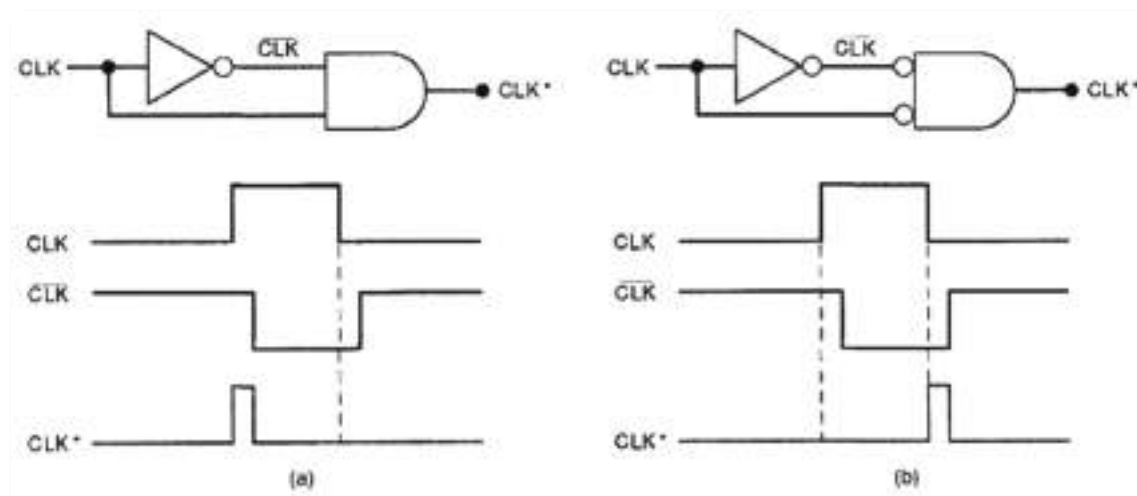


Fig. 20: Implementação de circuitos detentores de transição usados em flip-flops disparados por transição: (a) transição negativa; (b) transição positiva. A duração dos pulsos de CLK é geralmente de 2 a 5 nanos segundos.



Flip-Flop J-K

A Fig.21 (a) mostra um flip-flop J-K com CLOCK que é disparado pela transição positiva do sinal de CLOCK. As entradas J e K controlam o estado do FF do mesmo modo que as entradas S e C o fazem para um flip-flop S-C, mas com uma diferença muito importante: a condição na qual $J = K = 1$ não resulta num estado duvidoso na saída. Para esta condição, o FF vai sempre para o estado oposto quando a transição positiva ocorrer. A isto chama-se operação em modo de comutação (*toggle mode*). Neste modo, se ambas as entradas J e K estão em ALTO, o FF muda de estado (comuta) a cada transição positiva do CLOCK. A tabela de verdade da Fig.21 (a) resume como um flip-flop J-K responde à transição positiva para cada combinação de J e K . Observe que a tabela de verdade é igual à do flip-flop S-C (Fig.17), exceto para a condição $J = K = 1$.

Esta condição resulta em $Q = \bar{Q}$, o que significa que o novo valor de Q será o inverso do que tinha antes da transição positiva. Esta operação é chamada de comutação.

O funcionamento desse flip-flop é ilustrado através das formas de onda da Fig.21 (b).

Mais uma vez, consideramos que os tempos de Setup e hold estão a ser respeitados.

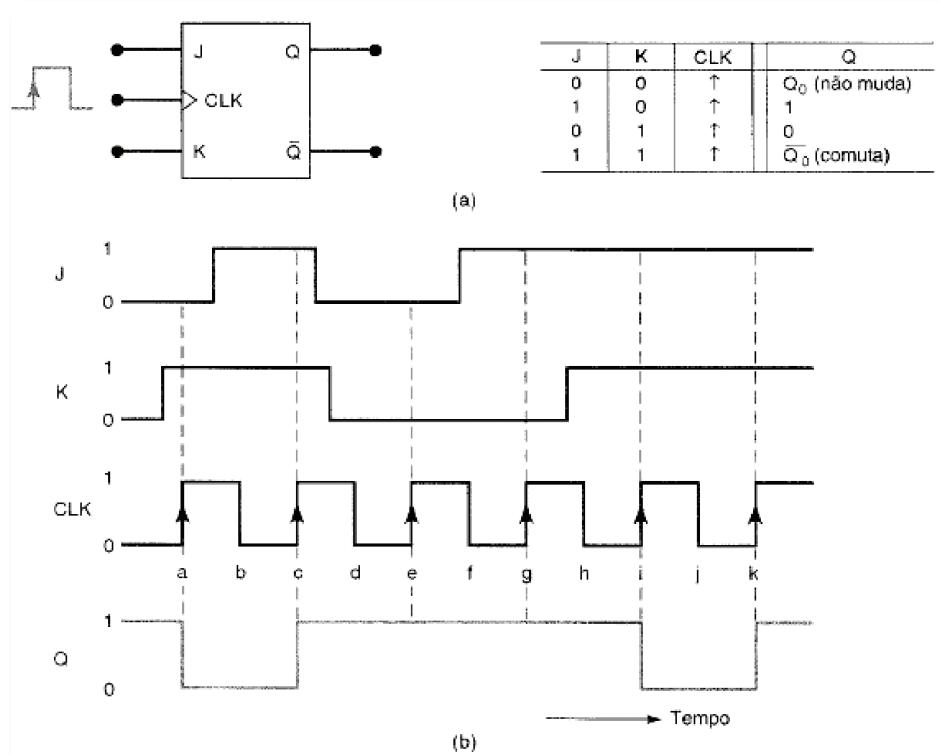


Fig. 21: (a) Flip-flop J-K com CLOCK que responde somente as transições positivas do sinal de CLOCK; (b) formas de onda



1. Inicialmente, todas as entradas estão em 0, e consideramos que a saída Q está em 1; isto é $Q_0 = 1$.
2. Quando a transição positiva do primeiro pulso de CLOCK ocorre (ponto a), temos a seguinte condição de entrada:
 $J = 0$ e $K = 1$ e, portanto, RESET do FF (irá para o estado $Q = 0$).
3. O segundo pulso de CLOCK encontra $J = K = 1$ quando faz a transição positiva (ponto c). Isto faz com que o flip-flop comute para seu estado oposto, $Q = 1$.
4. No ponto (e), tanto J como K são iguais a 0, e portanto o FF não muda de estado durante esta transição.
5. No ponto (g), $J = 1$ e $K = 0$. Esta condição faz com que o FF vá para o estado $Q = 1$. Entretanto, Q já é igual a 1, e portanto o FF permanece neste estado.
6. No ponto (i), $J = K = 1$, e portanto o FF comuta para seu estado oposto. A mesma coisa ocorre no ponto k .

Analizando essas formas de onda, verificamos que o FF não é afetado pelas transições negativas dos pulsos de CLOCK.

Também podemos notar que os níveis presentes nas entradas J e K somente afetam a saída se uma transição positiva do sinal de CLOCK ocorrer. As entradas J e K sozinhas jamais poderão alterar o estado do FF.

A Fig.22 mostra o símbolo para um flip-flop com CLOCK que é disparado na descida do sinal de CLOCK. A pequena bolha na entrada CLK indica que este FF vai ser disparado quando o sinal na entrada CLK fizer uma transição de 1 para 0. Este FF funciona do mesmo modo que o FF disparado por transição positiva mostrado na Fig.21, exceto pelo facto de que a saída vai mudar de estado apenas nas transições negativas do sinal de CLOCK (pontos b , d , f , h e j). Ambos os flip-flops são bastante utilizados.

O flip-flop J-K é muito mais versátil que o flip-flop S-C porque não possui estados duvidosos. A condição de entrada onde $J = K = 1$ produz uma operação de comutação, bastante utilizada em todos os tipos de contadores binários.

Em resumo, o flip-flop J-K pode fazer tudo que o flip-flop S-C pode, além de operar em modo de comutação.



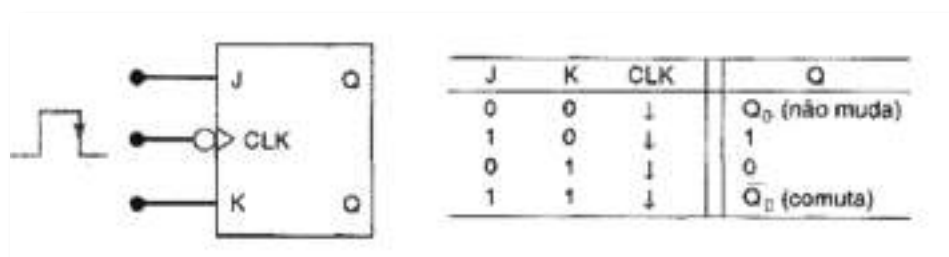


Fig. 22: Flip-flop J-K que dispara apenas nas transições negativas.

Circuito interno de um Flip-Flop J-K disparado por transição

Uma versão simplificada do circuito interno de um flip-flop J-K disparado por transição pode ser vista na Fig.23. Contém as mesmas três partes que possuía o flip-flop S-C disparado por transição (Fig.19). Na verdade, a única diferença entre estes dois circuitos está no facto de que as saídas Q e \bar{Q} são realimentadas para o circuito controlador de pulsos formado pelas portas NAND 1 e 2. Esta realimentação é que fornece ao flip-flop J-K a operação de comutação para a condição onde $J = K = 1$.

Vamos examinar a condição de comutação mais de perto, considerando que $J = K = 1$ e que a saída Q esteja em BAIXO quando o pulso de CLOCK ocorre. Com $Q = 0$ e $\bar{Q} = 1$, a porta NAND 1 vai direccionar CLK (invertido) para a entrada \overline{SET} do latch, fazendo com que a saída Q seja igual a 1. Se supusermos que a saída Q está em ALTO quando ocorre o pulso de CLOCK, a porta NAND 2 vai direccionar CLK (invertido) para a entrada \overline{CLEAR} (\overline{RESET}) do latch para produzir $Q = 0$. Logo, podemos dizer que a saída Q sempre vai para o estado oposto.

A fim de que a operação de comutação funcione como foi descrito anteriormente, o pulso CLK deve ser muito estreito. Deve voltar a 0 antes que as saídas Q e \bar{Q} comutem para novos valores, pois, caso contrario, os novos valores de Q e \bar{Q} vão fazer com que CLK comute as saídas do latch mais uma vez.



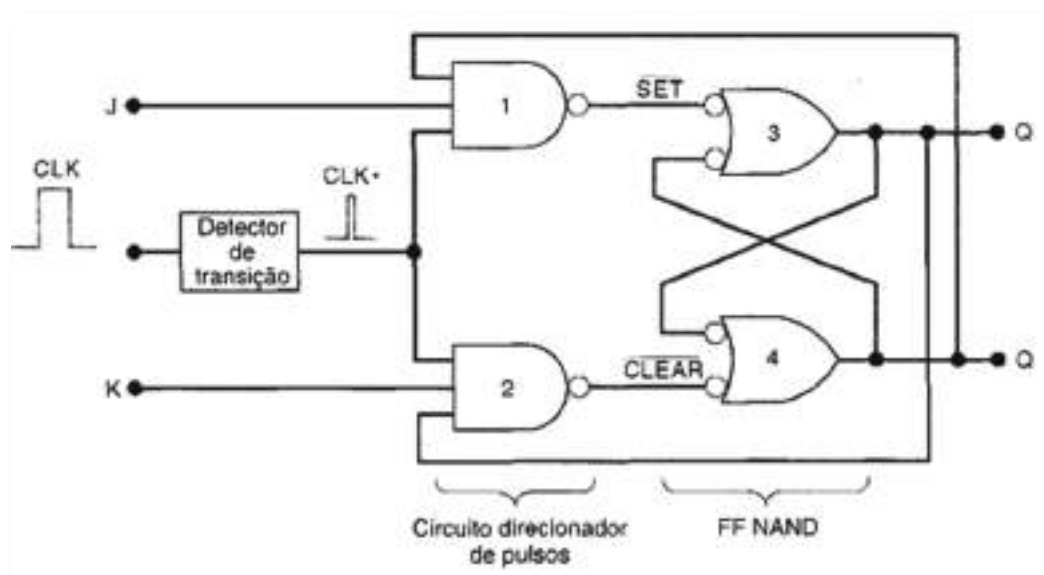


Fig. 23: Circuito interno de um flip-flop J-K comutado por transição.



Flip-Flop tipo D

A Fig.24 (a) mostra o símbolo e a tabela de verdade para um flip-flop D com CLOCK que é disparado na transição positiva. Ao contrário dos flip-flops J-K e S-C, este flip-flop possui apenas uma entrada de controlo síncrona, D , que é a inicial da palavra dados. A operação do flip-flop é muito simples: Q irá para o mesmo estado presente na entrada D quando ocorrer uma transição positiva na entrada CLK . Por outras palavras, o nível presente em D é armazenado no flip-flop no instante em que a transição positiva ocorre. As formas de onda na Fig.24 (b) mostram esta operação.

Suponha que a saída Q esta inicialmente em ALTO. Quando a primeira transição positiva ocorre no ponto (a), a entrada D está em BAIXO, logo, Q irá para BAIXO. Mesmo que o nível na entrada D mude entre os pontos a e b , a saída não é afetada, pois Q está armazenando o nível lógico BAIXO que estava presente na entrada D no ponto (a). Quando a transição positiva ocorre no ponto (b), Q vai para ALTO, uma vez que D está em ALTO neste momento. Q armazena este nível ALTO até que a transição positiva que ocorre no ponto (c) faz com que a saída Q vá para BAIXO, uma vez que D está em BAIXO neste momento. De modo semelhante, a saída Q assume os níveis presentes na entrada D quando ocorre uma transição positiva nos pontos d , e , f e g . Observe que a saída Q permanece em ALTO, no ponto (e), porque D ainda está em ALTO.

Mais uma vez, é importante lembrar que a saída Q pode mudar de estado somente quando ocorrer uma transição positiva. Os valores presentes em D no intervalo entre transições positivas não têm influência na saída.

Um flip-flop D disparado por transição negativa opera do mesmo modo que acabámos de descrever, exceto pelo facto de que a saída Q receberá qualquer valor que estiver em D quando ocorrer uma transição negativa na entrada CLK .

O símbolo para o flip-flop D que dispara numa transição negativa tem uma pequena bolha na entrada CLK .



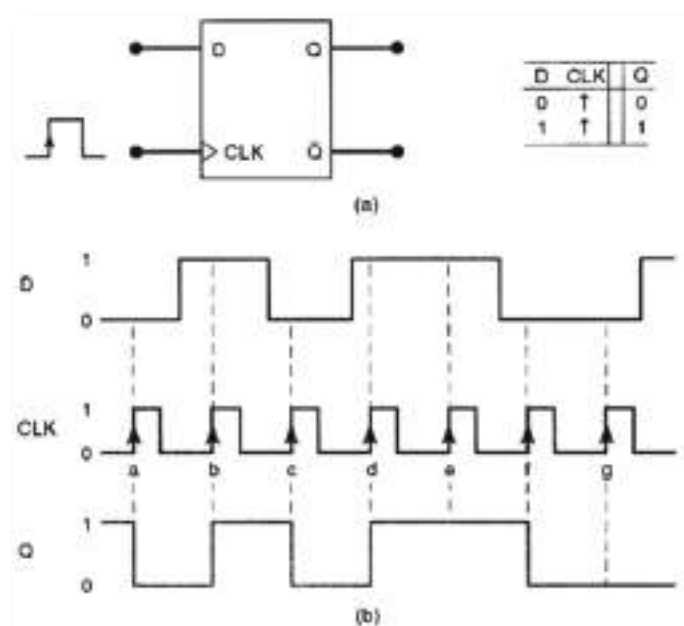


Fig. 24: (a) Flip-flop D que dispara apenas nas transições positivas; (b) formas de onda.

Implementação de um Flip-Flop D

Um flip-flop D disparado por transição pode ser facilmente implementado adicionando-se um INVERSOR ao flip-flop S-C como está apresentado na Fig.25. Se aplicarmos os dois valores possíveis de D a este circuito, vamos verificar que a saída Q assume o nível lógico presente na entrada D quando uma transição positiva ocorre.

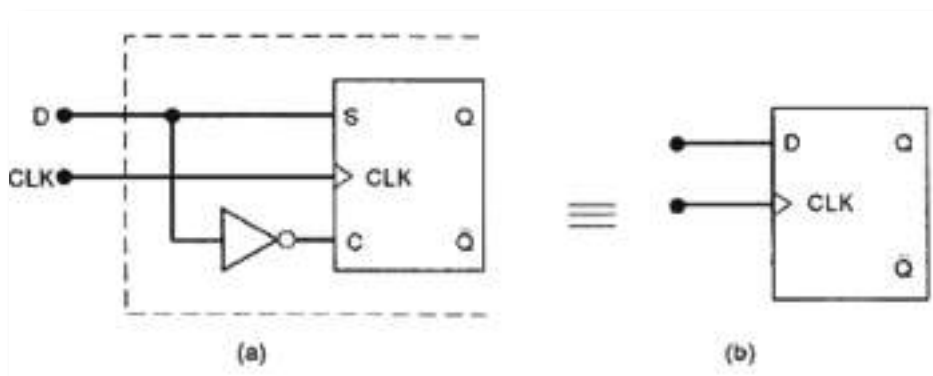


Fig. 25: Implementação de um flip-flop D disparado por transição a partir de um flip-flop S-C.

Exercício 1:

Como um flip-flop J-K pode ser modificado para operar como um flip-flop D?



Transferência de dados em Paralelo

Neste ponto, podemos questionar-nos sobre a utilidade do flip-flop D, uma vez que a saída Q parece ter o mesmo valor do que a entrada D . Esta afirmação não está inteiramente correta, pois a saída Q assume o valor da entrada D em determinados instantes, e portanto não é idêntica a D (como exemplo, veja as formas de onda na Fig.24).

Na maioria das aplicações do flip-flop D, a saída Q deve assumir valores da entrada D em determinados instantes de tempo precisamente definidos. Um exemplo disto pode ser observado na Fig.26. As saídas X , Y , Z de um circuito combinatório devem ser transferidas para os FFs Q_1 , Q_2 e Q_3 para armazenamento. Utilizando-se flip-flops D, os níveis lógicos presentes em X , Y e Z são transferidos, prospectivamente, para Q_1 , Q_2 e Q_3 , mediante a aplicação de um pulso, chamado TRANSFER, nas entradas CLK dos flip-flops. Os FFs podem armazenar estes valores para serem processados posteriormente. Este é um exemplo de transferência paralela de dados binários, onde os bits X , Y e Z são transferidos simultaneamente.

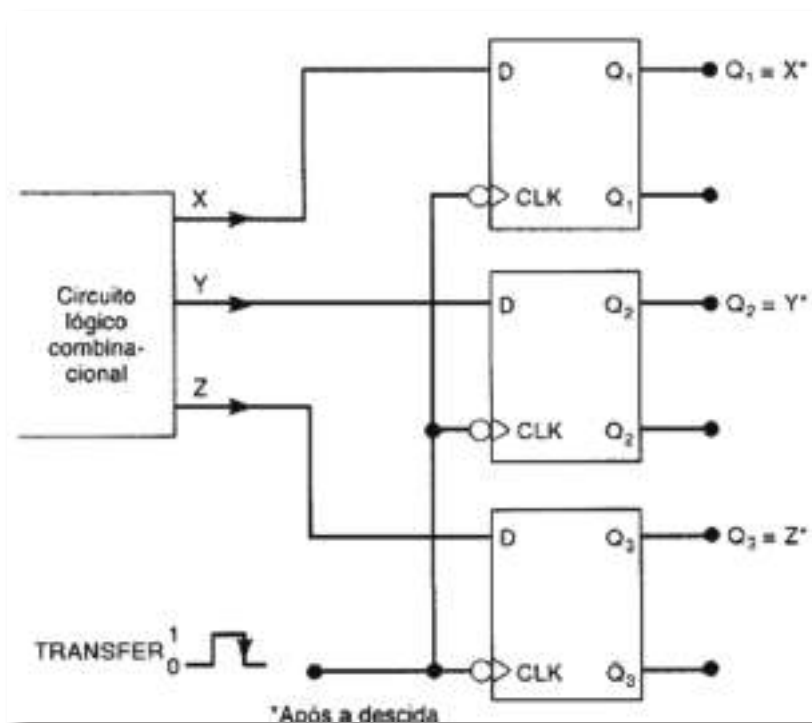


Fig. 26: Transferência de dados binários em paralelo usando flip-flop D



Latch D (Latch transparente)

O flip-flop disparado por transição utiliza um detetor de transição para assegurar que a saída vai responder à entrada apenas quando uma transição de disparo do sinal de CLOCK ocorrer. Se este detetor não for usado, o circuito resultante vai operar de um modo diferente. Este circuito é chamado de latch D e o circuito pode ser visto na Fig.27 (a).

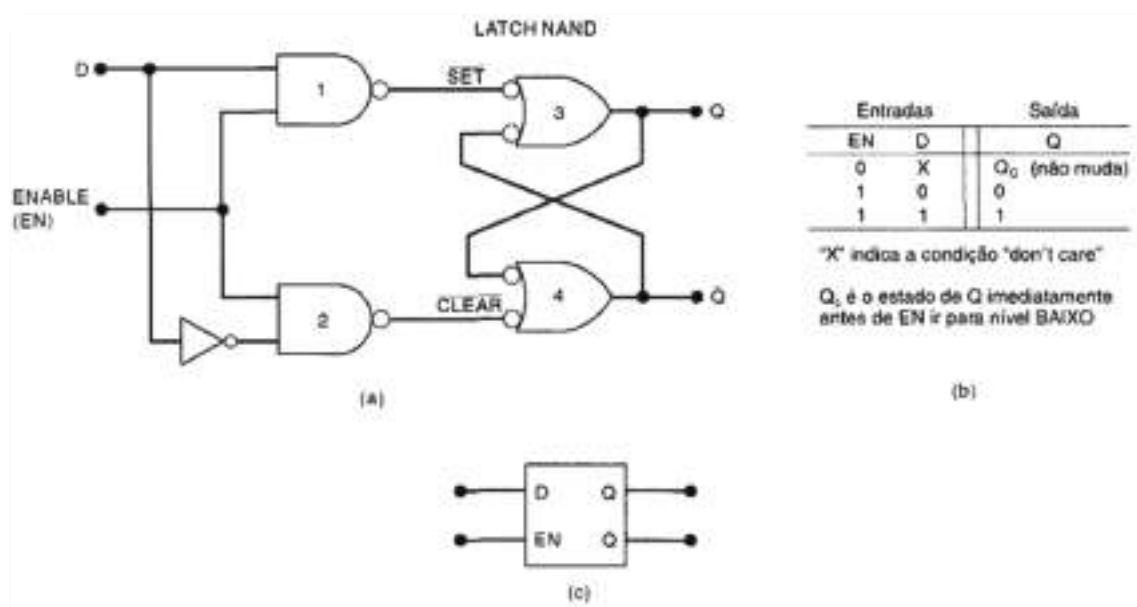


Fig. 27: Latch D : (a) estrutura; (b) tabela de verdade; (c) símbolo lógico.

O circuito contém um latch NAND é um circuito controlador formado pelas portas NAND 1 e 2, mas não possui o detetor de transição. A entrada em comum das portas controladoras é chamada de entrada de habilitação (do inglês *enable*, abreviando-se EN) em vez de ser chamada de entrada de clock, uma vez que o efeito sobre as saídas Q e \overline{Q} não está restrito às transições. A operação do latch D é descrita a seguir:

1. Quando EN está em ALTO, a entrada D vai produzir um nível BAIXO ou na entrada SET ou na entrada CLEAR do latch formados pelas portas NAND 3 e 4. Isto faz com que a saída Q fique no mesmo nível lógico do que a entrada D . Se D mudar de estado enquanto EN estiver ALTO, Q acompanhará estas mudanças. Por outras palavras, enquanto $EN = 1$, a saída Q será igual à entrada D . Deste modo, diz-se que o latch é «transparente».



2. Quando EN vai para o nível BAIXO, a entrada D é impedida de alterar o estado do latch NAND, uma vez que as saídas das portas controladoras estão ambas em ALTO. Logo, as saídas Q e \bar{Q} permanecerão no mesmo nível lógico em que estavam imediatamente antes de EN ir para o nível BAIXO. Por outras palavras, o valor das saídas está «fixo» neste nível, e não pode mudar de valor enquanto EN estiver em BAIXO, mesmo que D mude seu valor.

Esta operação está resumida na tabela de verdade da Fig.28 (b). O símbolo lógico para o latch D é mostrado na Fig.28 (c). Note que, apesar de a entrada EN operar de modo semelhante à entrada CLK em flip-flops disparados por transição, não existe o pequeno triângulo na entrada EN . Isto acontece porque o pequeno triângulo é usado estritamente para indicar entradas que podem causar mudanças na saída apenas quando uma transição ocorre. O latch D não é disparado por transição.

Exercício 2:

Determine a forma de onda para a saída Q do latch D com as formas de onda das entradas EN e D mostradas na Fig.29. Suponha que inicialmente $Q = 0$.

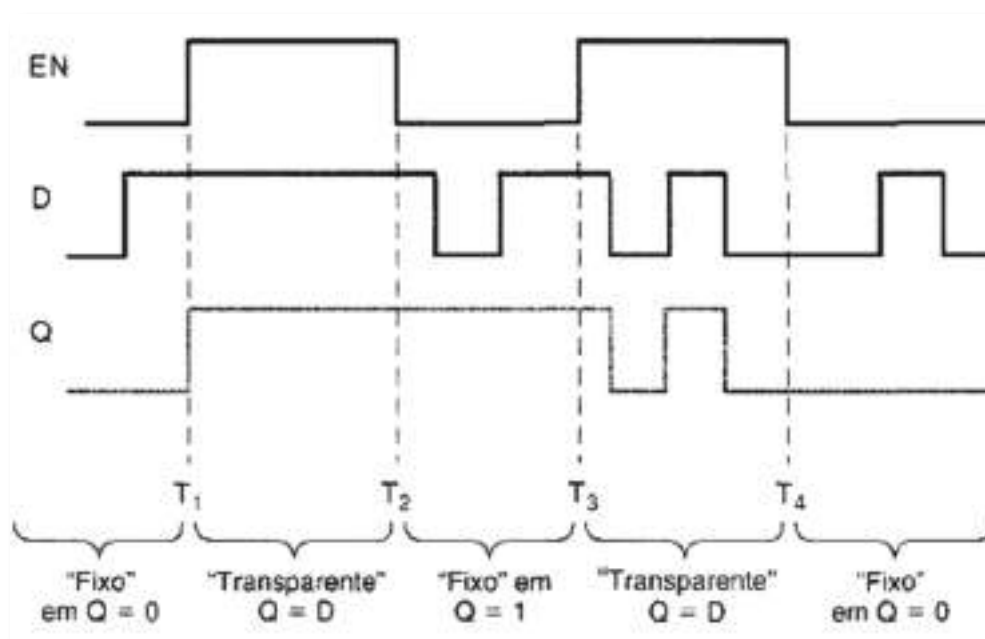


Fig. 28: Formas de onda do Exemplo 5-8 mostrando os dois modos de operação do latch D transparente.



Entradas Assíncronas

Para os flip-flops com CLOCK que estamos estudando, as entradas S , C , J , K e D são chamadas de entradas de controlo.

Também são chamadas de entradas síncronas, porque o efeito sobre a saída é sincronizado com a entrada CLK . Como já vimos, as entradas de controlo síncronas devem ser usadas em conjunto com o sinal de CLOCK para disparar o flip-flop.

A maioria dos flip-flops com CLOCK também possui uma ou mais entradas assíncronas que operam independentemente das entradas síncronas e da entrada de CLOCK. Estas entradas assíncronas podem ser usadas para colocar o flip-flop no estado 0 ou no estado 1, em qualquer instante, independentemente das condições das outras entradas. Por outras palavras, as entradas assíncronas são chamadas de entradas de sobreposição, pois sobrepõem-se a todas as outras entradas para colocar o flip-flop num determinado estado.

A Fig. 29 mostra um flip-flop J-K com duas entradas assíncronas identificadas como PRESET e CLEAR. Estas entradas são ativas em BAIXO, conforme indicam as bolhas presentes no símbolo do flip-flop. A tabela de verdade resume como estas entradas afetam a saída do flip-flop. Vamos examinar as várias possibilidades:

- $\overline{PRESET} = \overline{CLEAR} = 1$. As entradas assíncronas estão inativas e o flip-flop está livre para responder as entradas J , K e CLK , ou seja a operação síncrona pode ser realizada.
- $\overline{PRESET} = 0$; $\overline{CLEAR} = 1$. Como a entrada \overline{PRESET} está ativa, Q é imediatamente colocado em 1, quaisquer que sejam os níveis presentes nas entradas J , K e CLK . A entrada CLK não pode afetar o flip-flop enquanto $\overline{PRESET} = 0$.
- $\overline{PRESET} = 1$; $\overline{CLEAR} = 0$. Como a entrada \overline{CLEAR} está ativa, Q é imediatamente limpo ($Q = 0$), quaisquer que sejam os níveis presentes nas entradas J , K e CLK . A entrada CLK não pode afetar o flip-flop enquanto $\overline{CLEAR} = 0$.
- $\overline{PRESET} = \overline{CLEAR} = 0$. Esta condição não deve ser usada, pois pode resultar numa resposta duvidosa.



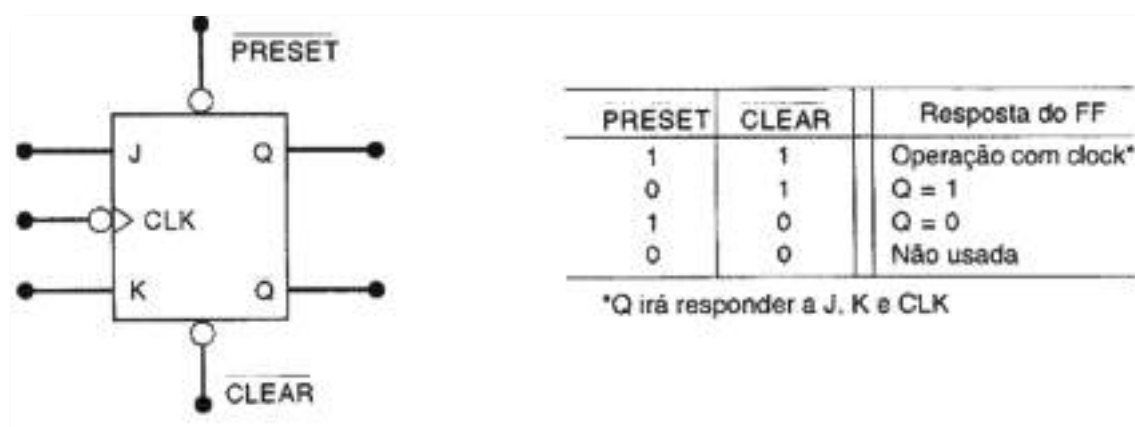


Fig. 29: Flip-flop J-K com CLOCK e entradas assíncronas

É importante perceber que essas entradas assíncronas respondem a níveis de tensão contínua (DC). Isto significa que se um nível 0 for mantido na entrada \overline{PRESET} , a saída Q permanecerá no estado $Q = 1$, independentemente do que estiver ocorrendo com as outras entradas. Do mesmo modo, um nível BAIXO constante em \overline{CLEAR} mantém o flip-flop no estado $Q = 0$. Então, podemos dizer que as entradas assíncronas podem ser utilizadas para manter o flip-flop num determinado estado pelo tempo que desejarmos. Na maioria das vezes, entretanto, as entradas assíncronas são usadas para colocar o flip-flop no estado desejado através da aplicação de um pulso momentâneo.

Muitos dos flip-flops com CLOCK que estão disponíveis em circuitos integrados possuem as duas entradas assíncronas, enquanto outros possuem apenas a entrada \overline{CLEAR} . Alguns flip-flops possuem as entradas assíncronas ativas em ALTO, em vez de ativas em BAIXO. Para estes casos, o símbolo do flip-flop não possui a bolha de inversão nas entradas assíncronas.

Designação para as entradas Assíncronas

Os fabricantes de circuitos integrados ainda não concordaram quanto à nomenclatura a ser utilizada para as entradas assíncronas. As denominações mais comuns são \overline{PRE} (abreviatura de PRESET) e \overline{CLR} (abreviatura de CLEAR). As designações S_D (SET direto) e R_D (RESET direto) também são usadas. De agora em diante, vamos usar as designações \overline{PRE} e \overline{CLR} para indicar as entradas assíncronas, uma vez que estas são as designações mais usadas. Quando estas entradas assíncronas são ativas em BAIXO, como geralmente



são, vamos usar uma barra sobreposta para indicar esta condição, isto é, \overline{PRE} e \overline{CLR} .

Embora a maioria dos CIs de flip-flops possua pelo menos uma ou mais entradas assíncronas, existem algumas aplicações nas quais elas não são usadas. Neste caso, elas devem ser mantidas permanentemente em seu nível inativo.

Muitas vezes durante o uso de flip-flops no restante do texto não mostraremos as entradas assíncronas não-utilizadas, e consideraremos que elas estão permanentemente conectadas ao seu nível lógico inativo.

Exercício 1:

A Fig.30 (a) mostra o símbolo de um flip-flop J-K que é disparado por transições negativas do sinal que está na entrada CLK e que possui entradas assíncronas ativas em BAIXO. Antes de prosseguir com este exemplo, observe o modo pelo qual as entradas são denominadas. Primeiro, note que o sinal de CLOCK aplicado ao flip-flop é denominado de \overline{CLK} (a barra sobreposta indica que o sinal é ativo na posição negativa), enquanto no outro lado da bolha, dentro do bloco, ele é denominado CLK . Do mesmo modo, as entradas assíncronas externas, ativas em BAIXO, são denominadas \overline{PRE} e \overline{CLR} , enquanto dentro do bloco, do outro lado da bolha, são denominadas PRE e CLR . O mais importante a ser lembrado é que a presença da bolha na entrada significa que esta responde a um nível lógico BAIXO.

As entradas J e K estão conectadas em ALTO neste exemplo.

Determine a saída Q em função das formas de onda de entrada mostradas na Fig.30 (a).

Suponha que a saída Q esta inicialmente em ALTO.



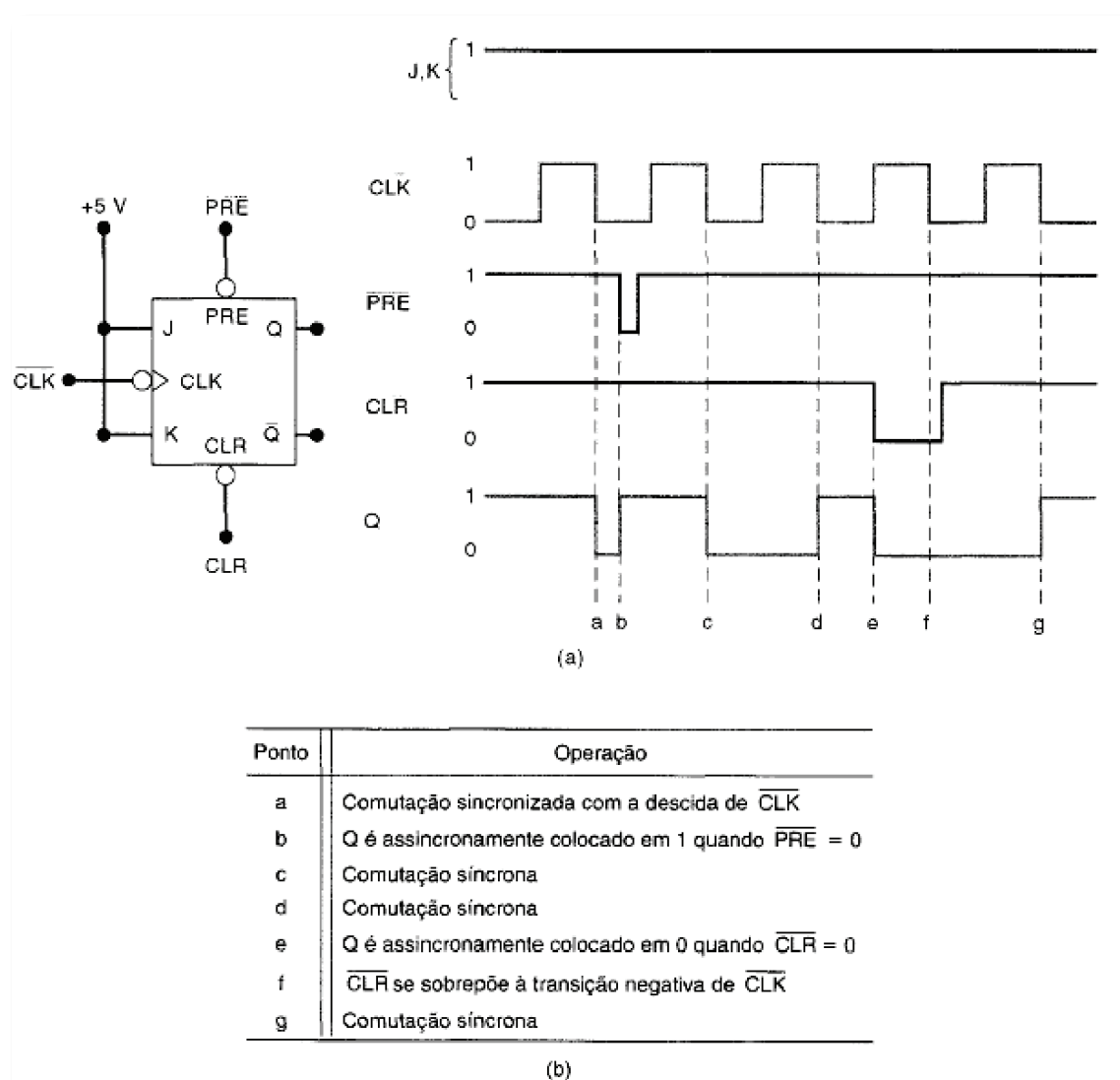


Fig. 30: Formas de onda do exemplo 1 mostrando como um flip-flop J-K com CLOCK responde as entradas assíncronas.



Considerações sobre temporização em Flip-Flops

Os fabricantes de CIs de flip-flops especificam muitos parâmetros importantes de temporização e características que devem ser considerados antes que um FF possa ser usado num circuito. Descreveremos os mais importantes e apresentaremos alguns exemplos reais de CIs de flip-flops das famílias lógicas TTL e CMOS.

Tempos de Setup e Hold

Os tempos de Setup e hold já foram discutidos, e você deve lembrar da Secção anterior que eles representam restrições que devem ser satisfeitas para disparar de maneira confiável num FF.

A folha de características do fabricante do CI sempre especifica os valores mínimos de t_s e t_H .

Atrasos de propagação

Sempre que um sinal causa a mudança de estado da saída de um FF, existe um atraso entre a aplicação do sinal e o momento em que a saída muda. A Fig.31 ilustra os atrasos de propagação que ocorrem em resposta a uma transição positiva na entrada *CLK*. Repare que estes atrasos são medidos entre os pontos de 50% de amplitude das formas de onda de entrada e saída. Os mesmos tipos de atrasos acontecem em resposta a sinais aplicados nas entradas assíncronas (PRESET e CLEAR). As folhas de características (*Datasheets*) dos fabricantes usualmente especificam os valores máximos para t_{PLH} e t_{PHL} . Os modernos CIs de flip-flops possuem atrasos de propagação que variam de uns poucos nanos segundos até por volta dos 100 ns. Os valores de t_{PLH} e t_{PHL} geralmente não são os mesmos, e eles aumentam de modo diretamente proporcional ao número de cargas sendo ligadas pela saída *Q*.

Os atrasos de propagação dos FFs têm um importante papel em determinadas situações que encontraremos mais à frente.



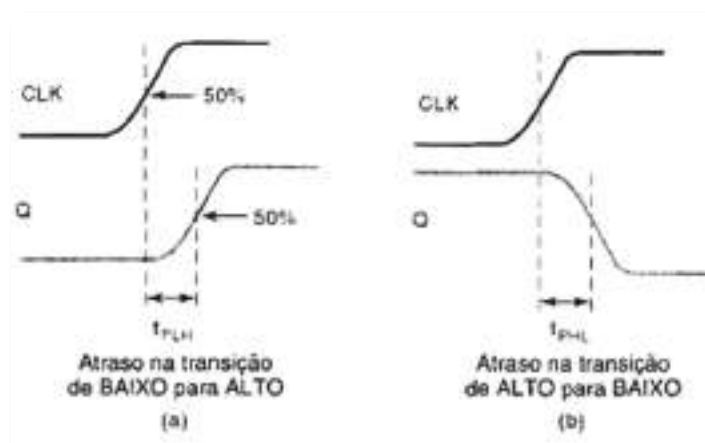


Fig. 31: Atraso de propagação nos FFs

Frequência máxima de CLOCK, f_{MAX}

Esta é a frequência mais alta que pode ser aplicada na entrada *CLK* de um FF e ainda dispará-lo de maneira confiável. O limite f_{MAX} varia de FF para FF, mesmo entre os FFs que têm o mesmo número. Por exemplo, o fabricante do CI 7470 flip-flop J-K testa vários destes FFs e pode constatar que os valores para f_{MAX} ficam na faixa de 20 a 35 MHz. Então especifica a f_{MAX} mínima como 20 MHz. Isto pode parecer confuso, mas um pouco de raciocínio deve tornar claro que o fabricante está a informar que não pode garantir que o FF 7470, que vamos usar nos nossos circuitos, vai operar acima de 20 MHz, a maioria deles funcionará acima disto, mas alguns deles não. Entretanto, se o circuito operar abaixo de 20 MHz, garante que os FFs funcionarão corretamente.

Tempos de duração em ALTO e BAIXO do Sinal de CLOCK

O fabricante também especifica o tempo de duração mínimo que o sinal de *CLK* deve permanecer em BAIXO antes de ir para ALTO, algumas vezes denominado $t_w(L)$, e o tempo mínimo que *CLK* deve ser mantido ALTO antes de voltar para BAIXO, algumas vezes chamado $t_w(H)$. Estes tempos são definidos na Fig.32 (a). O não atendimento a estes requisitos de tempos mínimos pode resultar em disparos não confiáveis. Note que estes valores de tempo são medidos entre os pontos a meio caminho das transições do sinal.



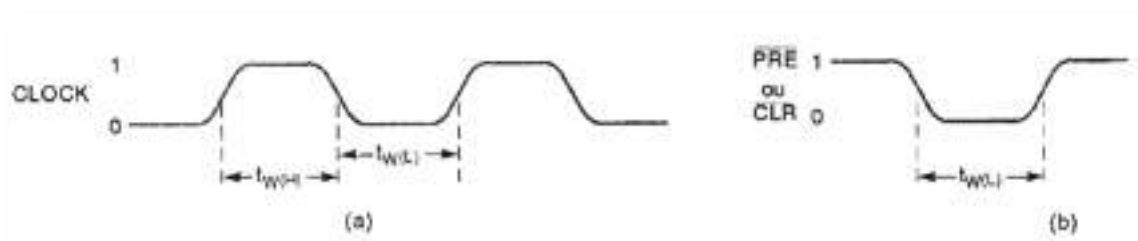


Fig. 32: (a) Tempos de duração do CLOCK em BAIXO e em ALTO; (b) largura de pulso assíncrono.

Largura dos pulsos Assíncronos

O fabricante também especifica o tempo de duração mínimo que as entradas PRESET e CLEAR devem ser mantidas no seu estado ativo, de modo que se faça o SET ou o RESET do flip-flop de maneira confiável. A Fig.32 (b) mostra $t_{W(L)}$ para entradas assíncronas ativas em BAIXO.

Tempos de transição do CLOCK

Para garantir um disparo confiável, os tempos de transição da forma de onda do CLOCK (tempos de subida e descida) devem ser mantidos bem pequenos. Se o sinal de CLOCK demorar muito para fazer a transição de um nível para o outro, o FF pode disparar de modo errático ou simplesmente não disparar. Os fabricantes usualmente não relacionam os requisitos de tempos de transição máximos para cada circuito integrado de FF. Em vez disso, normalmente isto é dado como um requisito geral para todos os CIs de uma determinada família lógica. Por exemplo, os tempos de transição são geralmente ≤ 50 ns para dispositivos TTL e ≤ 200 ns para CMOS. Estes requisitos variam entre os diferentes fabricantes e entre as diversas subfamílias lógicas TTL e CMOS.

CIs Reais

Como exemplos práticos desses parâmetros de temporização, vamos dar uma vista de olhos em vários circuitos integrados reais de FFs. Analisaremos, particularmente, os seguintes CIs:



- 7474 Duplo flip-flop D disparado pela borda (TTL padrão)
- 74LS112 Duplo flip-flop J-K disparado pela borda (TTL Schottky de baixa potencia)
- 74C74 Duplo flip-flop D disparado pela borda (CMOS de porta metálica)
- 74HC112 Duplo flip-flop J-K disparado pela borda (CMOS de alta velocidade)

A Tabela 2 relaciona diversos parâmetros de temporização para cada um destes FFs, conforme apresentados nos manuais dos fabricantes. Todos os valores relacionados são valores mínimos, exceto para os atrasos de propagação, que são valores máximos. Um exame da Tabela 2 revela dois aspectos interessantes.

1. Todos os FFs têm um t_H muito baixo; isto é típico na maioria dos modernos FFs disparados por transição.
2. A série 74HC de dispositivos CMOS têm valores de temporização comparáveis aos dos dispositivos TTL. A série 74C é muito mais lenta do que a série 74HC.

		TTL		CMOS	
		7474	74LS112	74C74	74HC112
t_p		20	20	60	25
t_H		5	0	0	0
t_{su}	de CLK para Q	40	24	200	51
t_{ap}	de CLK para Q	25	16	200	51
t_{su}	de \overline{CLR} para Q	40	24	225	41
t_{su}	de PRE para Q	25	16	225	41
$t_{p}(LO)$	tempo em BAIXO de CLK	37	15	100	25
$t_{p}(HO)$	tempo em ALTO de CLK	30	20	100	25
$t_{p}(LO)$	para PRE ou CLR	30	15	60	25
f_{max}	em MHz	15	30	5	30

Tabela 2: Parâmetros de temporização de flip-flops (em nanos segundos)

Exemplo 1:

Com referencia à Tabela 2, determine o seguinte:

- (a) Considere que a saída $Q = 0$. Quanto tempo demora para Q ir para ALTO quando uma transição positiva ocorre na entrada CLK de um 7474?
- (b) Suponha que a saída $Q = 1$. Quanto tempo demora para Q ir para BAIXO em resposta à entrada CLR de um 74HC112?
- (c) Qual é o pulso mais estreito que pode ser aplicado na entrada \overline{CLR} de um FF 74LS112 para limpar a saída Q de modo confiavel?



- (d) Qual dos FFs na Tabela 2 precisa que as entradas de controlo permanecem estáveis depois da ocorrência da transição ativa do CLOCK?
- (e) Para quais FFs as entradas de controlo devem ser mantidas estáveis, por um certo tempo mínimo, antes da transição ativa do CLOCK?



Divisão de Frequência e Contagem

Observe a Fig. 33 (a). Cada FF tem as suas entradas J e K em nível 1 e, portanto, irá mudar de estado (comutar) sempre que o sinal na sua entrada CLK for de ALTO para BAIXO. Os pulsos de CLOCK são aplicados apenas na entrada CLK do FF Q_0 . A saída de Q_0 está ligada à entrada CLK de Q_1 , e a saída de Q_1 por sua vez, está ligada à entrada CLK de Q_2 . As formas de onda na Fig. 33 (b) mostram como os FFs mudam de estado à medida que os pulsos são aplicados. Vamos destacar alguns pontos importantes:

1. O flip-flop Q_0 comuta na descida de cada pulso de CLOCK. Portanto, a forma de onda da saída Q_0 tem uma frequência que é exatamente igual a metade da frequência do sinal de CLOCK.
2. O flip-flop Q_1 comuta de cada vez que a saída Q_0 vai de ALTO para BAIXO. A forma de onda de Q_1 tem frequência igual a metade da frequência da saída Q_0 e, portanto, um quarto da frequência do sinal de CLOCK.
3. O flip-flop Q_2 comuta cada vez que a saída Q_1 vai de ALTO para BAIXO, logo, a forma de onda de Q_2 tem frequência igual a metade da frequência de Q_1 e, portanto, um oitavo da frequência do sinal de CLOCK.
4. Cada forma de onda é uma onda quadrada (50% da taxa de ciclo).

Conforme foi descrito anteriormente, cada FF divide a frequência de entrada por 2. Portanto, se adicionássemos um quarto FF a esta cadeia, ele teria frequência igual a um dezasseis avos da frequência do sinal de CLOCK, e assim por diante. Usando o número apropriado de FFs, este circuito poderia dividir uma frequência por qualquer potência de 2.

Especificamente, utilizando N flip-flops produziríamos uma frequência de saída no último flip-flop que seria igual a $1/2^N$ da frequência de entrada.

Esta aplicação de flip-flops é chamada de divisão de frequência. Muitas aplicações precisam de divisão de frequência. Por exemplo, um relógio de pulso, que é sem dúvida um relógio «quartz». O termo «relógio quartz» significa que um cristal de quartzo é utilizado num oscilador para gerar uma frequência bastante estável. A frequência natural de ressonância do cristal de quartzo de um relógio é ronda o 1 MHz ou mais.



Para que o indicador dos segundos seja atualizado a cada 1 segundo, a frequência do oscilador é dividida para gerar uma frequência de saída bastante estável e precisa de 1 Hz.

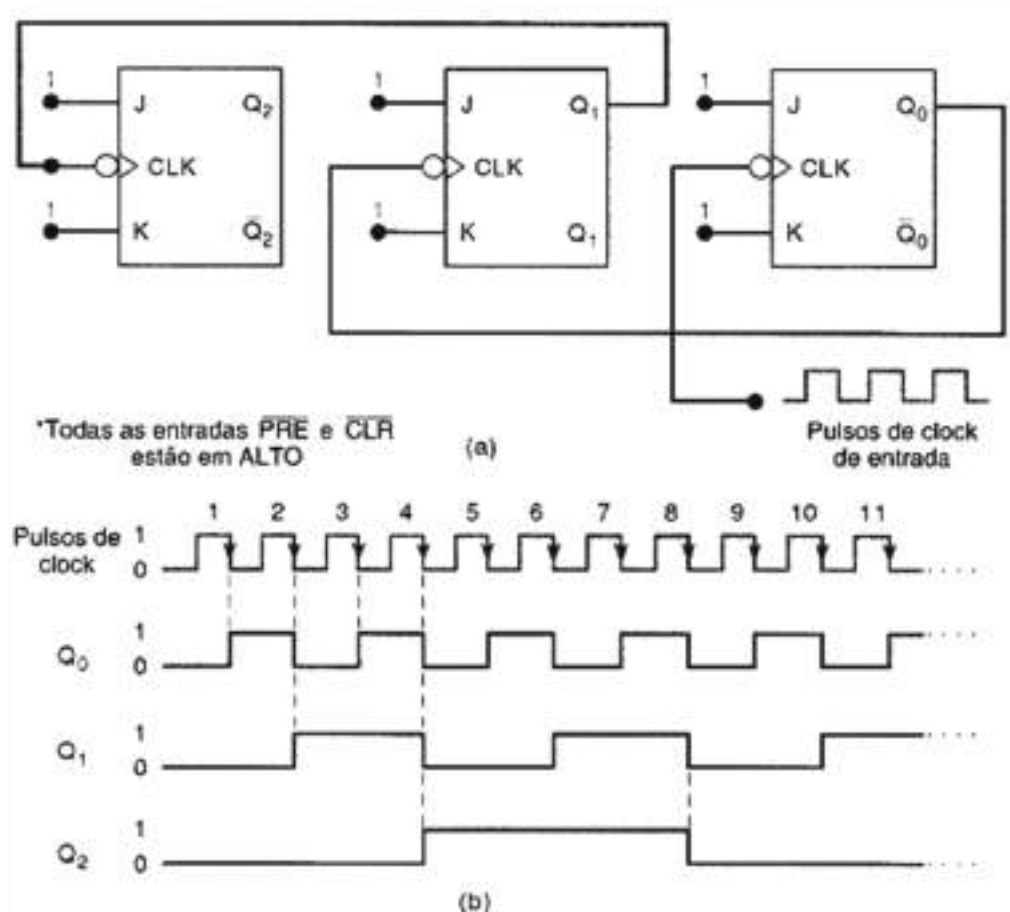


Fig. 33: Flip-Flops J-K ligados para formar um contador de três bits

Operação de contagem

Além de funcionar como um divisor de frequência, o circuito da Fig.33 também funciona como um contador binário. Isto pode ser demonstrado observando-se a sequência de estados dos FFs após a ocorrência de cada pulso de CLOCK. A Fig.34 apresenta os resultados numa tabela de estados. Vamos dizer que $Q_2 Q_1 Q_0$ representam um número binário onde Q_2 está na posição 2^2 , Q_1 está na posição 2^1 e Q_0 está na posição 2^0 . Os primeiros oito estados de $Q_2 Q_1 Q_0$ devem ser reconhecidos como a sequência de contagem binária de 000 a 111. Após a primeira transição negativa, os FFs estão no estado 001 ($Q_2 = 0$; $Q_1 = 0$; $Q_0 = 1$), que representa 001_2 (equivalente ao decimal 1). Após a segunda transição negativa, os FFs estão no estado 010₂, que é equivalente a 2₁₀. Após três pulsos, eles estão em $011_2 = 3_{10}$; após quatro pulsos, eles estão em



$100_2 = 4_{10}$ e assim sucessivamente, até que após 7 pulsos eles estão em $111_2 = 7_{10}$. Na oitava transição negativa, os FFs retornam ao estado 000, e a sequência binária repete-se para os pulsos seguintes.

Então, para os primeiros sete pulsos de entrada, o circuito funciona como um contador binário, no qual o estado dos FFs representa o número binário equivalente ao número de pulsos que já ocorreram. Este contador pode contar até $111_2 = 7_{10}$ antes de voltar a 000.

2^2 Q_2	2^1 Q_1	2^0 Q_0	
0	0	0	Antes dos pulsos de clock serem aplicados
0	0	1	Após pulso #1
0	1	0	Após pulso #2
0	1	1	Após pulso #3
1	0	0	Após pulso #4
1	0	1	Após pulso #5
1	1	0	Após pulso #6
1	1	1	Após pulso #7
0	0	0	Após pulso #8 retorna a 000
0	0	1	Após pulso #9
0	1	0	Após pulso #10
0	1	1	Após pulso #11

Fig. 34: A tabela de estados dos flip-flops mostra a sequência de contagem binária.

Diagrama de transição de estados

Uma outra maneira de mostrar como os estados dos FFs mudam após cada pulso de CLOCK é utilizar o diagrama de transição de estados, como pode ser visto na Fig.35. Cada círculo representa um estado possível, como está indicado pelo número binário que está dentro do círculo. Por exemplo, o círculo que contém o número 100 representa o estado 100 (isto é, $Q_2 = 1$; $Q_1 = Q_0 = 0$).

As setas que ligam um círculo a outro mostram como um estado muda para outro, quando o pulso de CLOCK é aplicado. Basta olhar para um estado em particular e podemos ver o estado que o precede e aquele que o sucede. Por exemplo, olhando para o estado 000, podemos ver que este estado é alcançado sempre que a contagem é 111 e um pulso



de CLOCK é aplicado. De modo semelhante, podemos ver que o estado 000 é sempre seguido pelo estado 001. Vamos usar os diagramas de transição de estados para nos ajudar a descrever, analisar e projetar contadores e outros circuitos sequenciais.

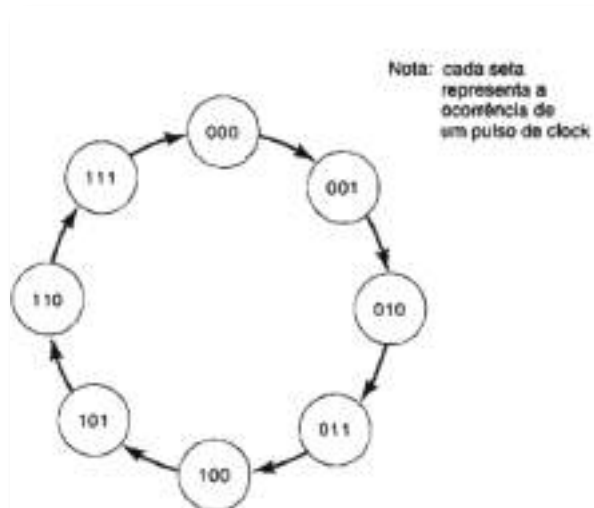


Fig. 35: O diagrama de transição de estados mostra como os estados do contador mudam a cada pulso de CLOCK.

Módulo do Contador

O contador da Fig.34 possui $2^3 = 8$ estados diferentes (000 a 111). Dizemos que este é um contador de *módulo 8*, onde o valor do módulo indica o número de estados da sequência binária. Se um quarto FF fosse adicionado, a sequência de estados contaria, em binário, de 0000 a 1111, num total de 16 estados. Este seria um contador de módulo 16. De um modo geral, se N flip-flops estão ligados na configuração mostrada na Fig.34, o contador resultante terá 2^N estados diferentes, e portanto será um contador de módulo 2^N . Ele será capaz de contar até $2^N - 1$ antes de voltar ao estado 0.

O módulo de um contador também indica a relação entre a frequência de entrada e a frequência de saída no último flip-flop. Por exemplo, um contador de quatro bits possui quatro FFs, onde cada um representa um dígito binário (bit) e é um contador de módulo 16, ele pode contar até 15 ($2^4 - 1$), e também pode ser usado para dividir a frequência de entrada por 16 (o módulo do contador).



Exemplo 1:

Consideremos que o contador de módulo 8 da Fig.34 está no estado 101. Qual será o estado (a contagem) após 13 impulsos terem sido aplicados?

Exemplo 2:

Considere um circuito contador que possui seis FFs ligados segundo o diagrama da Fig.34 ($Q_5 Q_4 Q_3 Q_2 Q_1 Q_0$).

- (a) Determine o módulo do contador.
- (b) Determine a frequência na saída do último FF (Q_5) quando a frequência de entrada é 1 MHz.
- (c) Qual é a faixa de contagem para este contador?
- (d) Suponha que o estado (contagem) inicial é 000000. Qual será o estado deste contador após 129 pulsos?



Dispositivos Schmitt-Trigger

Um circuito Schmitt-trigger não é classificado como um flip-flop, mas possui um certo tipo de características de memória que o torna bastante útil em determinadas situações. Uma destas situações está apresentada na Fig. 36 (a). Neste caso, um INVERSOR comum é acionado por uma entrada lógica que possui tempos de transição relativamente longos. Quando estes tempos de transição ultrapassam o valor máximo permitido (este depende de cada família lógica em particular), podem ocorrer oscilações nas saídas de portas lógicas e de inversores à medida que o sinal de entrada passa pelo intervalo de indeterminação. Estas mesmas condições de entrada podem produzir disparos inesperados de flip-flops.

Um dispositivo que possui uma entrada do tipo Schmitt-trigger é projetado para aceitar sinais cuja transição é lenta e fornecer uma saída livre de oscilações. Esta saída geralmente possui tempos de transição muito rápidos (geralmente 10 ns) e é independente das características do sinal de entrada.

A Fig. 36 (b) mostra um inversor Schmitt-trigger e a resposta a uma entrada que varia lentamente.



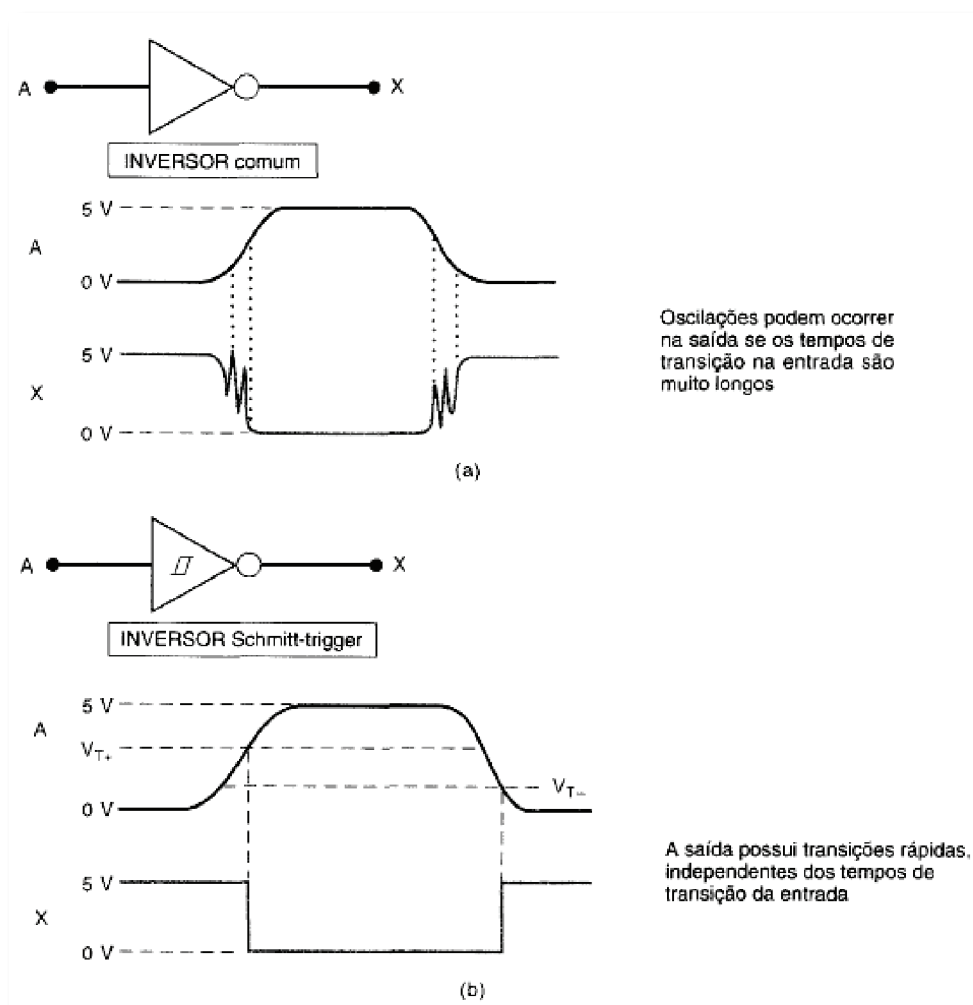


Fig. 36: (a) Se os tempos de transição são muito longos, a saída de um dispositivo lógico pode oscilar ou mudar de estado de modo imprevisível; (b) um dispositivo lógico com entradas do tipo Schmitt-trigger produzirá uma saída com transições rápidas.

Se examinar as formas de onda da Fig.36 (b), deve notar que a saída não muda de ALTO para BAIXO até que a entrada ultrapasse a tensão de limiar superior, V_{T+} . Uma vez que a saída está em BAIXO, ela permanece neste estado, mesmo que a entrada caia abaixo de V_{T+} (esta é a característica de memória) e até que ela caia abaixo da tensão de limiar inferior, V_{T-} . Os valores destes dois limiares variam de uma família lógica para outra, mas V_{T-} será sempre menor do que V_{T+} .

O inversor Schmitt-trigger, e todos os outros dispositivos que possuem entradas deste tipo, utilizam um símbolo especial, mostrado na Fig.36 (b), para indicar que podem responder de modo confiável (há entradas que variam lentamente). Os projetistas de circuitos lógicos usam CIs com entradas Schmitt-trigger para converter sinais que variam



lentamente em sinais com transições rápidas e que podem acionar entradas de CIs comuns.

Vários CIs estão disponíveis com entradas Schmitt-trigger. O 7414, 74LS14 e o 74HC14 são CIs inversores sêxtuplos com entradas Schmitt-trigger. O 74LS13 e o 74HC13 são CIs NANDs duplos de quatro entradas Schmitt-trigger.



Bibliografia

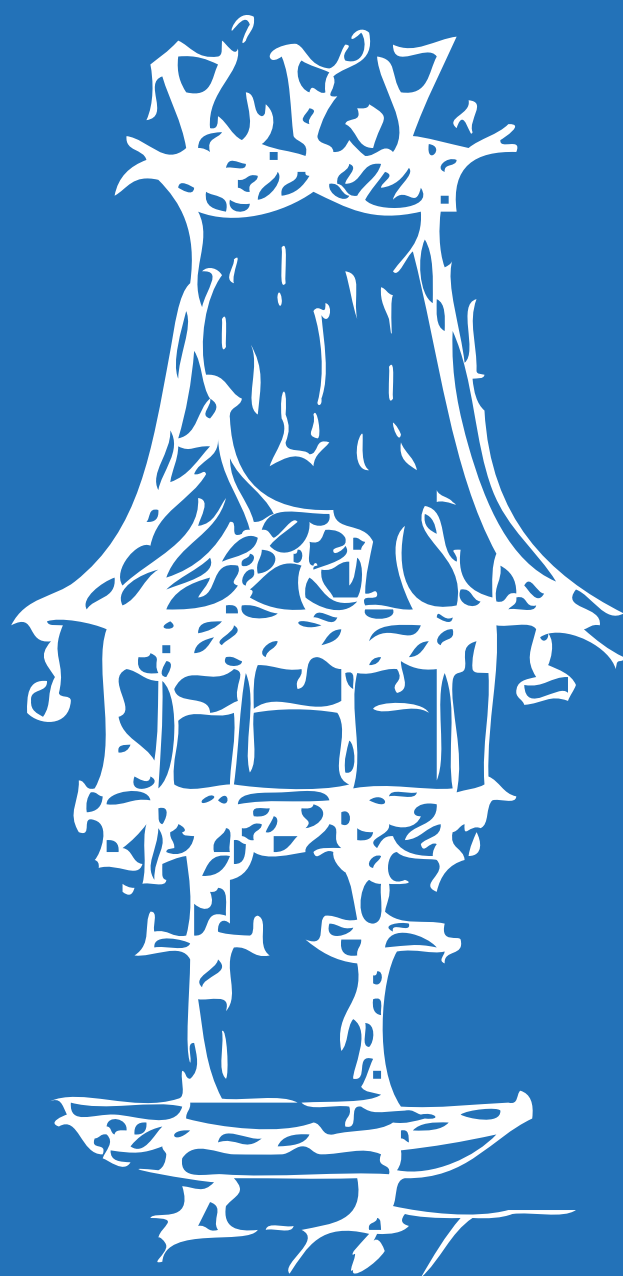
PADILHA, António e outros, *Electrónica Digital*. McGrawHill. (s.d.).

PADILHA, António, *Sistemas Digitais*. McGrawHill. (s.d.).

PEREIRA, A. Silva; ÁGUA, Mário; BALDAIA, Rogério, *Sistemas Analógicos e Digitais, 11.º Ano. Curso Tecnológico de Electrotecnia e Electrónica*. Porto Editora. (s.d.).

PEREIRA, A. Silva; ÁGUA, Mário; BALDAIA, Rogério, *Sistemas Digitais, 11.º Ano. Curso Tecnológico de Electrotecnia e Electrónica*. Porto Editora. (s.d.).







Memórias

Módulo 5

Apresentação

Este módulo tem carácter teórico-prático, devendo por isso decorrer em parte em ambiente laboratorial de modo que o aluno possa verificar e comprovar os conhecimentos teóricos adquiridos sobre os diversos tipos de memórias, suas características e formas de associação.

Introdução

A abordagem deste módulo sobre Memórias leva-nos a uma melhor compreensão do funcionamento de vários tipos de aparelhos, que incorporam circuitos que utilizam estas características, existentes no mercado assim como a melhor escolha deste tipo de equipamentos para que se ajuste às crescentes evoluções disponíveis pelas diversas marcas.

Este módulo requer um conhecimento básico de matemática e análise de circuitos eletrónicos assim como a respetiva compreensão desses circuitos.

Objetivos de aprendizagem

- Conhecer as características mais importantes de uma memória.
- Identificar os vários tipos (classes) de memórias.
- Reconhecer as PLA'S (arranjos lógicos programáveis) nas suas diversas configurações.
- Conhecer a organização interna e configuração externa das memórias.
- Implementar associações de memórias para aumentar a capacidade e/ou a palavra de um sistema.



Âmbito de conteúdos

- Memórias:
 - Características
 - Classes
 - PLA`S (arranjos lógicos programáveis).
 - Configuração externa.
 - Configuração interna.



Introdução

Para muitos de nós, a realização de um projeto de um sistema digital consiste em determinar a função lógica que o sistema que queremos projetar deve apresentar como resposta aos estímulos recebidos, e então construir um circuito lógico complexo que execute essa função a partir de circuitos lógicos simples (portas OR, AND, XOR, Flip-flops, contadores, registradores, etc.).

Mas, devido à complexidade dos sistemas atuais, este tipo de projeto está a tornar-se inviável devido a vários problemas que o projeto de portas lógicas acarreta, tais como:

- Alto número de CI's contendo os circuitos lógicos (portas lógicas) simples necessários;
- Atraso global do sistema alto devido à contribuição individual dos atrasos de cada porta lógica individualmente;
- Alto custo do projeto;
- Necessidade de um grande layout físico para acomodar todos os componentes;
- Alto consumo do sistema;
- Possíveis erros de conexão e/ou mau contatos (confiabilidade do sistema);
- Possíveis indisponibilidades dos circuitos necessários no mercado, acarretando em atraso na finalização dos projetos;
- Necessidade de protótipos para testes, que acarreta em mais gastos.

Além disto, a evolução tecnológica que tem vindo a acontecer nos últimos 35 anos. Com a evolução, ocorreu um aumento da capacidade de processamento dos sistemas e isso acarretou uma maior complexidade dos sistemas a ser projetados (e com isso, um maior número de portas lógicas necessárias ao projeto). E também gerou uma maior escala de integração dos CI's, que é a VLSI (Integração em Altíssima Escala).

Então, com a inviabilidade de se efetuar os projetos digitais da maneira convencional (com portas lógicas), foram implementadas pesquisas a fim de se obter uma forma alternativa mais viável de se efetuar projetos em dispositivos que contivessem milhares de portas lógicas internamente, e que essas portas pudessem ser programadas de acordo com a necessidade do projeto, para implementar a função desejada (em vez de ter de



interligar uma quantidade considerável de CI's contendo portas lógicas, usar-se-ia esse novo dispositivo). Esses estudos geraram a chamada «lógica programável».

Com a lógica programável surgiram vários novos dispositivos passíveis de ser programados pelo projetista (seja enviando o projeto para a produtora dos dispositivos efetuar a programação, ou programando-o o próprio projetista). E, com esses dispositivos, surgiu a necessidade de uma nova forma de se projetar, pois as formas tradicionais de projetos baseados em tabelas da verdade, em *softwares* de projeto lógico a partir de portas lógicas (o Eletronic Workbench é um exemplo), entre outros, já não eram viáveis.

De entre as novas técnicas que surgiram, a que despontou como a mais promissora foi a «descrição de *hardware*». Nesta modalidade de projeto, o projetista, com o auxílio do computador, descreve o *hardware* a ser projetado (o seu sistema digital) utilizando uma HDL (*Hardware* Description Language – Linguagem de Descrição de *Hardware*). Uma HDL é muito parecida com uma linguagem de programação de alto nível, como C ou Pascal. O projeto utilizando HDL torna-se parecido com a programação, uma vez que o projetista inicialmente não se preocupa com a tecnologia que vai ser utilizada na implementação do projeto, e sim com a funcionalidade lógica do projeto.

Após a descrição ser feita, existem várias ferramentas de simulação para testar a funcionalidade do projeto antes da sua implementação. E isto é importante porque reduz drasticamente o tempo de testes, uma vez que não é necessária a construção de protótipos e que, na ocorrência de um erro ou mudança no projeto, é muito simples modificar-se a descrição do sistema em HDL.

No término da etapa de teste, é então escolhido o dispositivo que melhor se adapta ao projeto (número de portas, tempo de resposta, etc.), e então utilizamos um *software* de síntese lógica, disponibilizado pelo fabricante do dispositivo, para convertermos a nossa descrição em HDL para um arquivo que contenha os dados necessários para a programação do dispositivo. E, uma vez realizada a programação, o dispositivo está pronto a ser utilizado.

Deve ressaltar-se que existem dispositivos programáveis que só podem ser programados uma única vez (que são os que o projetista envia a descrição para o fabricante programar), e os que podem ser reprogramados de acordo com a necessidade (que são os programáveis pelo projetista).



Lógica programável / Lógica tradicional

Os componentes da lógica programável são dispositivos que possuem na sua lógica interna centenas (ou milhares) de portas lógicas, flip-flops e registradores, que são interligados internamente. Essas interconexões são os pontos programáveis da lógica. Podemos então programar essas conexões para permanecerem fechadas ou abertas, de acordo com a necessidade do projeto.

Essas interconexões podem ser entendidas como fusíveis, que de acordo com a necessidade do projeto podem ou não ser queimados (desfazendo ou não a conexão entre portas lógicas). Essa «queima» é realizada pelo projetista, utilizando um software de programação do dispositivo.

Existem vários tipos de dispositivos lógicos programável (PLD – Programmable Logic Devices), como os mostrados abaixo:

- PLA
- PAL
- Dispositivos Lógicos Programáveis Complexos (CPLD)
- Arranjo de Portas Programáveis em Campo (FPGA)

Podemos também considerar as memórias PROM como dispositivos de lógica programável se elas forem utilizadas para implementar funções lógicas.

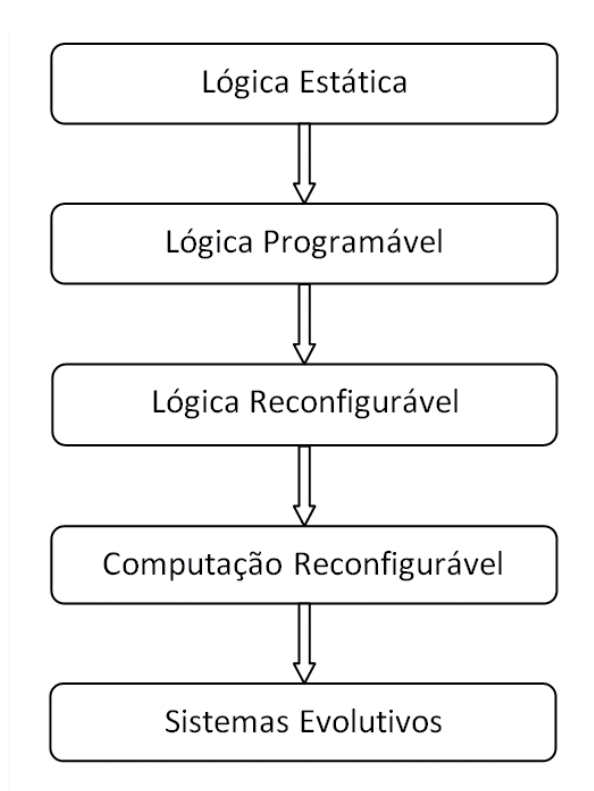
Vantagens

Estes circuitos apresentam as seguintes vantagens em relação à lógica tradicional, que são as seguintes:

- Facilidade para o desenvolvimento de protótipos
- Simulação do projeto intrínseca (no próprio hardware)
- Baixo risco financeiro de desenvolvimento de projetos
- Facilidade para introduzir mudanças no projeto
- Rápida produção



Evolução de Sistemas de Hardware



Memórias PROM

O conceito de programação de *hardware* (sistemas digitais) materializou-se com a necessidade de se construir unidades de memória, cujo conteúdo fixo não era perdido ao desligar o sistema. Esta necessidade foi resolvida com a criação das memórias ROM, que vinham de fábrica com o seu conteúdo já determinado. Com a evolução, surgiram as memórias PROM (ROM programável), cuja programação ocorria pela «queima» dos fusíveis internos (interconexões entre as portas lógicas básicas que compõem a PROM). Temos na figura a seguir o modelo de uma PROM. Internamente, nada mais é do que uma estrutura AND-OR, com a matriz AND fixa e a matriz OR programável. Então, podemos ver a matriz AND da PROM como um decodificador completo de endereços que pode ser programado a partir da matriz OR. Ao ser produzida, a PROM vem com todas as conexões internas. Para programá-la, devemos aplicar níveis de tensão apropriados a fim de manter ou não a conexão de cada entrada de cada porta OR («queimar» ou não os fusíveis internos). Uma vez feita a programação, esta não pode ser desfeita. Com a evolução, surgiram novos tipos de ROMs que solucionaram essa limitação das PROMs, que são as EPROMs apagáveis por radiação ultravioleta, e as PROMs apagáveis eletricamente (EEPROM).

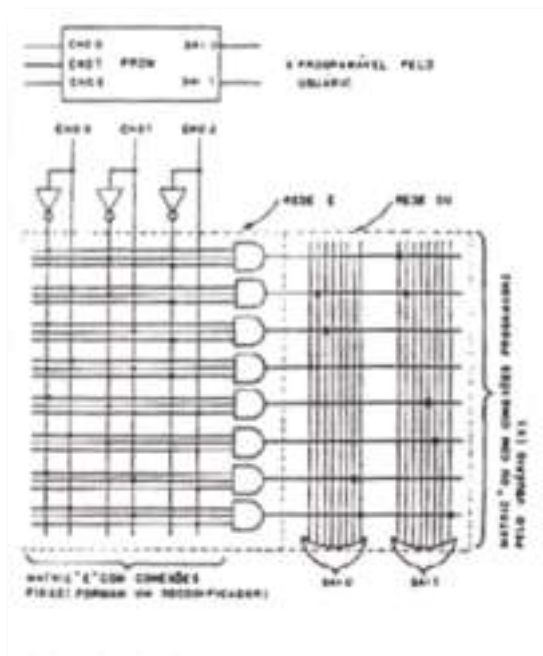


Fig.1 – Memória PROM de 8 x 2 bits



Então, podemos ver a PROM não como uma memória apenas de leitura, mas também como um circuito combinatório genérico de n entradas e m saídas, cuja função lógica executada pode ser facilmente programável, como mostra a figura abaixo.

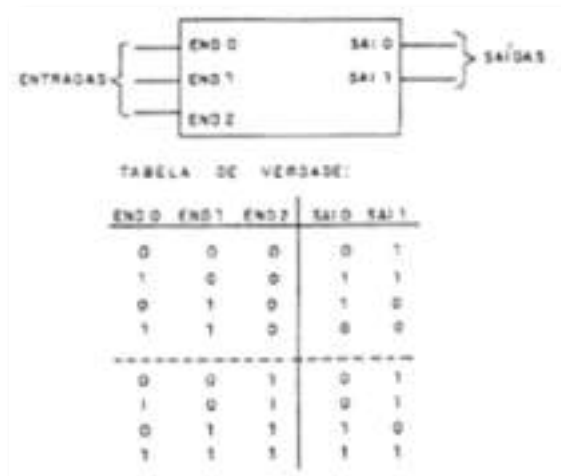


Fig. 2 – PROM vista como um circuito combinatório

Mas o uso de PROMs como dispositivos programáveis apresenta algumas desvantagens, como por exemplo:

- A memória não aproveita as vantagens das técnicas de minimização porque implementam diretamente a tabela da verdade;
- Quando um sistema possui menos saída que o comprimento da palavra da memória, temos porções de memória que não são utilizadas totalmente.

Por isso, os investigadores empenharam-se em produzir dispositivos melhores que as memórias para implementar projetos lógicos, esses componentes são chamados de Dispositivos Lógicos Programáveis (PLD – Programmable Logic Devices).



Os Dispositivos Lógicos Programáveis (PLD)

Dispositivos Lógicos Programáveis são dispositivos (circuitos integrados) configuráveis pelo usuário usados para implementar uma grande variedade de funções lógicas, tanto sequenciais como combinacionais. PLDs podem implementar qualquer expressão booleana ou função construída a partir de estruturas lógicas. Sua programação é efetuada pelo utilizador utilizando uma ferramenta computacional de síntese lógica fornecida pelo fabricante do dispositivo.

Podemos ter PLDs com pontos internos de programação permanentes ou reprogramáveis. Os pontos de programação são os «fusíveis» (conexões) que interconectam os elementos internos do PLD. É através da queima ou não desses «fusíveis» que programamos o dispositivo.

Devido a complexidade da estrutura interna dos PLDs, podemos dividi-los em duas categorias, como mostra a figura abaixo. E cada categoria temos os dispositivos mais representativos.

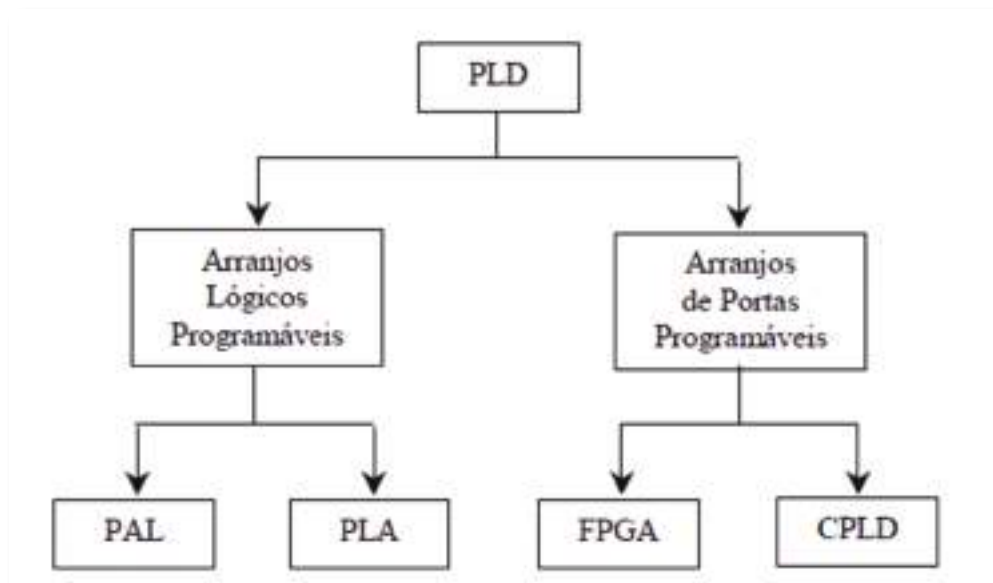


Fig. 3 – Divisão dos PLDs

É importante destacar que estes componentes têm o objetivo principal de oferecer a versatilidade, o baixo custo, a confiabilidade e a velocidade da microeletrônica em



estruturas menos rígidas que as tradicionais. E, para que esses dispositivos sejam utilizáveis de maneira simples, os fabricantes oferecem programas de computador que, a partir de descrições simplificadas do circuito que se deseja projetar (em HDL), consegue-se gerar rapidamente a programação correta do dispositivo. Ressaltando-se que esses *softwares*, além de gerar o conteúdo a ser gravado (programado) no componente, oferecem recursos de simulação (verificação), documenta completamente o projeto e, ainda, gera roteiros de testes.



Arranjos Lógicos Programáveis

Os dispositivos, que são arranjos lógicos programáveis, possuem uma estrutura interna semelhante baseada na estrutura interna AND-OR das PROMs. A estrutura consiste num número de entradas ligadas a um número de portas AND. As saídas das portas AND são conectadas às entradas de um número de portas OR, cujas saídas são as saídas do dispositivo. Nestes dispositivos, podemos ter tanto as duas matrizes de portas programáveis, como apenas a matriz de portas AND programáveis. E isto gerou dois tipos de dispositivos: as PLAs e as PALs. Estes dispositivos, assim como as PROMs, só podem ser programados uma única vez.

PLA

A estrutura de uma PLA é muito semelhante à de uma PROM, mas possui uma menor quantidade de portas AND (não formando um decodificador completo), e possuindo tanto a matriz de portas AND, como a matriz de portas OR programáveis. Temos a seguir a estrutura típica de uma PLA.

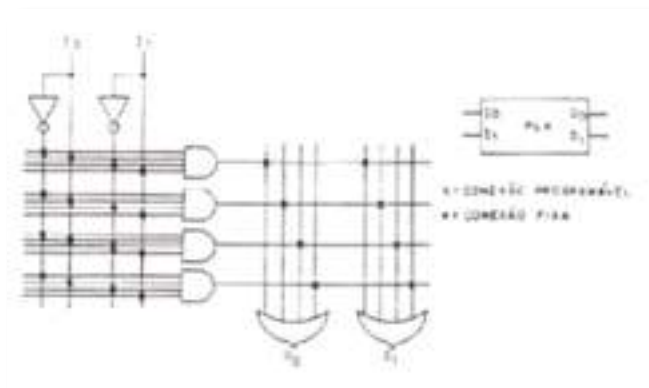


Fig. 4 – Estrutura típica de uma PLA

Possuindo as duas matrizes de portas programáveis, as PLAs possibilitam que o projetista implemente termos minimizados (que não utilizam todas as entradas), como mostra o diagrama simplificado abaixo (note que cada linha de entrada das portas AND representam todas as entradas disponíveis na portas, estando apenas em notação simplificada).



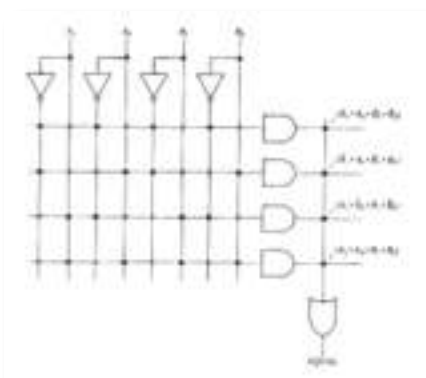


Fig. 5 – Implementação de uma função lógica usando PLA



Dispositivo PAL

Com o uso da PLA's verificou-se que existiam situações em que a flexibilidade da matriz OR programável era desnecessária, o que não justificava o uso da PLAs. Além disso, a existência das duas matrizes (AND e OR) programáveis acarretava uma grande propagação de atraso entre a entrada e a saída do dispositivo (além de um maior custo do dispositivo).

Por causa disso, foi criado um dispositivo baseado no PLA só que mais simples, que é o PAL. O PAL assemelha-se ao PLA, tendo apenas a matriz de portas AND programável (a de portas OR é fixa).

A PAL[®] é um dispositivo programável cuja marca foi registrada pela American Micro Devices (AMD). Os primeiros dispositivos programáveis PAL[®] surgiram no final da década de 70 e o seu sucesso crescente levou a PAL[®] a ser atualmente um dos dispositivos lógicos programáveis mais utilizado.

As PAL[®] baseiam-se no mesmo princípio de implementação da forma AND-OR. No entanto, consideram que a flexibilidade de programação associada à matriz de saída não traz grandes benefícios à capacidade de produção de funções lógicas. Consequentemente, por enquanto na PLA as matrizes de entrada e de saída são ambas programáveis, na PAL[®] apenas a matriz de entrada é programável. A matriz de saída tem uma estrutura fixa não programável. Por este motivo, a PAL[®] é mais fácil de programar e tem um custo mais baixo quando comparada com a PLA. No entanto, não é tão flexível relativamente à programação, pois a matriz de saída é fixa.

Estrutura interna

Para ilustrar a arquitetura típica de uma PAL[®], consideremos uma configuração exemplo com apenas quatro entradas e quatro saídas como demonstra a figura seguinte.



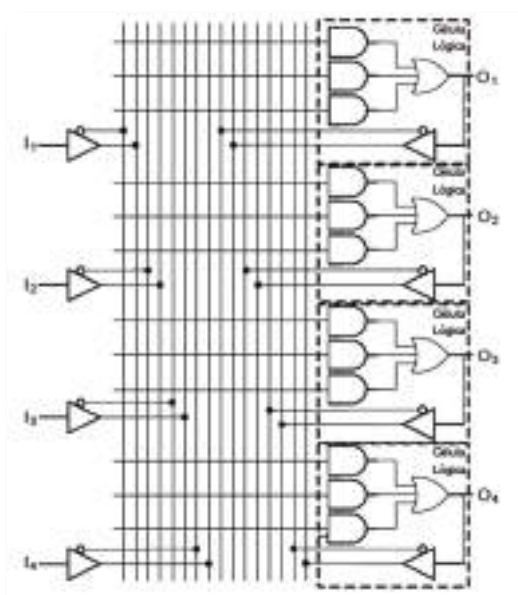


Fig. 6 - Exemplo de arquitetura de uma pequena PAL®

O dispositivo representado na figura anterior apresenta uma estrutura regular formada pela matriz de entrada programável, por quatro entradas (I_n) complementadas e não complementadas e por quatro saídas. Cada uma das saídas é gerada por uma estrutura AND-OR que, neste caso particular, é idêntica para todas as saídas e formada por três portas AND com entradas programáveis. Cada uma das portas AND tem dezasseis ligações de entrada programáveis cujos sinais provêm das quatro entradas, complementadas e não complementadas, e das quatro saídas, complementadas e não complementadas, que são reintroduzidas na matriz de entrada através de um buffer. Ao conjunto de lógica associada a cada uma das saídas (no exemplo, a estrutura AND-OR com três portas AND e o buffer de realimentação) designa-se célula ou macrocélula. O número de entradas e de saídas, bem como a estrutura da macrocélula, são os principais parâmetros lógicos diferenciadores do tipo de PAL®.

Implementação de funções lógicas com PAL

A implementação de um conjunto de funções com uma determinada PAL® está dependente do número de entradas e de saídas e do número de termos de produto. Devido à limitação do número de termos de produto por célula, as funções devem ser previamente simplificadas antes de serem implementadas na PAL®. Se, mesmo



após simplificação, o número de produtos for superior ao existente numa única célula, então é necessário usar duas ou mais células para implementar a função recorrendo à realimentação das saídas das células através dos *buffers*. Ao contrário da PLA, em que um termo de produto podia ser diretamente partilhado por duas ou mais portas OR, na PAL® isso não é possível. Como tal, as funções podem ser simplificadas individualmente sem ter em conta a partilha de termos de produto (simplificação multifunção). Por outro lado, uma vez que é possível realimentar o valor da função de saída, pode ser útil identificar termos AND-OR comuns a duas ou mais funções.

Consideremos, como exemplo, a utilização da PAL® da figura 14 para implementação de quatro funções lógicas, F0, F1, F2, F3:

$$F_3 = \overline{A}BC + A\overline{B}CD;$$

$$F_2 = AB + A\overline{C}\overline{D} + \overline{A}BCD;$$

$$F_1 = A\overline{D} + \overline{B}C;$$

$$F_0 = A\overline{D} + \overline{B}C + \overline{A}BD + \overline{B}\overline{D};$$

A PAL® tem entradas suficientes para implementar as funções de quatro variáveis cada. Além disso, numa primeira aproximação, o número de saídas também é suficiente para implementar as quatro funções. No entanto, o número de termos de produto da função F0 é superior ao de uma única célula, que tem apenas três. Neste caso, é necessário usar mais do que uma célula para implementar a função F0. Consequentemente, à primeira vista, a PAL® proposta não seria solução, pois apenas suportaria a implementação de apenas três das quatro funções. Contudo, ao analisarmos as funções, verificamos que a expressão lógica da F1 ($A\overline{D} + \overline{B}C$) faz parte da expressão lógica de F0 ($A\overline{D} + \overline{B}C + \overline{A}BD + \overline{B}\overline{D}$).

Assim, a função F0 pode ser escrita como:

$$F_0 = F_1 + \overline{A}BD + \overline{B}\overline{D}$$

e desta vez já só tem três termos de produto, suportável por uma única célula da PAL®. Caso o conjunto de funções fosse formado apenas por F3, F2 e F0, apenas teríamos de criar uma nova função que realizaria parte de F0.

Por exemplo:

$$F_1 = \overline{B}C + \overline{A}BD + \overline{B}\overline{D}$$

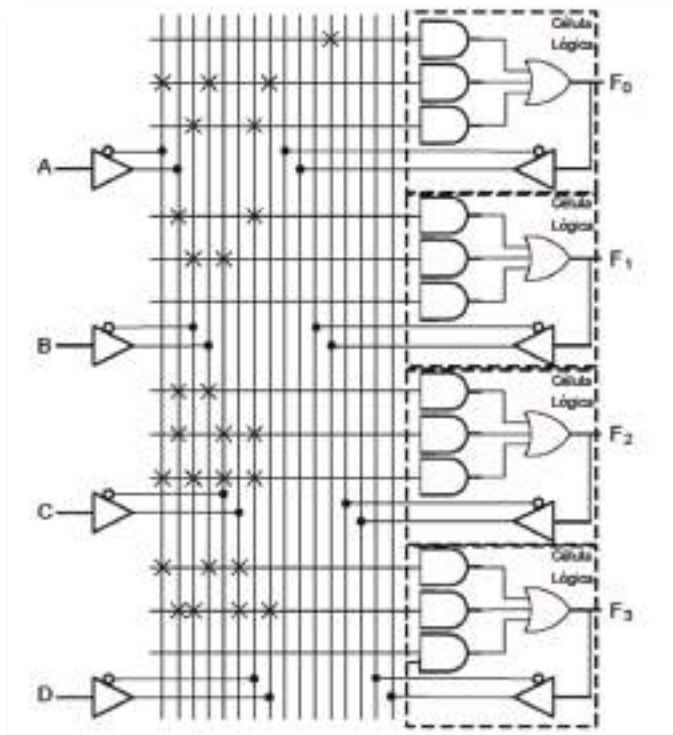
Ou



$$F_3 = \overline{A}D + \overline{B}C$$

Depois de simplificar as funções, basta programar a matriz da PAL®, ver a figura seguinte.

Fig. 7 - Implementação das quatro funções na PAL®



Na implementação ilustrada na figura, a representação das ligações é idêntica à usada nos dispositivos anteriores, ou seja, uma cruz na intersecção indica que o sinal dessa coluna entra na porta AND da linha respetiva.

Felizmente, como acontece com todos os outros dispositivos programáveis, o fabricante disponibiliza ferramentas de CAD para geração do ficheiro de programação a partir da descrição das funções lógicas e para programação da PAL®, tendo como entrada o ficheiro de programação.

PAL Comerciais

As PAL® comerciais são consideravelmente maiores que a utilizada no exemplo anterior. De entre as PAL® comerciais, existem as que apenas implementam lógica combinatória e as que também implementam lógica sequencial, pois incluem elementos de memória nas macrocélulas. Nesta secção, vamos descrever a estrutura de uma PAL® combinatória, PAL16L8, e duas PAL® sequenciais bastante utilizadas, PAL22V10 e ATF750C.



Arquitetura da PAL[®] combinatória 16L8

A PAL16L8 suporta um máximo de 16 entradas e 8 saídas (daqui advêm os números usados na designação 16L8) e tem uma matriz programável com 64 linhas e 32 colunas, perfazendo um total de 2048 ligações programáveis (ver figura 8). As células de saída são formadas por:

- Uma estrutura AND-OR com 7 portas AND de 32 entradas cada, correspondentes ao conjunto das 16 entradas complementadas e não complementadas;
- Uma porta three-state inversora à saída da porta OR cuja entrada de enable é controlada por uma oitava porta AND também com 32 entradas. As entradas não conectadas são interpretadas pela porta AND como sendo o valor lógico 1. Assim, por exemplo, caso se pretenda que a porta three-state esteja sempre ativa, basta não ligar nenhum sinal à porta AND de controlo;
- Um buffer de realimentação da saída respetiva para a matriz de entrada.

Embora a PAL[®] tenha 16 pinos de entrada e 8 pinos de saída, o encapsulamento do integrado apenas tem 20 pinos, incluindo os dois pinos de alimentação. Para que tal seja possível, 6 dos 20 pinos são bidirecionais, podendo ser usados como entrada, como saída ou como entrada/saída.



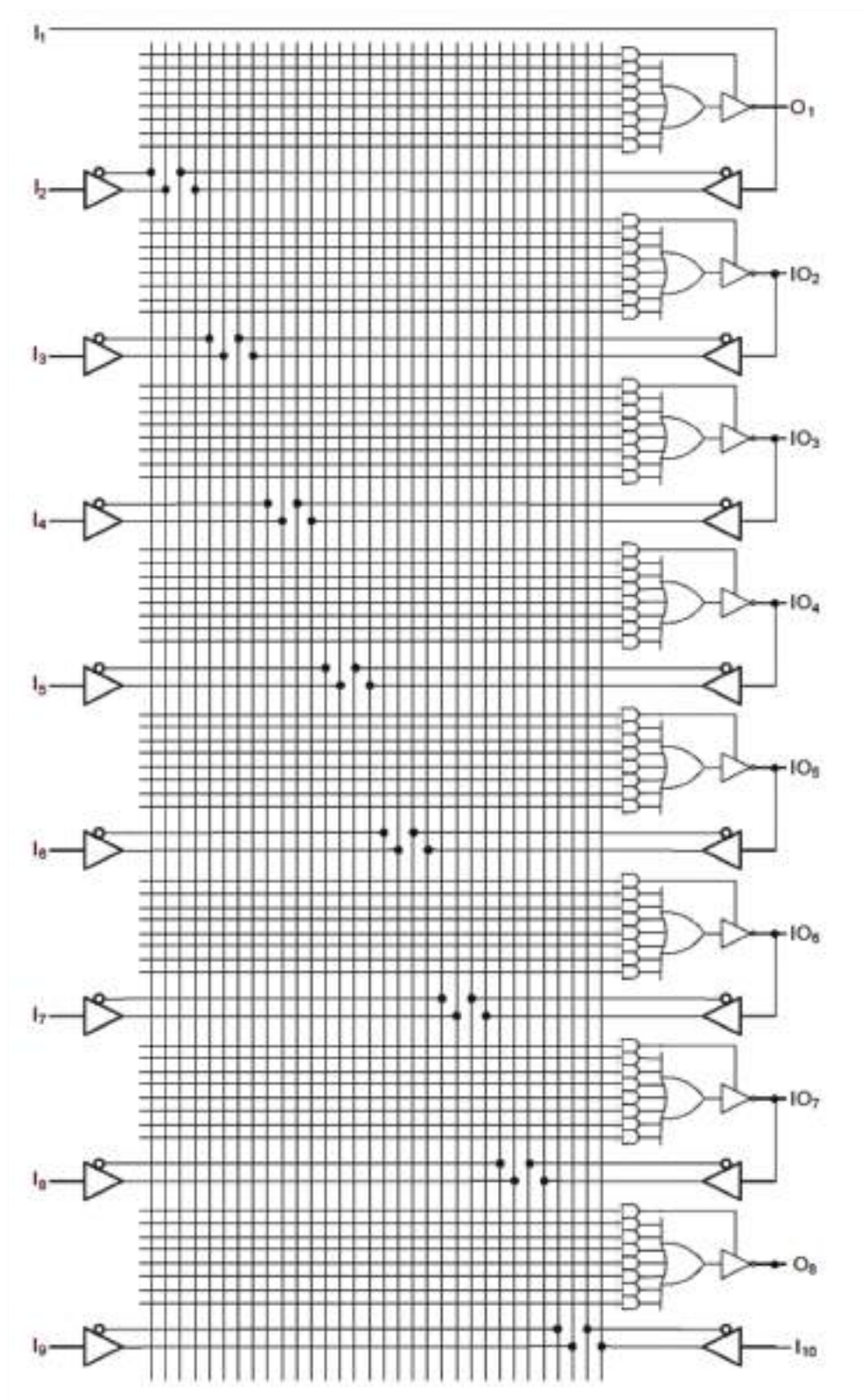


Fig. 8 - Arquitetura da PAL® 16L8

Os pinos bidirecionais podem ser usados de várias formas, dependendo da função que se atribui ao pino:



- Pode ser usado como entrada. Para tal, é necessário que a porta three-state de saída esteja sempre inativa;
- Pode ser usado como saída. Para tal, basta ativar o three-state. Neste caso, o sinal à saída pode ser realimentado para a matriz, através do *buffer* de realimentação, como se se tratasse de uma entrada. Esta realimentação é usada sempre que o número de termos de produto de uma célula não é suficiente para implementar uma função, como foi visto na secção anterior;
- Pode ser usado como pino de entrada/saída. Nesta configuração, o estado do pino é controlado pela porta three-state de saída. Quando se quer enviar dados para o exterior, ativa-se a porta three-state. No caso de se querer receber dados pelo pino, desativa-se a porta three-state e a entrada de dados é feita através do *buffer* de realimentação;
- Uma outra configuração interessante, é a de realimentação da função de saída para a mesma célula que produziu a saída. Isto permite implementar circuitos lógicos com realimentação.

Para ilustrar a utilização da PAL16L8 na implementação de funções lógicas combinatórias, vamos implementar o multiplicador de dois bits, a partir das suas funções complementares que se apresentam de seguida (não esquecer que a saída da célula inverte a função):

$$\overline{M3} = \overline{I3} + \overline{I2} + \overline{I1} + \overline{I0}$$

$$\overline{M2} = \overline{I3} + \overline{I1} + I2 \cdot I0$$

$$\overline{M1} = \overline{I3} \cdot \overline{I1} + \overline{I1} \cdot \overline{I0} + \overline{I3} \cdot I2 + I2 \cdot \overline{I0} + I3 \cdot I2 \cdot I1 \cdot I0$$

$$\overline{M0} = \overline{I2} + \overline{I0}$$

Estas funções são facilmente implementáveis na PAL® sem recorrer à realimentação, uma vez que o número de termos de produto de qualquer uma das funções é inferior a 7, ver implementação na figura seguinte.



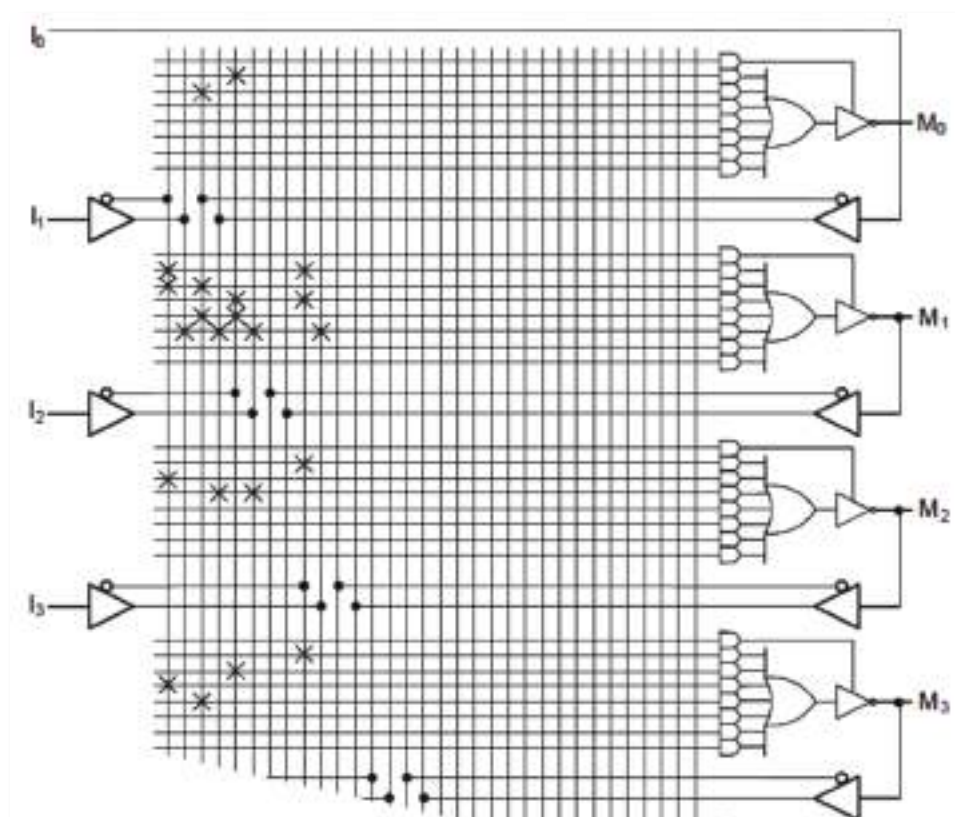


Fig. 9 - Implementação do multiplicador de 2-bits na PAL® 16L8

No exemplo, todos os three-states associados às saídas das funções têm de estar ativos. Para tal, basta não ligar qualquer entrada às AND de controlo respetivas.

Arquitetura de PAL® sequencial PAL16R8

A PAL® 16L8 apenas implementa funções combinatórias porque não tem elementos de memória, como latches ou flip-flops. Com o objetivo de implementar circuitos sequenciais em PAL®, surgiu a primeira geração de PAL® sequenciais com a designação PAL16R8. Este dispositivo programável tem 8 entradas e 8 saídas, uma entrada de relógio comum a todos os flip-flops tipo D e uma entrada de controlo comum a todas as portas three-state de saída, ver a figura seguinte.



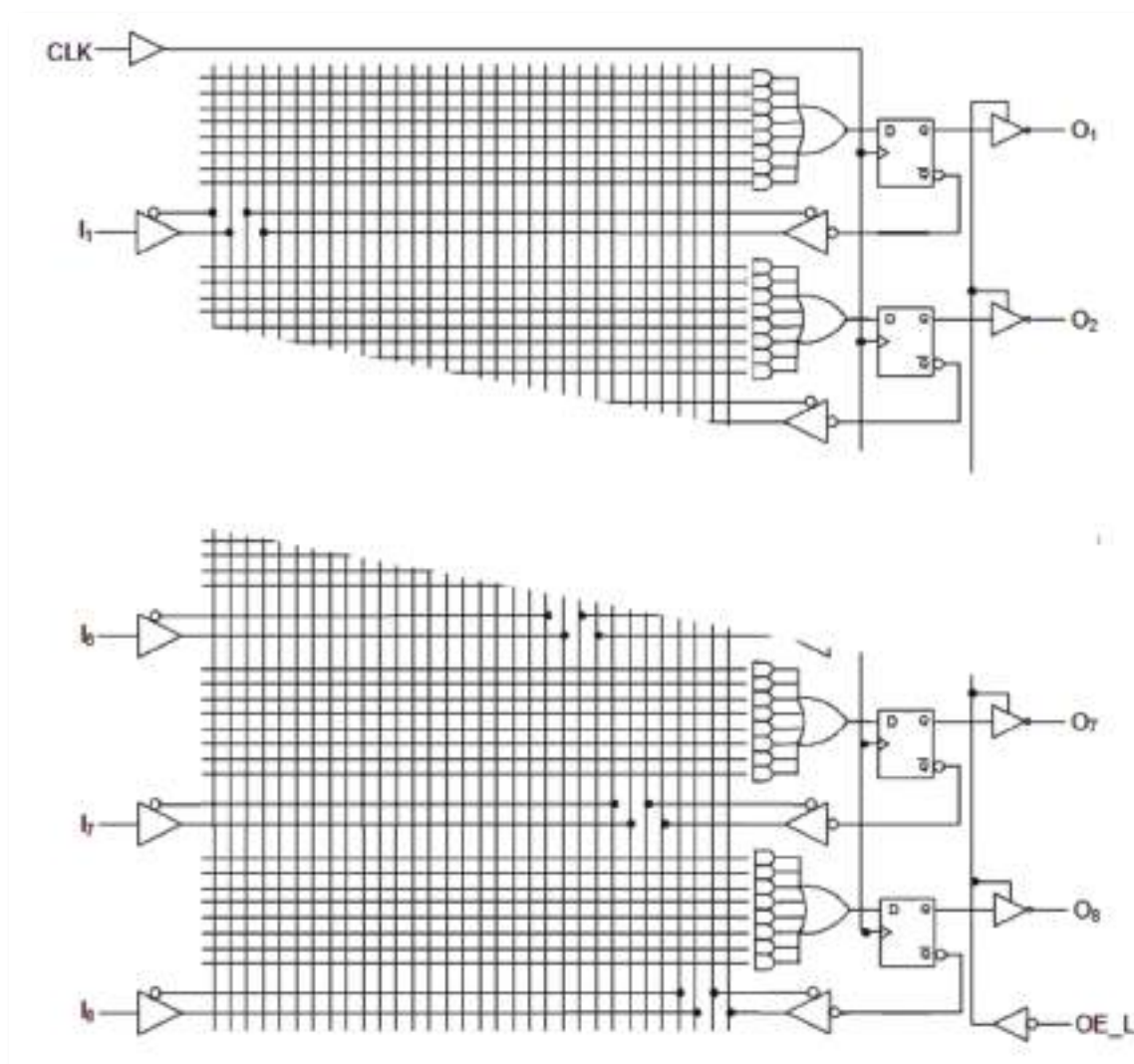


Fig. 10 - Arquitetura da PAL® 16R8

As saídas complementares dos flip-flops tipo D são realimentadas para a malha, facilitando a implementação de contadores, de registros de deslocamento, etc. Note-se que as saídas dos flip-flops estão disponíveis sem passar pelas portas three-state. Assim, os flip-flops podem mudar para um outro estado função do estado presente, independentemente do estado das portas three-state, porque as saídas são realimentadas antes dos three-states.

Apesar da sua aplicabilidade, as PAL® 16L8 e 16R8 estão limitadas pelo facto de apenas poderem ser usadas unicamente na implementação de circuitos combinatórios ou de circuitos sequencias e nunca em simultâneo. Uma vez que muitas aplicações necessitam de saídas combinatórias juntamente com saídas sequenciais, surgiram depois variantes destas PAL® em que algumas das saídas eram combinatórias, a célula não tinha flip-



flop, e outras eram sequências, a célula continha um flip-flop (e.g., PAL16R6). Apesar da sua aplicabilidade, o facto de uma saída estar à partida definida como combinatória ou sequencial restringia consideravelmente o tipo de PAL® ao projeto. Para ultrapassar esta limitação, surgiram as famílias de PAL® em que cada macrocélula podia ser configurada individualmente como combinatória ou como sequencial.

Nas secções seguintes, descrevem-se duas destas PAL®, nomeadamente a ATF22V10 e a ATF750C.

Arquitetura de PAL® sequencial ATF22V10

A ATF22V10 tem as seguintes características, ver a figura seguinte:

- 12 pinos de entrada dedicados e 10 pinos de saída que podem ser configurados como entrada, saída ou entrada/saída. As entradas não utilizadas devem ser ligadas a Vcc ou a GND;
- 10 macrocélulas programáveis como combinatórias ou sequenciais e ativas a HIGH ou ativas a LOW;
- Número de termos de produto por macrocélula variável de 8 a 16 termos. Pela figura 11 é possível identificar o número de termos de produto associados a cada uma das macrocélulas através do número escrito na linha de entrada. Por exemplo, a macrocélula associada à saída IO1 tem 8 termos de produto, enquanto que a associada à saída IO4 tem 14 termos de produto;
- Cada um dos flip-flops recebe sinais comuns de relógio (activo no flanco ascendente), de reset assíncrono e de preset síncrono;
- Os termos de produto com todas as ligações em aberto assumem o estado lógico HIGH;
- As saídas incluem um *buffer* three-state controlado por um termo de produto. Sempre que o pino é usado para entrada, o three-state é configurado por forma a estar sempre inativo. Caso o pino seja configurado como saída, o three-state está sempre ativo. É ainda importante referir que a realimentação combinatória é feita a partir da saída do three-state. Como tal, a realimentação combinatória só é possível se o pino for configurado como saída. Através do controlo do three-state, é possível usar um pino I/O como bidirecional.



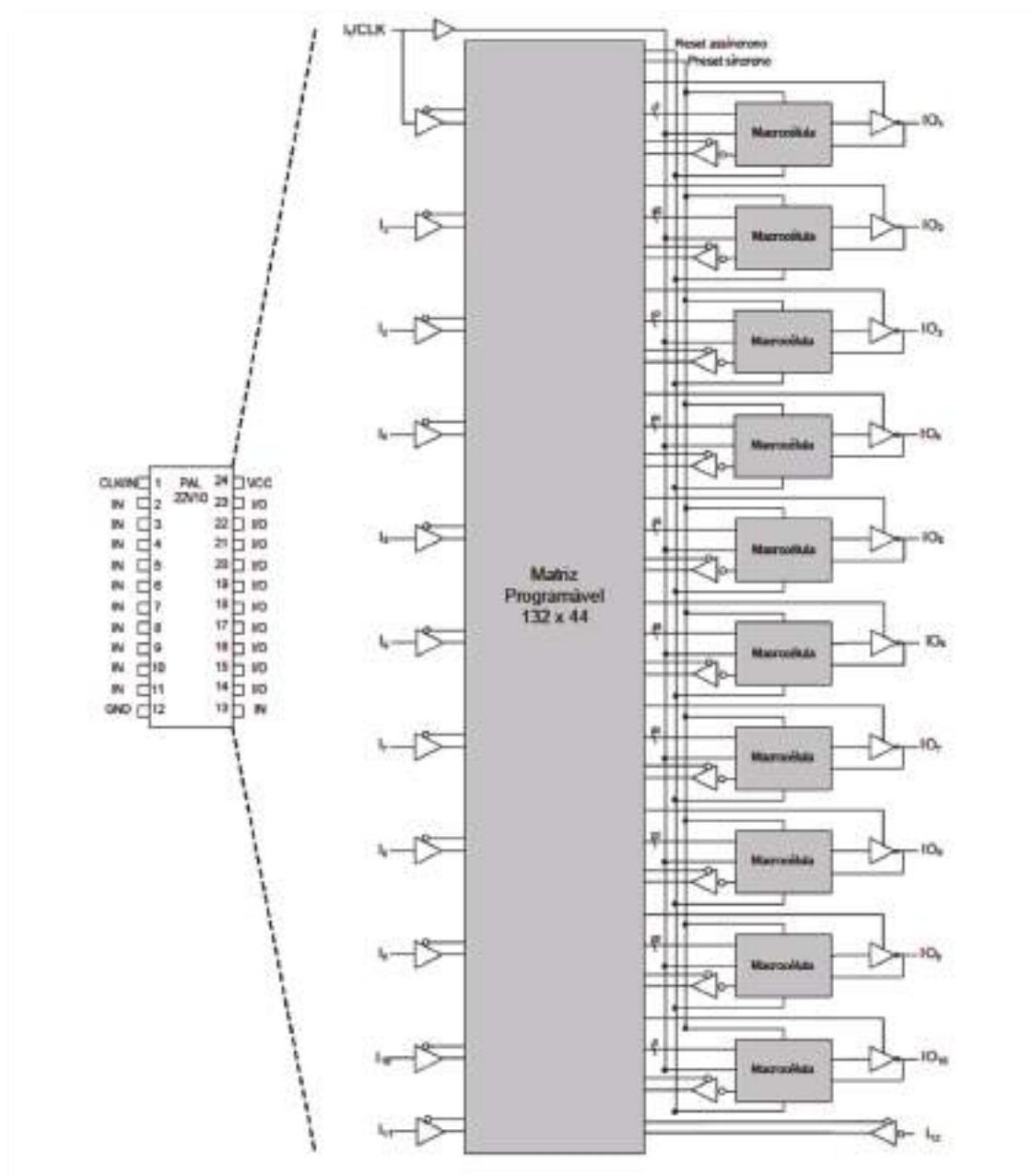


Fig. 11 - Arquitetura da PAL® ATF22V10

A macrocélula de saída pode ser configurada de acordo com uma de quatro configurações diferentes: saída combinatória ativa a HIGH, saída combinatória ativa a LOW, saída sequencial ativa a HIGH ou saída sequencial ativa a LOW. A escolha da configuração é feita de acordo com a aplicação do utilizador e é conseguida através de dois bits de configuração: S0 e S1, ver a figura seguinte.



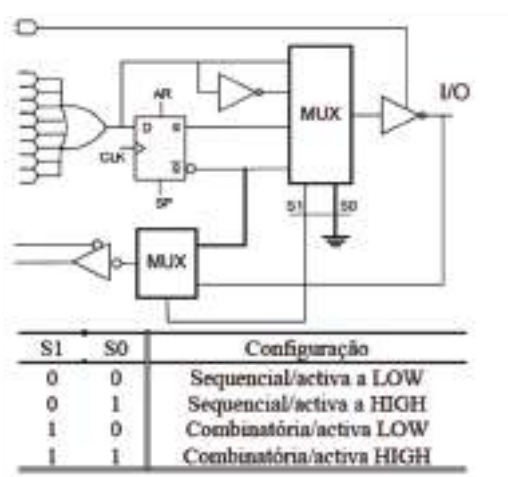


Fig. 12 - Arquitetura da macrocélula da PAL® 22V10

A configuração sequencial ativa a LOW é conseguida com ambos os bits de S1 e S0 ao nível lógico 0. Neste caso, o multiplexer de saída coloca à entrada do three-state o sinal proveniente da saída Q do flip-flop. Por outro lado, o MUX de entrada (em baixo, na figura), realimenta a matriz programável com a saída complementada do flip-flop. Na mesma figura, está ilustrado o circuito lógico equivalente após a configuração da macrocélula.

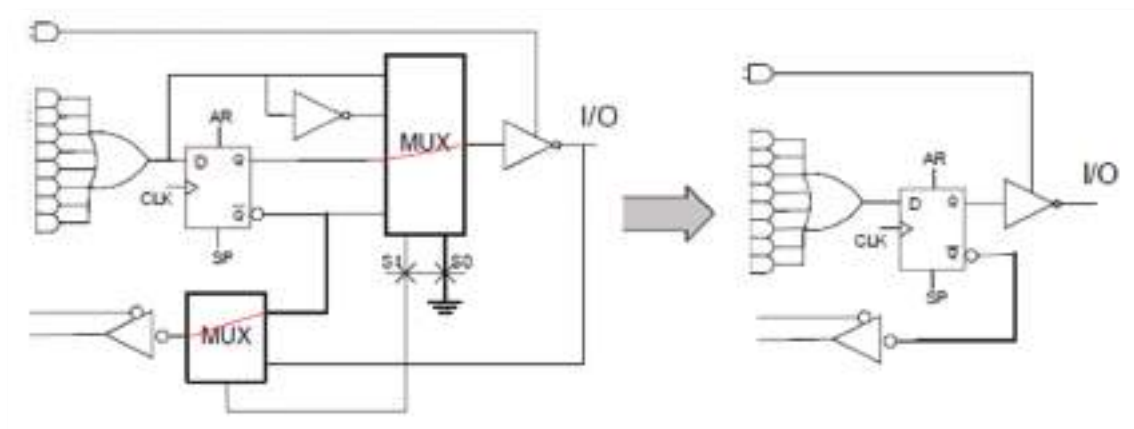


Fig. 13 - Configuração sequencial ativa a LOW da macrocélula da ATF22V10

Para obter uma saída registrada ativa a HIGH, o multiplexer encaminha a saída complementada do flip-flop que depois é invertida pelo three-state, ver a figura seguinte.



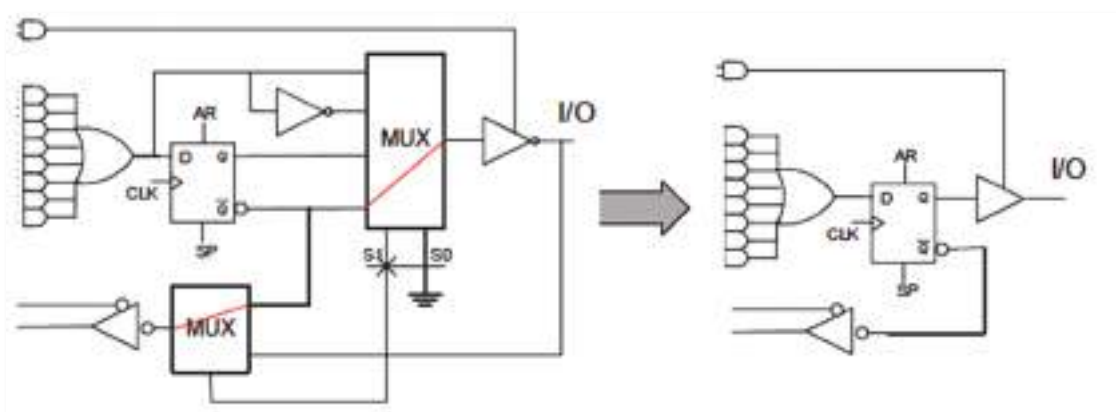


Fig. 14 - Configuração sequencial ativa a HIGH da macrocélula da ATF22V10

Para ter saídas combinatórias, basta selecionar as entradas do multiplexador que não passam pelo flip-flop. Para a saída ativa a LOW, seleciona a entrada negada e o multiplexador de entrada seleciona o sinal vindo da saída do three-state, ver a figura seguinte.

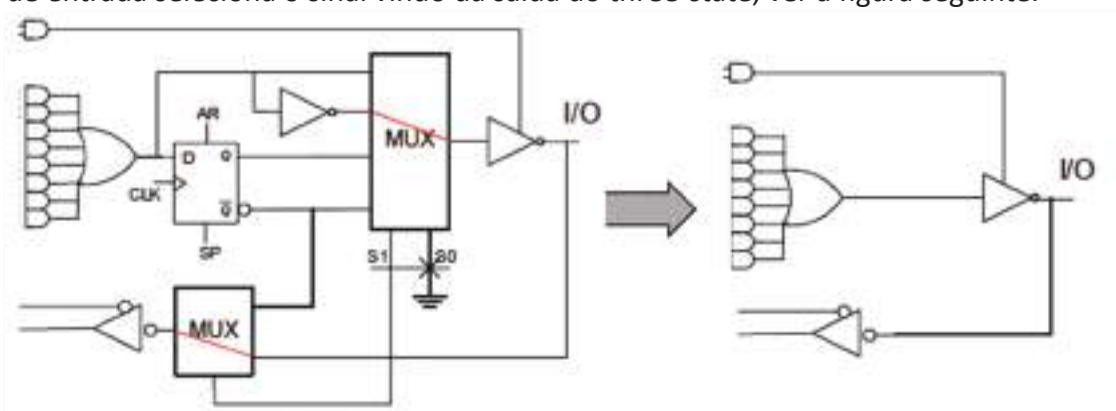


Fig. 15 - Configuração combinatória ativa a LOW da macrocélula da ATF22V10

Para a saída ativa a HIGH, seleciona a entrada não negada e o multiplexador de entrada seleciona o sinal vindo da saída do three-state, ver a figura seguinte.

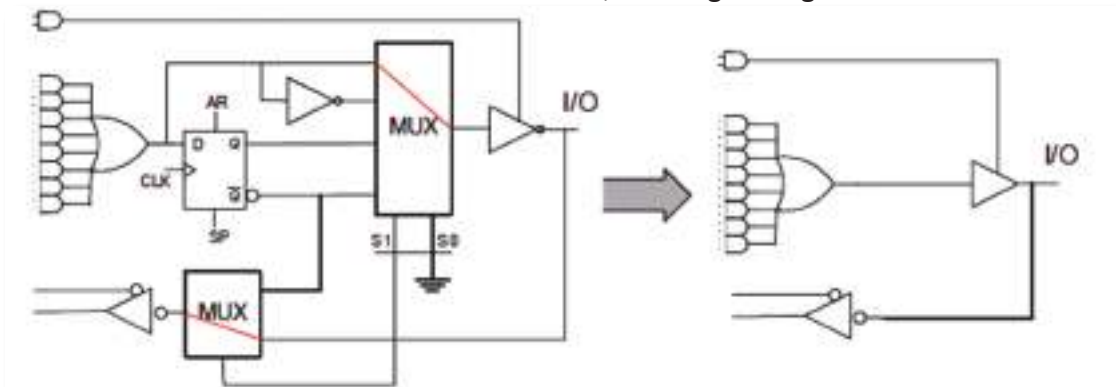


Fig. 16 - Configuração combinatória ativa a HIGH da macrocélula da ATF22V10



A figura 17 ilustra a arquitetura completa da PAL® ATF22V10.

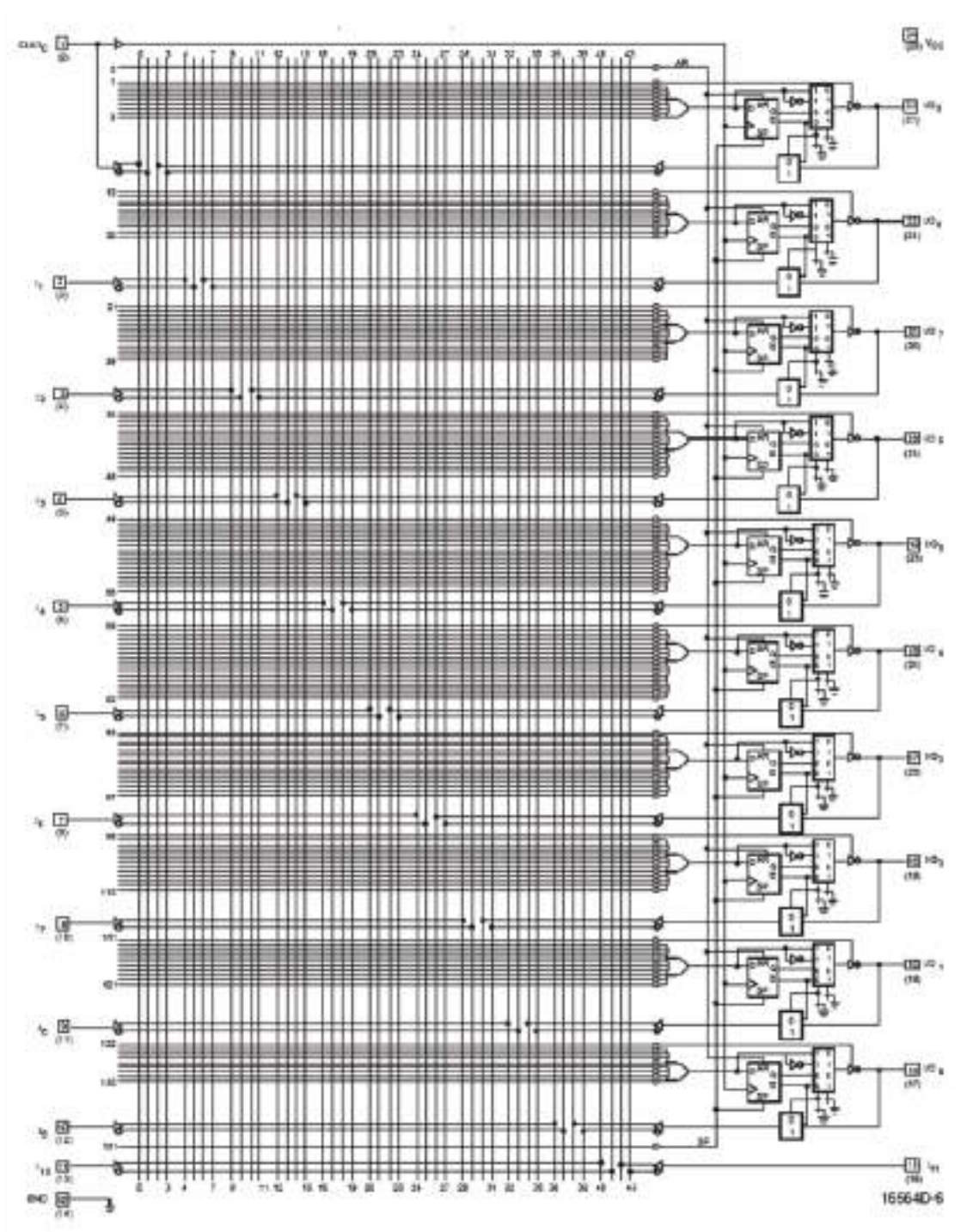


Fig. 17 - Arquitetura completa da PAL® ATF22V10

Arquitetura de PAL® sequencial ATF750C/CL

A PAL® ATF22V10 foi um passo em frente na flexibilização da arquitetura da PAL®. No entanto, ainda apresenta algumas limitações, como o número reduzido de flip-flops,



um sinal de relógio comum a todos os flip-flops, inviabilizando a implementação na mesma PAL[®] de máquinas sequencias com relógios diferentes, tipo de flip-flop fixo, não permitindo otimizar a lógica combinatória com a utilização de flip-flops mais adequados ao problema, etc. Com vista a melhorar todos estes aspetos, surgiu recentemente uma nova família de PAL[®] - ATF750C/ATF750CL - mais flexível e com maior densidade de integração que ultrapassa as limitações descritas anteriormente. A ATF750C(L) tem as seguintes características principais:

- É uma extensão à lógica da 22V10 e é compatível com as PAL[®] ATF750B/BL e ATF750/L (versões anteriores desta família de PAL[®]);
- Tem 12 pinos de entrada dedicados e 10 pinos de saída que podem ser configurados como entrada, saída ou entrada/saída;
- Tem 20 flip-flops que podem ser configurados individualmente como sendo do tipo D ou do tipo T. As saídas dos flip-flops podem realimentar a matriz programável de entrada. Todos os flip-flops são inicializados a 0 após ativação da alimentação;
- Tem 10 macrocélulas programáveis como combinatórias ou sequenciais com termos de soma combinados ou separados;
- Tem um total de 171 termos de produto e dois termos de soma por saída. Os termos de soma têm associados entre quatro e oito termos de produto. Pela figura 18 é possível identificar o número de termos de produto associados a cada uma das macrocélulas através do número escrito na linha de entrada;
- O sinal de relógio e de reset de cada um dos flip-flops pode ser controlado individualmente por um termo de produto. Adicionalmente, cada um dos flip-flops pode ser configurado individualmente com uma entrada de relógio proveniente do pino de entrada direta de relógio. O preset de todos os flip-flops é controlado por um sinal síncrono comum a todos provenientes de um termo de produto;
- Os termos de produto com todas as ligações em aberto assumem o estado lógico HIGH;
- As saídas incluem um *buffer* three-state controlado por um termo de produto. Sempre que o pino é usado para entrada, o three-state é configurado por forma a estar sempre inativo. Caso o pino seja configurado como saída, o three-state



está sempre ativo. É ainda importante referir que a realimentação combinatória é feita a partir da saída do three-state. Como tal, a realimentação combinatória só é possível se o pino for configurado como saída. Através do controlo do three-state, é possível usar um pino I/O como bidirecional. Os pinos de saída incluem ainda um controlo para programação da polaridade.

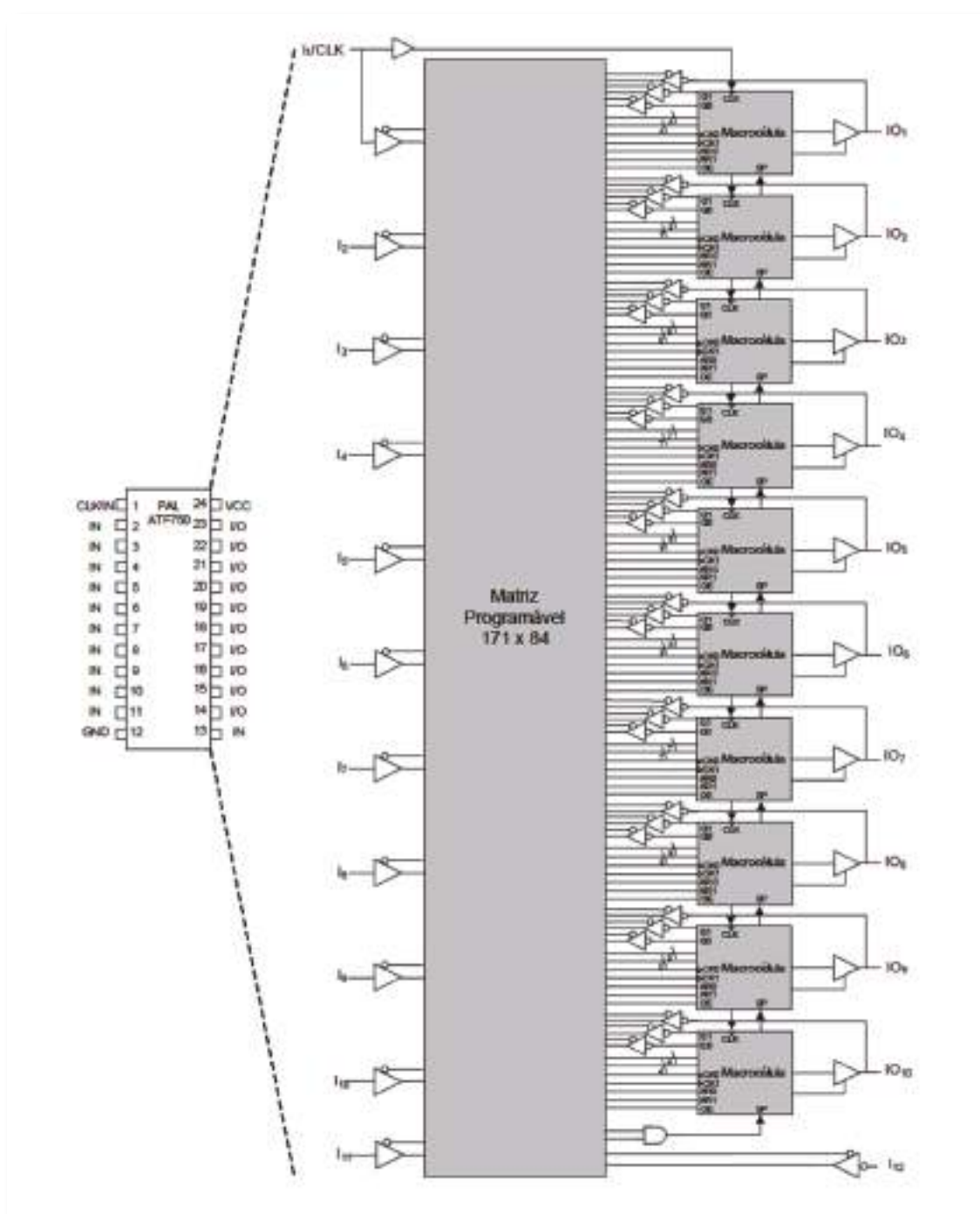


Fig. 18 - Arquitetura completa da PAL® ATF750C/CL



Quando comparado com a PAL® 22V10, a ATF750C(L) tem o dobro da densidade lógica, incluindo o dobro do número de flip-flops, e é mais flexível. A flexibilidade traduz-se por ter sinais de relógio e sinais de reset independentes para cada um dos flip-flops e caminhos de realimentação a partir das saídas dos flip-flops que permitem a utilização destes sem utilizar pinos de entrada/saída. Além disso, ao poder configurar os flip-flops como sendo do tipo D ou do tipo T facilita a implementação de contadores neste tipo de PAL®.

A macrocélula da ATF750C(L) é formada pelos termos de produto (o número de termos de produto da macrocélula depende do pino a que esta associada. Na figura seguinte é possível identificar o número de termos de produto associados a cada uma das macrocélulas através do número escrito na linha de entrada. Por exemplo, a macrocélula associada à saída IO1 tem 4 termos de produto associados a cada um dos termos de soma, por dois termos de soma, por dois flip-flops e por lógica de programação para configurar a célula como combinatória ou sequencial, bem como o tipo de polaridade associado à saída. Existe ainda um multiplexer que seleciona a origem do sinal de relógio aplicado a cada um dos flip-flops, ver a figura seguinte.

Na figura, é possível identificar os termos de produto associados a cada um dos termos de soma, os dois flip-flops e o three-state de saída. A escolha entre sequencial e combinatório é feita pelo multiplexer de saída através do seletor Sel1 configurável.

Dependendo do valor deste seletor, a saída vem do flip-flop (sequencial) ou do termo de soma (combinatório).

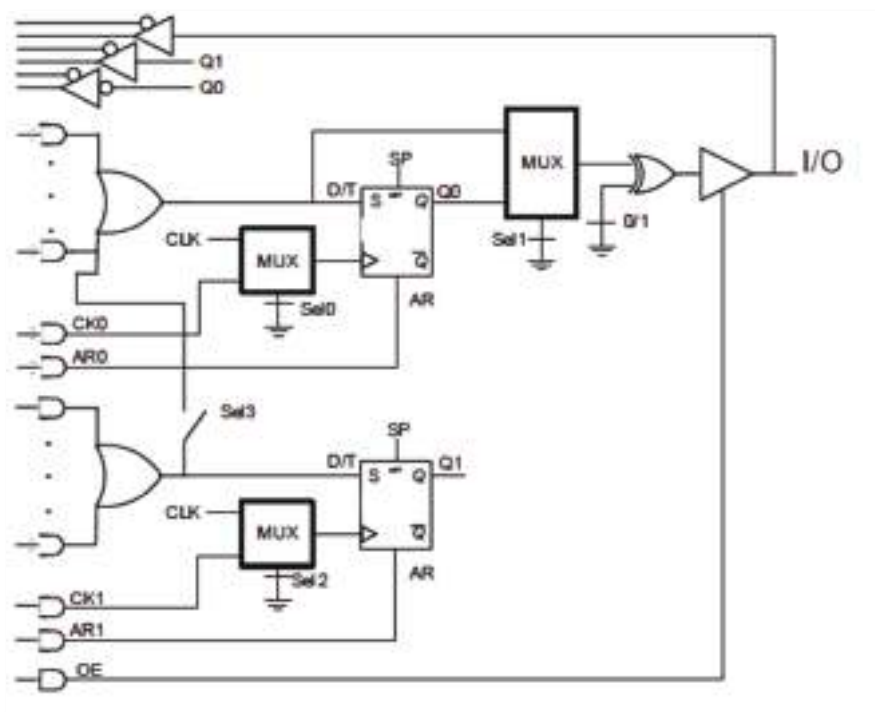


Fig. 19 -
Arquitetura da
macrocélula de
saída da PAL®
ATF750C/CL



O número de termos de produto do termo de soma superior pode ser aumentado com a utilização dos termos de produto associados ao termo de soma inferior. Para tal, basta ligar o seletor Sel3. A origem do sinal de relógio é controlada pelo multiplexer ligado à entrada de relógio dos flip-flops que escolhe entre o sinal de relógio síncrono ligado ao pino 1 da PAL® (CLK), comum a todos os flip-flops, ou o sinal gerado por um termo de produto (CLK0 e CLK1).

A polaridade de saída é configurada através de uma porta XOR que funciona como *buffer* inversor ou não inversor. Os sinais de reset assíncrono (AR0 e AR1) e de controlo do three-state de saída (OE) são provenientes de um termo de produto.

Todas as células têm três caminhos de realimentação para a matriz configurável (representado na zona superior da figura 20), um vindo de cada um dos flip-flops e outro do pino de saída. No caso de uma saída combinatória, a realimentação vem sempre do próprio pino. No caso de uma saída sequencial, a realimentação pode vir do pino ou do registo.

A macrocélula da ATF750C/CL permite várias configurações. De seguida, descrevemos algumas das mais utilizadas:

- Saída combinatória;
- Saída combinatória mais um flip-flop;
- Saída sequencial;
- Saída sequencial mais um flip-flop;
- Utilização de dois flip-flops com pino de I/O usado como entrada;
- Saída combinatória com registo do valor no flip-flop.

Em todas as configurações, um flip-flop não associado a um pino de saída é designado flip-flop interno. Na PAL® ATF750 o flip-flop Q1 é sempre interno, enquanto o flip-flop Q0 pode ser interno ou estar associado a um pino.

Configuração com saída combinatória.

Esta é a configuração mais simples, em que não são utilizados flip-flops, ver a figura seguinte.



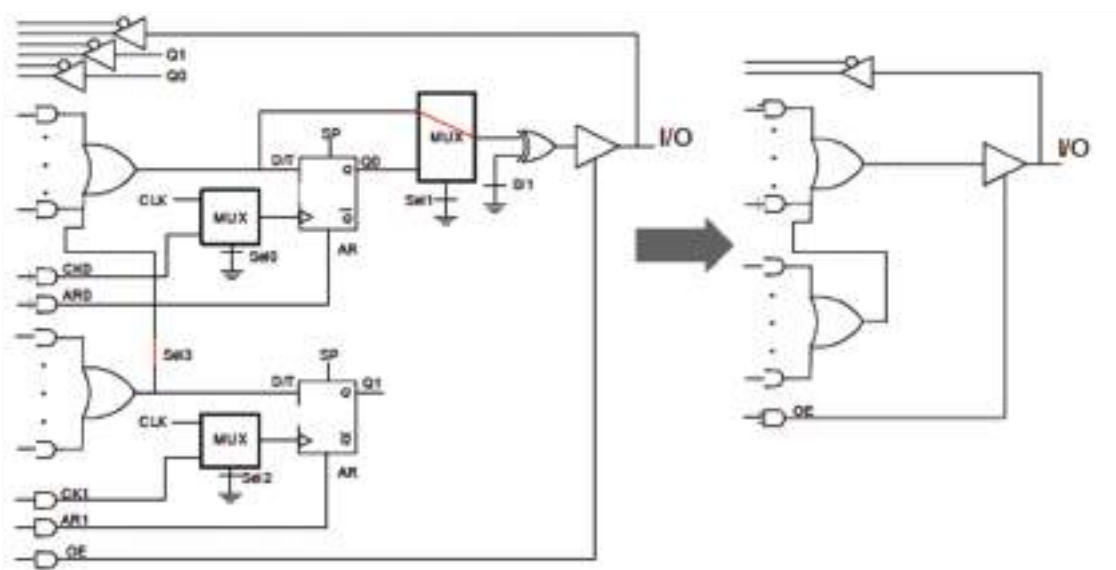


Fig. 20 - Configuração com saída combinatória

O multiplexer de saída é configurado para a entrada combinatória. O Sel3 é ligado caso sejam necessários mais termos de produto para produzir a saída combinatória. A utilização dos termos de produto provenientes do termo de soma inferior é possível porque o segundo flip-flop não está a ser usado.

Configuração com a saída combinatória e mais um flip-flop.

Nesta configuração, é produzida uma saída combinatória com o termo de soma superior, e o termo de soma inferior é usado com o seu flip-flop para produzir uma variável Q1 registada, ver a figura seguinte.

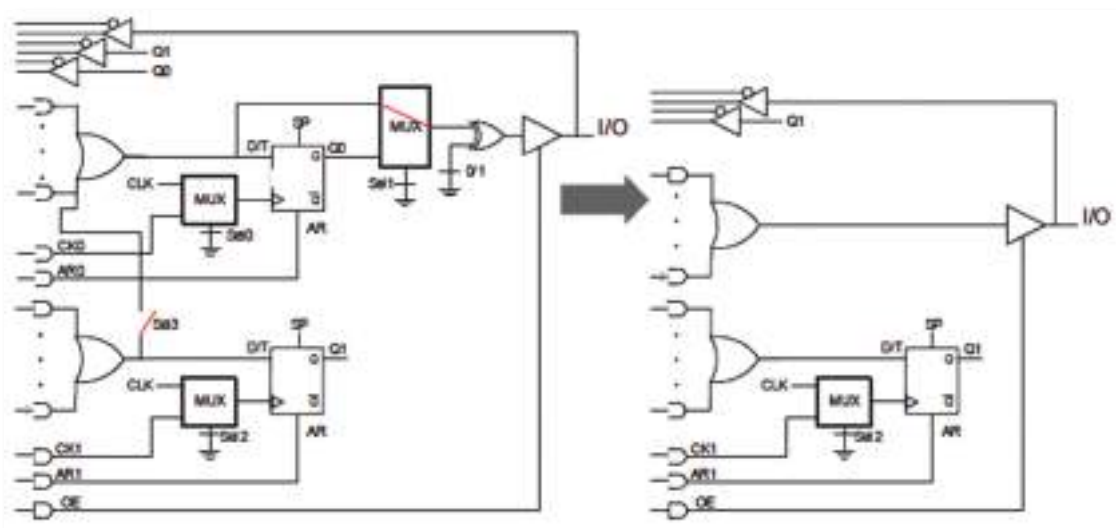


Fig. 21 - Configuração com saída combinatória e mais um flip-flop



Nesta configuração não é possível associar os termos de produto do termo de soma inferior aos do termo de soma superior, uma vez que estão a ser usados pelo flip-flop inferior. Esta configuração pode ser usada, por exemplo, quando o circuito a implementar inclui uma saída combinatória e uma variável registada que não tem de ser enviada para um pino de saída.

Saída sequencial

O modo mais simples de gerar uma saída sequencial é utilizando o flip-flop superior, que tem um caminho direto para o pino de saída. Também é possível usar o flip-flop inferior, mas neste caso a saída do flip-flop não tem um caminho direto para uma saída, tendo de ser realimentado para a matriz para poder aparecer numa saída combinatória, ver a figura seguinte.

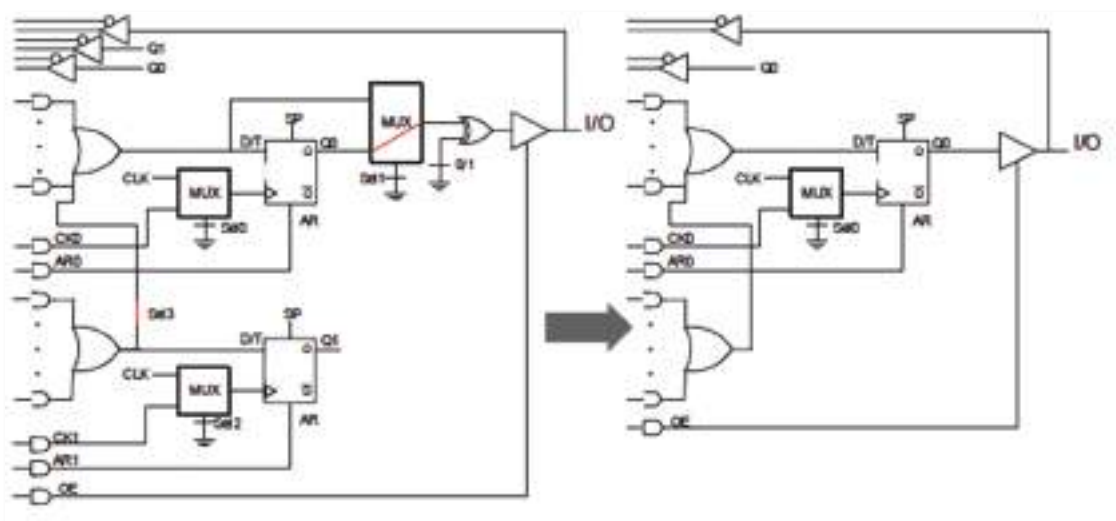


Fig. 22 - Configuração com saída sequencial

Também nesta configuração se podem adicionar os termos de produto inferiores ao OR superior, uma vez que o flip-flop inferior não está a ser usado.

Saída sequencial mais um flip-flop.

Esta configuração usa o flip-flop superior para gerar uma saída registada e o inferior para registar uma variável interna, ver a figura seguinte.



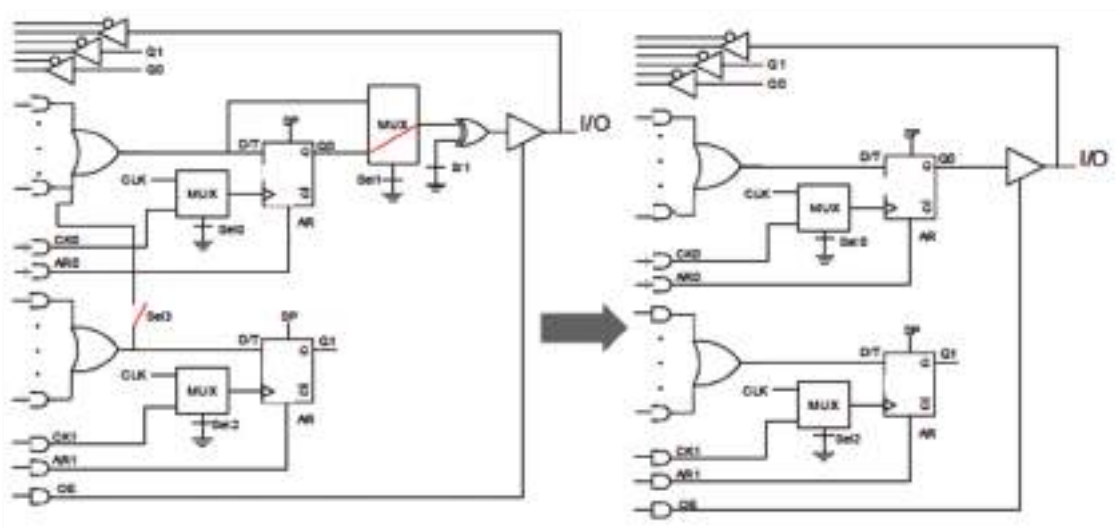


Fig. 23 - Configuração com saída sequencial mais um flip-flop interno

Utilização de dois flip-flops com pino de I/O usado como entrada.

A ATF750C/CL tem a vantagem de ter realimentações diretamente das saídas dos flip-flops. Como tal, os flip-flops podem ser usados sem estar associados a qualquer pino I/O, libertando o pino I/O para poder ser usado como entrada, ver a figura seguinte.

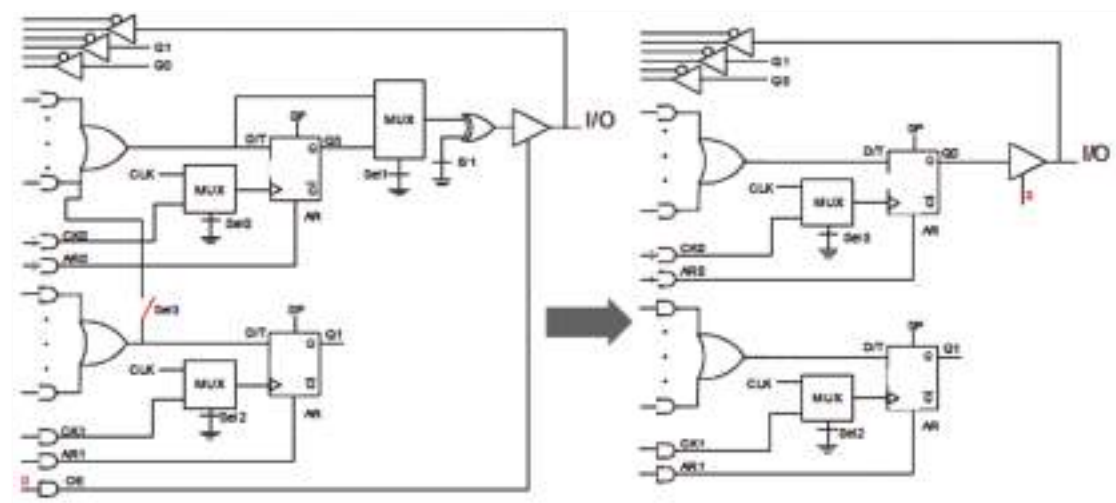


Fig. 24 - Configuração com dois flip-flops internos e o pino usado como entrada combinatória

Repare que nesta configuração o three-state de saída está sempre inativo (entrada de controlo a 0). Isto permite que se utilize o pino como entrada sem gerar qualquer conflito com o valor à saída do flip-flop superior.



Saída combinatória com registo do valor no flip-flop.

Esta última configuração ilustra a utilização de uma macrocélula com saída combinatória, em que o valor de saída é registado no flip-flop respetivo, ver a figura seguinte.

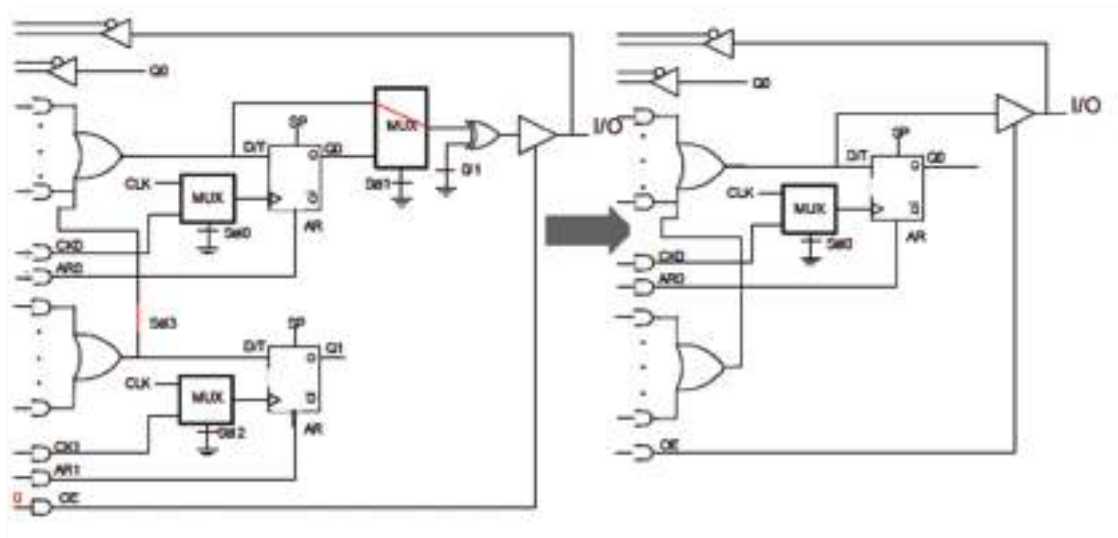


Fig. 25 - Configuração com saída combinatória registada num flip-flop interno

Neste caso, o segundo flip-flop pode ser usado ou não. No exemplo da figura não foi utilizado, permitindo usar os seus termos de produto no termo de soma superior.



Programação e teste de um circuito utilizando uma PAL

Esta secção descreve o modo de configuração das PAL® recorrendo à linguagem de descrição de *hardware* Atmel-CUPL. Nesta descrição, vamos considerar apenas as PAL® ATF22V10 e ATF750C/CL.

Configuração da PAL® ATF22V10 usando CUPL

Recordando a macrocélula da ATF22V10 (ver figura 26), verifica-se que na sua configuração tem de se indicar se se trata de uma saída combinatória ou sequencial com flip-flop tipo D. No caso de ser uma saída sequencial, é ainda necessário configurar os sinais ligados às entradas AR, SP e D do flip-flop. Para qualquer tipo de saída, tem de se indicar a função lógica associada à entrada de controlo do three-state de saída. Por omissão, o valor é colocado no valor lógico 1.

O sinal de relógio não exige qualquer configuração pois é comum a todos os flipflops e está sempre associado ao pino 1 da PAL®.

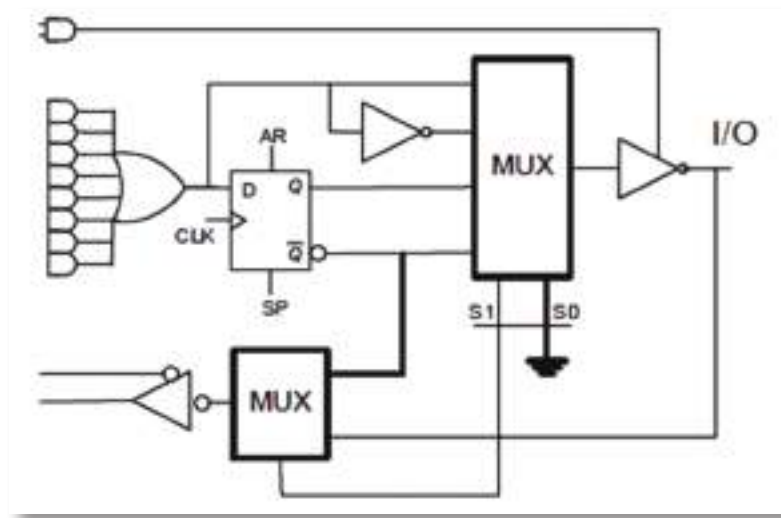


Fig. 26 - Macrocélula da ATF22V10

Deste modo, as extensões CUPL válidas para este dispositivo programável são: OE (controlo do three-state de saída), AR, SP e D (sinais associados ao flip-flop).



Para configurar a saída como combinatória, basta indicar a função lógica associada a essa saída. Por exemplo:

$O21 = I1 \# I2;$

Para configurar a saída como sequencial, é necessário indicar as expressões lógicas associadas à entrada do flip-flop. Por exemplo:

$O21.d = I1 \# I2;$

$O21.ar = reset;$

$O21.sp = 'b'0;$

Adicionalmente, pode especificar-se um sinal de controlo do three-state de saída, quer seja uma saída combinatória, quer seja sequencial. Por exemplo:

$O21.oe = enable;$

A indicação de que se pretende usar o valor presente numa saída combinatória ou sequencial para realimentação da matriz é feita utilizando o nome associado ao pino no lado direito de uma expressão sem qualquer extensão. Por exemplo:

$O22 = O21 \& I2;$

Significa que a saída combinatória O22 é igual ao produto lógico entre a entrada I2 e a função lógica da saída O21.

Configuração da PAL® ATF750C/CL usando CUPL

A programação da macrocélula da PAL® ATF750 inclui a configuração do pino como entrada, saída ou entrada/saída e do sinal de controlo do three-state de saída. Além disso, um pino de saída ou de entrada/saída pode ser configurado como combinatório ou sequencial. Sempre que se utiliza um flip-flop, é necessário configurar as entradas de reset (AR), preset (SP) e de relógio, bem como o tipo de flip-flop (D ou T). Deste modo, as extensões CUPL válidas para este dispositivo são: OE (controlo do three-state de saída), AR, SP, D, T, DFB (sinais associados aos flip-flops), CK, CKMUX (sinais associados à configuração do sinal de relógio) e IO (sinal associado à identificação da realimentação do pino). Para melhor percebermos a configuração da macrocélula, recorda-se a sua configuração na figura seguinte.



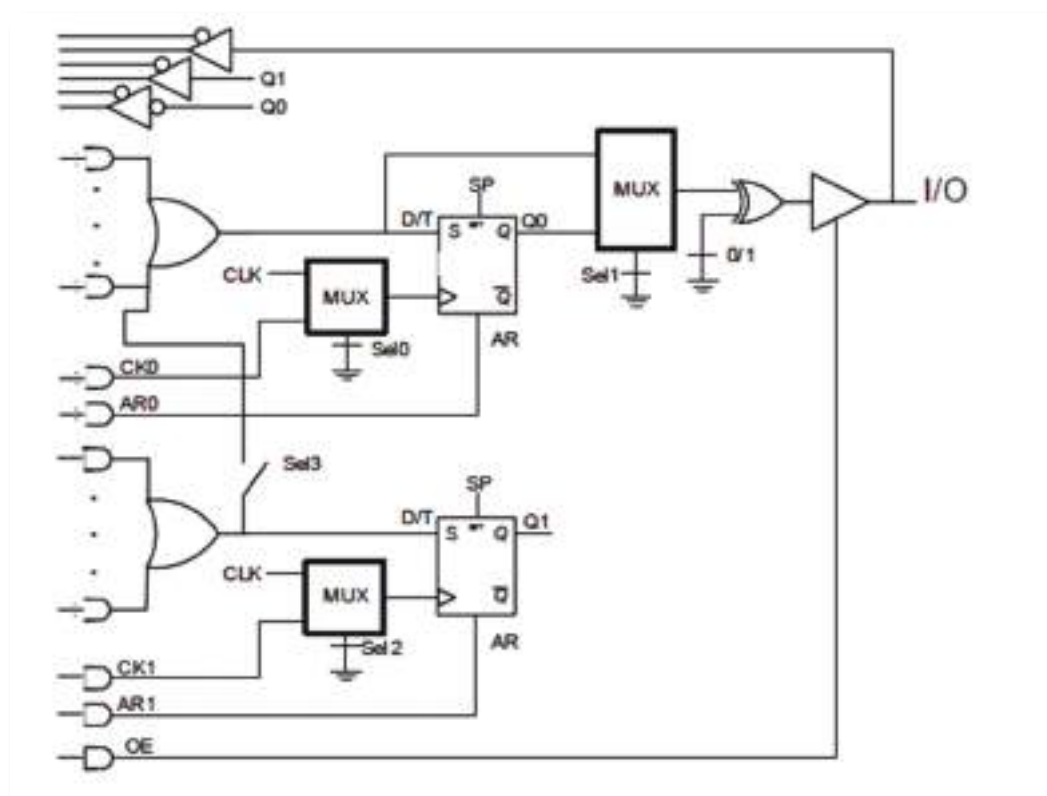


Fig. 27 - Macro célula da ATF750X

Atribuição de nós e pinos.

O flip-flop Q1 interno é identificado por um número de nó. O flip-flop Q0 é identificado por um número de nó quando usado como interno e um número de pino quando associado a um pino, ver na tabela seguinte.

Na tabela, uma linha corresponde a uma macrocélula. Por exemplo, o registo Q0 do pino 17 (ou do nó 38) e o registo Q1 do nó 28 pertencem à mesma macrocélula.

Pino Q0	Nó Q0	Nó Q1
14	35	25
15	36	26
16	37	27
17	38	28
18	39	29
19	40	30
20	41	31
21	42	32
22	43	33
23	44	34

Tabela 1 – identificação dos nós e dos pinos



Considere os exemplos seguintes:

pin [2, 3, 4, 5] = [I1, I2, I3, I4];

pin [20, 21, 22, 23] = [O20, O21, O22, O23];

pinnode [34, 31, 44] = [O23Q1, O20Q1, O23Q0];

Na primeira declaração atribuíram-se quatro pinos de entrada. Depois, de acordo com a tabela 5, atribuíram-se quatro pinos 20, 21, 22 e 23 que podem ser usados como combinatórios ou sequencias. Na declaração seguinte atribuíram-se três nós internos, dois com o registo Q1 e um terceiro com o registo Q0. O primeiro registo Q1 e o registo Q0 definidos como nós pertencem à mesma macrocélula, de acordo com a tabela 5.1.

Identificação das realimentações.

Cada macrocélula tem três caminhos de realimentação para a matriz de entrada: um de cada um dos registos e um terceiro do pino. Para um flip-flop interno Q1, o caminho de realimentação é identificado pelo nome do nó. Para um flip-flop associado a um pino de saída, o caminho de realimentação pode vir do registo ou do pino. No primeiro caso, é identificado pelo nome do pino. No segundo caso, é identificado pelo nome do pino acompanhado da extensão IO, ver a figura seguinte.

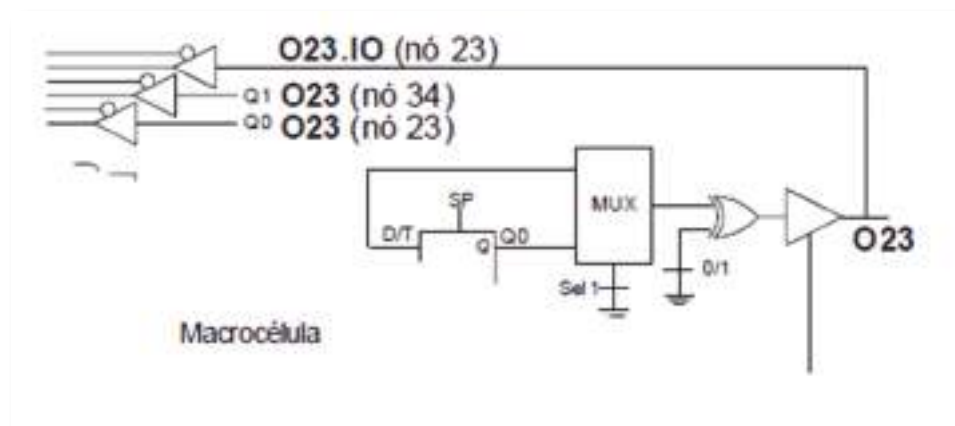


Fig. 28 - Identificação dos pontos de realimentação

Para uma saída combinatória, a realimentação vem do pino, pelo que é identificada pelo nome do pino. Por exemplo:

O22.d = I1 \$ I2;

O23Q1.d = I2 # I3 # I4;



O21 = O23 & O23Q1 & O23.io:

No exemplo, definiram-se as expressões de entrada dos registos Q0 (O23) e Q1 (O23Q1) de uma macrocélula. De seguida, atribuiu-se à saída O21 o produto lógico entre o valor da saída do registo Q0 (O23), o valor da saída do registo interno Q1 (O23Q1) e o valor do pino O23 (O23.io), todos pertencentes à mesma macrocélula.

Existe um caso particular de realimentação, associado à configuração f) descrita e ilustrada na secção anterior, correspondente à configuração em que temos uma saída combinatória cujo valor é registado no flip-flop associado a essa saída. Sendo uma saída combinatória, esta é identificada pelo nome do pino. Como tal, a realimentação a partir do flip-flop não pode usar o mesmo nome. Para ultrapassar esta dificuldade, utiliza-se a extensão. DFB associada ao nome do pino para identificar a saída do flip-flop Q0, ver a figura seguinte.

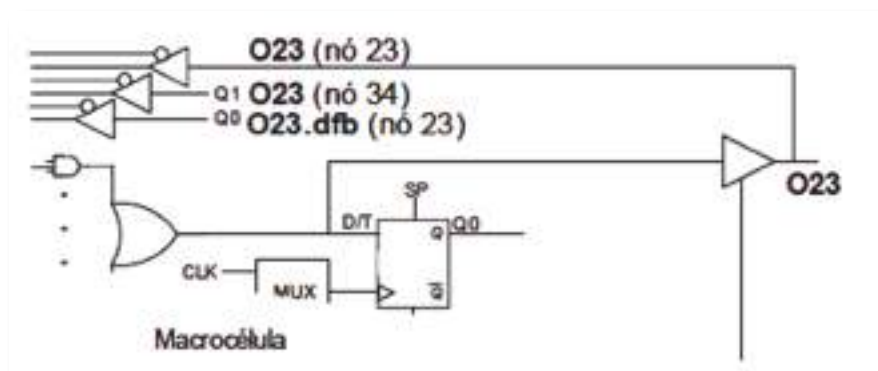


Fig. 29 - Identificação do ponto de realimentação com flip-flop e saída combinatória

Por exemplo,

```
O21 = I2;
```

```
O23.d = O21.dfb;
```

Neste exemplo, atribuiu-se a entrada I2 à saída O21 (saída combinatória) e atribuiu-se ao flip-flop Q1 (O23) o valor realimentado vindo da saída do flip-flop Q0 (O21.dfb) pertencente à macrocélula associada à saída combinatória O21.

Atribuição dos sinais AR, SP e OE.

Os sinais AR (reset assíncrono) dos flip-flops são configurados individualmente com um termo de produto. O sinal SP (preset síncrono) dos flip-flops é comum a todos e é



configurado como um único termo de produto. Os three-states de saída também são configurados individualmente através dum termo de produto. A configuração destes sinais faz-se com as extensões do CUPL .AR, .SP e .OE. Por exemplo:

O23.ar = I1 & I2;

O23.sp = I3;

O23.oe = I2 & I4;

Configuração dos sinais de relógio

Cada um dos registos pode ser configurado com um sinal de relógio independente proveniente de um termo de produto ou diretamente do pino de relógio (pino 1 da PAL®). A configuração do sinal de relógio é feita com as extensões .CK e .CKMUX. No caso de se pretender usar o sinal diretamente do pino de relógio, é necessário associar um nome ao pino 1 que depois é atribuído à entrada de relógio do flip-flop usando o nome do nó ou do pino a que está associado o flip-flop seguido da extensão .CKMUX. No caso de se pretender usar um relógio proveniente de um termo de produto, deve usar-se o nome do nó ou do pino a que está associado o flip-flop seguido da extensão .CK. por exemplo:

pin 1 = sync_clk;

pin 2 = async_clk;

O22.ckmux = sync_clk;

O23.ck = async_clk & I1;

O23Q1.ck = sync_clk;

Neste exemplo, o flip-flop associado à saída O22 recebe um sinal de relógio direto do pino 1. O flip-flop Q0 associado ao pino O23 (O23) recebe um sinal de relógio vindo de um termo de produto (async_clk & I1). O flip-flop Q1 associado ao nó O23 (O23Q1) recebe um sinal de relógio de um termo de produto, que neste exemplo é formado apenas pelo sinal aplicado ao pino 1 (sync_clk). Repare que, embora o O22 e o O23Q1 recebam o mesmo sinal de relógio, o primeiro recebe o sinal diretamente, enquanto o segundo recebe o sinal através de um termo de produto.



Identificação do tipo de flip-flop.

Na PAL® ATF750, o flip-flop pode ser configurado como sendo do tipo D ou do tipo T. Para tal, basta usar a extensão .D ou .T, respetivamente, quando se definem as funções de entrada dos flip-flops. Por exemplo,

$O23.d = I1 \& I2;$

$O22.t = I1 \& I2;$

O primeiro caso configura o flip-flop como tipo D e o segundo como tipo T.



Estrutura de um programa em PALASM

Pretende-se com o presente texto ilustrar os procedimentos que permitem obter circuitos integrados de acordo com especificações concretas dos utilizadores, recorrendo a dispositivos de lógica programável de pequena complexidade, nomeadamente a PALs. Concretamente, pretendem-se utilizar circuitos integrados de lógica programável de pequena complexidade, de forma que se concretize determinada função. Nesse sentido será utilizado o PALASM, de que seguidamente se apresentam características gerais. O utilitário PALASM permite, a partir de uma especificação textual baseada em equações booleanas ou máquinas de estado, obter o ficheiro JEDEC associado, a partir do qual será possível programar o dispositivo de lógica programável.

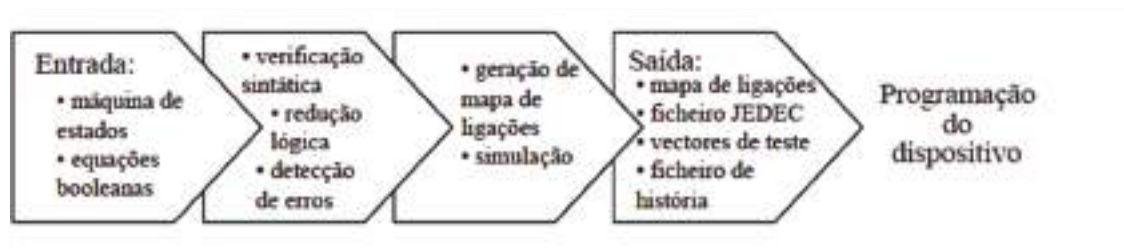


Fig. 30 - Sequência da programação em PALASM

No ficheiro de entrada poderão ser incluídas especificações de simulação que permitirão realizar a verificação do correto funcionamento do dispositivo sem que para isso seja necessário a programação efetiva do dispositivo.

O processamento do ficheiro de entrada é iniciado com a evocação do utilitário PALASM e pode ser resumido na seguinte figura em que os vários blocos representam utilitários com tarefas específicas:



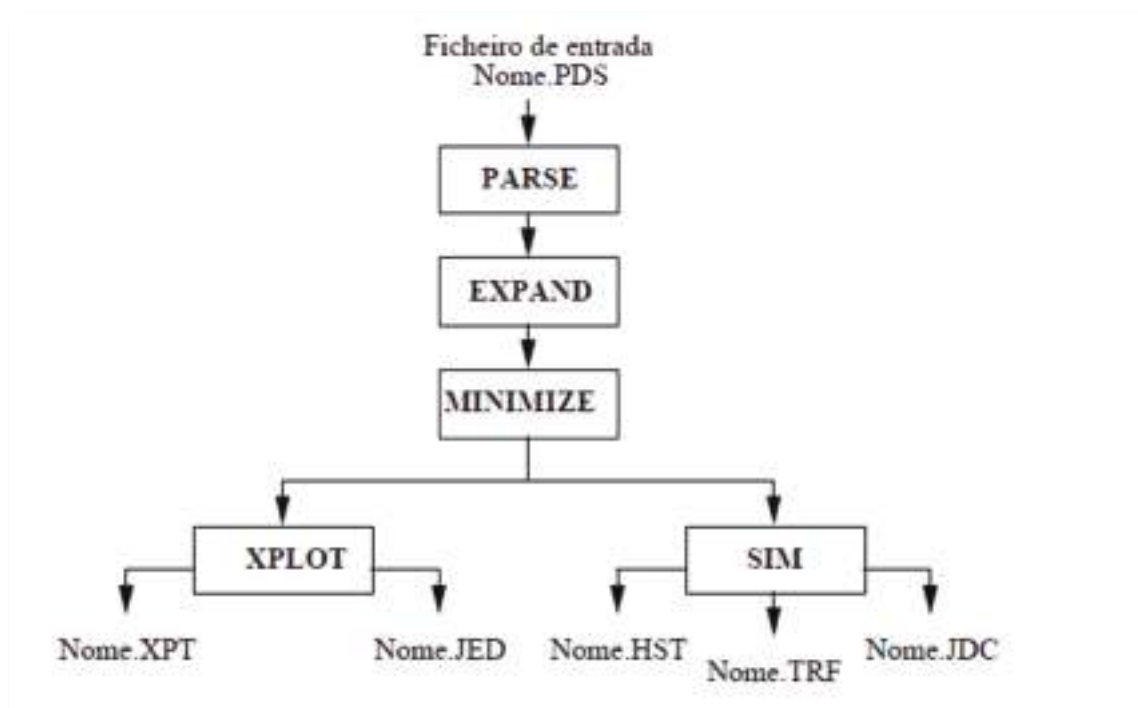


Fig. 31 - Utilitários do programa PALASM

PARSE

Verificação sintática do ficheiro de entrada, quando isento de erros gera um ficheiro intermédio;

EXPAND

Utiliza como entrada o ficheiro intermédio gerado por PARSE e realiza duas funções: expande as equações de entrada (nomeadamente os exclusivos em dispositivos que não os suportem diretamente) e converte a especificação de uma máquina de estados em equações booleanas; um novo ficheiro intermédio é gerado;

MINIMIZE

Realiza redução lógica das equações, eliminando redundâncias e minimizando expressões AND e OR;

XPLOT

Produt o mapa de ligações e ficheiro JEDEC, caso não ocorram erros;



SIM

Simula a operação do dispositivo, calculando os valores de saída com base nos sinais de entrada, nas equações booleanas e na realimentação. Produz três saídas: um ficheiro de história (que armazena todos os dados de simulação), um ficheiro de trace que armazena dados de simulação referentes apenas aos pinos especificados) e um ficheiro JEDEC com vetores de teste (para permitir que os resultados de simulação possam ser utilizados pelo programador quando da programação efetiva do dispositivo).

Ainda estão disponíveis outros utilitários, destacando-se os seguintes:

JEDMAN

Converte ficheiros JEDEC em equações booleanas;

TREPL2

Permite converter ficheiros intermédios em ficheiros de equações booleanas;

SCRSIM

Apresentador (no ecrã ou na impressora) das formas de onda geradas pela simulação.

Ficheiro de especificação

O ficheiro de entrada, através do qual se realiza a especificação baseada em equações booleanas ou na caracterização da máquina de estados, pode ser dividido nos seguintes segmentos:

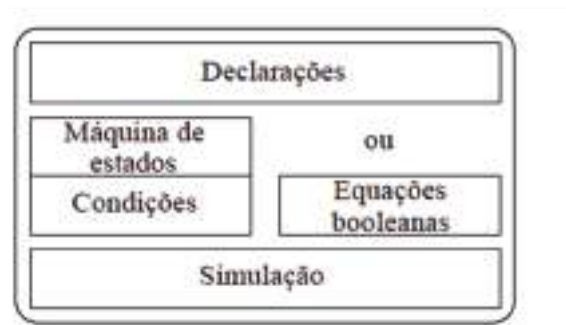


Fig. 32 - Segmento do ficheiro de entrada



A informação associada a cada um dos segmentos é a seguinte:

- Segmento de declarações - Identificação do projeto e do dispositivo a programar, identificação de nomes simbólicos associados aos pinos e definições de «macros»;
- Segmento de equações booleanas - Indicação das funções booleanas que definem as saídas, tendo em conta entradas e realimentação; indicação de equações que definem funções programáveis;
- Segmento de máquina de estados - Codificação de estados e atribuição de pinos; equação para transição de estados e saídas; Segmentos de condições - Valores das variáveis de entrada que de terminam as transições de estado;
- Segmento de simulação - Indicação dos procedimentos de teste e/ou simulação do dispositivo.

Segmento de declarações

Este segmento é composto por duas zonas: cabeçalho e definições.

O cabeçalho armazena informação geral, nomeadamente título, código de revisão, referência a autor, companhia ou instituição e data.

A zona de definições pode conter três indicações: do dispositivo a programar (numa linha iniciada pela palavra reservada CHIP), nomes associados aos pinos e definições de nomes simbólicos (a utilizar ao longo da especificação como substitutos de expressões). Apenas a linha de especificação do dispositivo é obrigatória.

Exemplo:

```

TITLE CONTADOR 3 COM RESET E CONGELADOR
PATTERN
REVISION P0.00
AUTHOR  Batman
COMPANY FCT - UNL
DATE 90-7-15
;
CHIP CONTADOR_3 PAL16V8
;PINS
;1      2      3      4      5      6      7      8      9      10
CLOCK  M      R      NC      NC      NC      NC      NC      NC      NC      GND
;11     12     13     14     15     16     17     18     19     20
OE      NC      NC      NC      /Q2   /Q1      NC      NC      NC      VCC
STRING  INPUT      ' M + R '
  
```

Fig. 33 - Segmento do ficheiro de declarações



Segmento de equações booleanas

Este segmento, que se inicia por uma linha com a palavra reservada EQUATIONS, contém a indicação das equações booleanas e funções programáveis que se pretendem programar no dispositivo.

As equações poderão corresponder (nos casos de interesse) a circuitos combinatórios ou sequenciais. Têm por base a indicação do nome do pino de saída que se pretende definir (no início da linha) seguido do símbolo '=' (nos circuitos combinatórios) ou dos símbolos ':=' (em saídas com flip -flop), seguido da expressão de definição. A expressão é composta por referências a pinos ou a macros (definidas anteriormente com STRING) separados por operadores, a saber, / para negação, * para AND, + para OR, :+: para XOR. Atenção particular deverá ser dada à polaridade dos sinais, isto é, à indicação do nome no pino de entrada/saída (ex.: A ou /A) e à sua utilização nas equações.

Das funções programáveis possíveis realce-se a facilidade de controlar o tri-state dum pino de saída, através de uma linha com a indicação do termo (constituído por um produto devido a questões de topologia das PALs) que determina essa dependência; quando o valor desse termo tomar o valor HIGH então a saída em questão apresenta baixa impedância.

```
CHIP CONTADOR_3 PAL16V8
;PINS
;1      2      3      4      5      6      7      8      9      10
CLOCK M      R      NC      NC      NC      NC      NC      NC      NC      GND
;11     12     13     14     15     16     17     18     19     20
OE      NC      NC      NC      /Q2    /Q1      NC      NC      /X      VCC

EQUATIONS
; saídas com memória
Q1 := /R * M * /Q2 * /Q1
    + /R * /M * /Q2 * Q1
Q2 := /R * M * /Q2 * Q1
    + /R * /M * Q2 * /Q1
; saída combinatória
X = /Q2 * /Q1 * M
X.TRST = VCC
```

Fig. 34 - Segmento do ficheiro de equações booleanas

Segmentos de máquina de estados e condições

Este segmento que se inicia com a palavra reservada STATE (e deverá ser seguido pelo segmento CONDITIONS) contém a especificação de uma máquina de estados. Esta



especificação poderá ser facilmente obtida de um diagrama de estados que inclua: todos os estados e respectivos nomes, condições no que respeita às variáveis de entrada que impõem transições de estados (sincronizados pelo sinal de relógio) e indicações dos valores das variáveis de saída.

Podemos decompor este segmento em três partes:

- Sobre valores a tomar por omissão (por exemplo, se é uma máquina de Moore ou de Mealy ou definir implicitamente estados seguintes quando isso não possa ser determinado pela especificação de transições),
- Codificação de estados (isto é, atribuição de valores lógicos a um conjunto de pinos para representar um estado),
- Transições entre estados e saídas associadas.

Dos valores a tomar por omissão refiram-se as seguintes opções:

- MEALY_MACHINE ou MOORE_MACHINE - tipo de máquina pretendida;
- DEFAULT_BRANCH nome_de_estado (define próximo estado quando isso não possa ser determinado com base na especificação) ou DEFAULT_BRANCH HOLD_STATE (mantém a máquina no estado atual quando o próximo estado não puder ser determinado com base na especificação) ou DEFAULT_BRANCH NEXT_STATE (determina o próximo estado da máquina com base em NEXT_STATE quando o próximo estado não puder ser determinado com base na especificação) A codificação de estados é realizada através duma equação que define uma combinação de valores de saída e tem um formato do tipo:

Nome_de_estado = Pino_de_saída_1 * ... * Pino_de_saída_n

As transições entre estados são representadas por equações dependentes de condições (a definir posteriormente) e próximos estados e têm um formato do tipo:

Nome_de_estado := Condição_1 -> Estado_Seguinte

...

+ Condição_n -> Estado_Seguinte

+ -> Nome_de_estado_local_por_omissão

Uma transição incondicional entre dois estados deverá ser referida como

Nome_de_estado := VCC -> Estado_Seguinte



A atribuição de saídas a um determinado estado (que só tem sentido se os bits associados forem diferentes dos das variáveis de estado) pode ser especificada do seguinte modo:

- Saídas que recorram a elemento de memória em máquinas de Mealy (válidas no ciclo de relógio seguinte ao novo estado ser atingido)

Nome_de_estado.OUTPUT := Condição_1 -> Saídas

...

+ Condição_n -> Saídas

+> Saídas_locais_por_omissão

- Saídas «combinatórias» em máquinas de Mealy (válidas no ciclo de relógio em que o novo estado é atingido)

Nome_de_estado.OUTPUT = Condição_1 -> Saídas

...

+ Condição_n -> Saídas

+> Saídas_locais_por_omissão

- Saídas que recorram a elemento de memória em máquinas de Moore (válidas no ciclo de relógio em que o novo estado é atingido)

Nome_de_estado.OUTPUT := Saídas

- Saídas «combinatórias» em máquinas de Moore (válidas no ciclo de relógio em que o novo estado é atingido)

Nome_de_estado.OUTPUT = Saídas

O segmento de condições é iniciado por uma linha com a palavra reservada **CONDITIONS** e especifica equações que determinam as transições entre estados.

Uma equação de especificação de condição tem um formato do tipo

Nome_de_condição = Entrada_1 * ... * Entrada_x

...

+ Entrada_y * ... * Entrada_n



Exemplo:

```

STATE
; Omissões
MOORE MACHINE
DEFAULT BRANCH HOLD STATE
; Codificação de estados
S0 = /Q1 * /Q0      ; 0-0
S1 = Q1 * /Q0       ; 1-0
S2 = Q1 * Q0        ; 1-1
S3 = /Q1 * Q0       ; 0-1
; Transições e saídas
S0 := C0 -> S1
    + C1 -> S1
    + C2 -> S2
S1 := VCC -> S2
S2 := VCC -> S3
S3 := C0 -> S1
    + C1 -> S1
    + C2 -> S2
CONDITIONS
C0 = /SEN1 * /SEN2
C1 = /SEN1 * SEN2
C2 = SEN1 * SEN2
C3 = SEN1 * /SEN2

```

Fig. 35 - Exemplo de um segmento de estado e condições

Segmento de simulação

O segmento de simulação é indicado em último lugar e está baseado numa linguagem própria que dispõe de estruturas que permitem ciclos de repetição, saltos condicionais, ativação de sinais, verificação de estados de sinais para além da possibilidade de selecionar os sinais observáveis.

Um resumo dos comandos de simulação é apresentado seguidamente:

SETF Especifica novos valores para as entradas (ATENção especial às polaridades dos sinais);

CLOCKF Gera um ciclo de sinal de relógio no pino dedicado;

CHECK Verifica que os valores esperados e obtidos são coincidentes;

TRACE_ON Define os sinais para os quais se pretende visualizar a simulação;

TRACE_OFF Anula o comando TRACE_ON;

FOR ... TO ... DO loop Repete um conjunto de comandos por um número fixo de vezes;

WHILE ... DO loop Repete um conjunto de comandos até que uma condição seja atingida;

IF ... THEN ... ELSE Salto condicional.



Exemplo:

```

SETF A /OE B
CLOCKF CLK
CHECK Q0 /Q1
TRACE ON A /OE B Q0 Q1
TRACE_OFF

FOR J := 3 TO 8 DO
BEGIN
  SETF A /B
  CLOCKF CLK
END

WHILE (I<2) DO
BEGIN
  CLOCKF CLK
END

IF J = 5 THEN
BEGIN
  CHECK Q0
END
ELSE
BEGIN
  CHECK /Q0
END

```

Fig. 36 - Exemplo de um segmento de simulações

Exemplo de aplicação

Implementação de um contador decrescente de módulo 3 baseado na PAL16V8, dispondo de duas entradas síncronas, uma (RESET) para reset síncrono e outra (MORE) para mudança de estado.

Descrição do funcionamento

Saída Seg.	= (Saída - 1) módulo 3	se RESET = 0 e MORE = 1
	= Saída	se RESET = 0 e MORE = 0
	= Saída inicial	se RESET = 1



Diagrama de estados

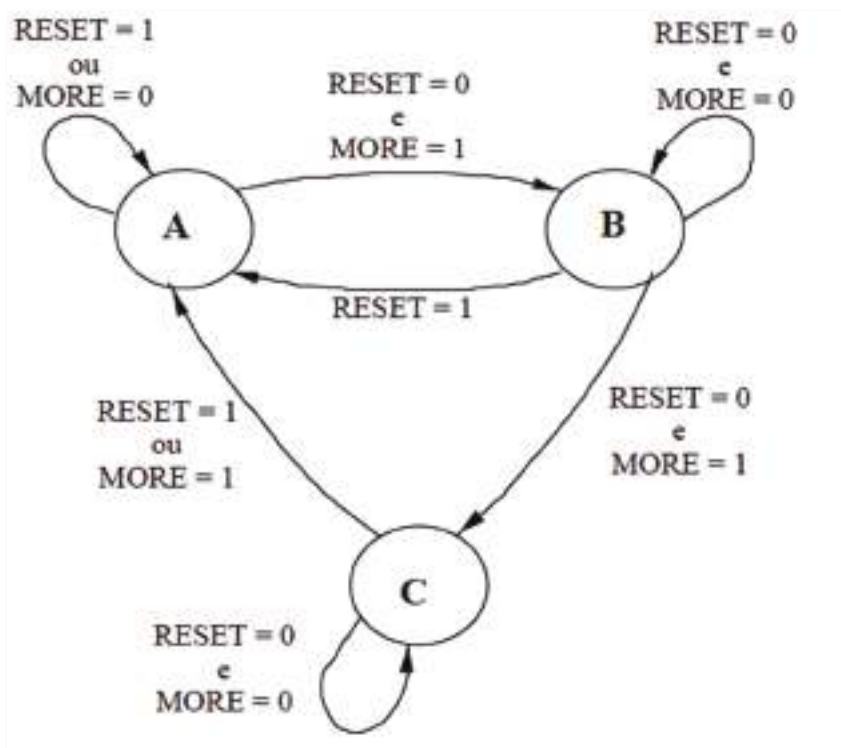


Tabela de estados

Est. anterior	Estado seguinte				RESET
	0	0	1	1	
	0	1	1	0	
A	A	B	A	A	
B	B	C	A	A	
C	C	A	A	A	

Resolução utilizando método convencional

Codificação de estados:

Estado	Q ₂ Q ₁	Saídas
A	00	11
B	01	10
C	10	01

<- estado e saídas iniciais



Tabela de estados:

Est. anterior (Q ₂ Q ₁)	Est. seguinte (Q ₂ Q ₁)				
	0	0	1	1	
					RESET
	0	1	1	0	MORE
A 00	00	01	00	00	
B 01	01	10	00	00	
C 10	10	00	00	00	

Mapas associados:

Est. anterior (Q ₂ Q ₁)	Q ₂ seg. = D ₂				
	0	0	1	1	
					RESET
	0	1	1	0	MORE
A 00	0	0	0	0	
B 01	0	1	0	0	
Ø 11	x	x	x	x	
C 10	1	0	0	0	

Est. anterior (Q ₂ Q ₁)	Q ₁ seg. = D ₁				
	0	0	1	1	
					RESET
	0	1	1	0	MORE
A 00	0	1	0	0	
B 01	1	0	0	0	
Ø 11	x	x	x	x	
C 10	0	0	0	0	

Expressões:

$$D_1 = Q_1 * /RESET * /MORE + /Q_1 * /Q_2 * /RESET * MORE$$

$$D_2 = Q_2 * /RESET * /MORE + Q_1 * /RESET * MORE$$

Resolução utilizando método ad-hoc:

Est. anterior Est. seguinte

A A se (RESET=1 ou MORE=0) ou

B se (RESET=0 e MORE=1)



B B se (RESET=0 e MORE=0) ou

C se (RESET=0 e MORE=1) ou

A se (RESET=1)

C C se (RESET=0 e MORE=0) ou

A se (RESET=1 ou MORE=1)

Isto é:

$A \rightarrow A * (\text{RESET} + \text{MORE}) + B * (\text{RESET} * \text{MORE})$

$B \rightarrow B * (\text{RESET} * \text{MORE}) + C * (\text{RESET} * \text{MORE}) + A * (\text{RESET})$

$C \rightarrow C * (\text{RESET} * \text{MORE}) + A * (\text{RESET} + \text{MORE})$

Codificação de estados:

Estado Q₂ Q₁ Saídas

A 00 11 <- estado e saídas iniciais

B 01 10

C 10 01

Tabela resultante:

Estado anterior	Estado seguinte
(Q ₂ Q ₁)	(Q ₂ Q ₁)
A (0 0)	(00).(RESET + MORE) + (01).(RESET.MORE)
B (0 1)	(01).(RESET./MORE) + (10).(RESET.MORE) + (00).RESET
C (1 0)	(10).(RESET./MORE) + (00).(RESET+MORE)

Tabela referente à variável Q₂:

Estado anterior	Estado seguinte
Q ₂ Q ₁	(Q ₂)
A 0 0	0.(RESET + MORE) + 0.(RESET.MORE)
B 0 1	0.(RESET./MORE) + 1.(RESET.MORE) + 0.RESET
C 1 0	1.(RESET./MORE) + 0.(RESET+MORE)

Tabela referente à variável Q₁:

Estado anterior	Estado seguinte
-----------------	-----------------



	Q2 Q1	(Q1)
A	0 0	$0.(\text{RESET} + \text{MORE}) + 1.(/ \text{RESET}.\text{MORE})$
B	0 1	$1.(/ \text{RESET}./ \text{MORE}) + 0.(/ \text{RESET}.\text{MORE}) + 0.\text{RESET}$
C	1 0	$0.(/ \text{RESET}./ \text{MORE}) + 0.(\text{RESET} + \text{MORE})$

Tabela final:

	Estado anterior	Estado seguinte	
	Q2 Q1	(Q2)	(Q1)
A	0 0	0	$/ \text{RESET}.\text{MORE}$
B	0 1	$/ \text{RESET}.\text{MORE}$	$/ \text{RESET}./ \text{MORE}$
C	1 0	$/ \text{RESET}.$	$/ \text{MORE} 0$

Expressões:

$$D1 = / \text{RESET} * \text{MORE} * / \text{Q2} * / \text{Q1} + / \text{RESET} * / \text{MORE} * / \text{Q2} * \text{Q1}$$

$$D2 = / \text{RESET} * \text{MORE} * / \text{Q2} * \text{Q1} + / \text{RESET} * / \text{MORE} * \text{Q2} * \text{Q1}$$

Ficheiro fonte para PALASM

Alternativa 1, baseada na indicação direta das equações:

TITLE CONTADOR 3 COM RESET E CONGELADOR

PATTERN

REVISION P0.00

AUTHOR Alguém

COMPANY FCT - UNL

DATE 90-7-15

;

CHIP CONTADOR_3 PAL16V8



```
;PINS
;1      2      3      4      5      6      7      8      9      10
  CLOCK M      R      NC      NC      NC      NC      NC      NC      NC      GND

;11     12     13     14     15     16     17     18     19     20
  OE      NC      NC      NC      /Q2    /Q1      NC      NC      NC      VCC
```

EQUATIONS

```
Q1 := /R * M * /Q2 * /Q1 + /R * /M * /Q2 * Q1
Q2 := /R * M * /Q2 * Q1 + /R * /M * Q2 * /Q1
```

SIMULATION

; a especificação da simulação é apresentada mais à frente

Alternativa 2, baseada na indicação da máquina de estados:

```
TITLE CONTADOR 3 COM RESET E CONGELADOR
PATTERN
REVISION P1.00
AUTHOR Alguém
COMPANY FCT - UNL
DATE 90-7-15
;
CHIP CONTADOR_3 PAL16V8

;PINS
;1      2      3      4      5      6      7      8      9      10
  CLOCK M      R      NC      NC      NC      NC      NC      NC      NC      GND

;11     12     13     14     15     16     17     18     19     20
  OE      NC      NC      NC      /Q2    /Q1      NC      NC      NC      VCC

STATE

; OMISSÕES
MOORE_MACHINE

;CODIFICACAO DE ESTADOS

STATE1 = /Q2 * /Q1
STATE2 = /Q2 * Q1
STATE3 = Q2 * /Q1
```



```

;TRANSIÇÕES E SAÍDAS
;as saídas são obtidas directamente das variáveis de estado

STATE1 := INC    -> STATE2
        + RST    -> STATE1
        + IGU    -> STATE1

STATE2 := INC    -> STATE3
        + RST    -> STATE1
        + IGU    -> STATE2

STATE3 := INC    -> STATE1
        + RST    -> STATE1
        + IGU    -> STATE3

CONDITIONS
INC = /R * M
IGU = /R * /M
RST = R

```

SIMULATION

; a especificação da simulação é apresentada mais à frente

Alternativa 3, baseada na indicação da máquina de estados:

```

TITLE CONTADOR 3 COM RESET E CONGELADOR
PATTERN
REVISION P1.01
AUTHOR Alguém
COMPANY FCT - UNL
DATE 90-7-15
;
CHIP CONTADOR_3 PAL16V8

;PINS
;1      2      3      4      5      6      7      8      9      10
CLOCK  M      R      NC      NC      NC      NC      NC      NC      NC      GND

;11     12     13     14     15     16     17     18     19     20
OE      NC      NC      NC      /Q2   /Q1      NC      NC      NC      VCC

STATE

; OMISSÕES
MOORE_MACHINE
DEFAULT_BRANCH STATE1 ; que permitira' considerar RESET
                      ; implicitamente

;CODIFICACAO DE ESTADOS

STATE1 = /Q2 * /Q1
STATE2 = /Q2 * Q1
STATE3 = Q2 * /Q1

;TRANSIÇÕES E SAÍDAS
;as saídas são obtidas directamente das variáveis de estado

```



```

STATE1 := INC    -> STATE2
        + IGU    -> STATE1

STATE2 := INC    -> STATE3
        + IGU    -> STATE2

STATE3 := INC    -> STATE1
        + IGU    -> STATE3

```

```

CONDITIONS
INC = /R * M
IGU = /R * /M

```

SIMULATION

; a especificação da simulação é apresentada mais à frente

Especificação da simulação

```

SIMULATION
TRACE_ON R M /Q2 /Q1

SETF R          ; RESET ON
M              ; INCREMENTA
/OE            ; ACTIVA SAIDAS
/CLOCK

CLOCKF CLOCK    ; PARA ACTIVAR RESET
CHECK /Q2 /Q1   ; VERIFICAÇÃO DE STATE1
SETF /R

CLOCKF CLOCK
CHECK /Q2 Q1    ; VERIFICAÇÃO DE INCREMENTO PARA STATE2
SETF /M
CLOCKF CLOCK
CHECK /Q2 Q1    ; VERIFICAÇÃO DE CONGELAMENTO EM STATE2
SETF M

CLOCKF CLOCK
CHECK Q2 /Q1    ; VERIFICAÇÃO DE INCREMENTO PARA STATE3
SETF /M
CLOCKF CLOCK
CHECK Q2 /Q1    ; VERIFICAÇÃO DE CONGELAMENTO EM STATE3
SETF M

CLOCKF CLOCK
CHECK /Q2 /Q1   ; VERIFICAÇÃO DE INCREMENTO PARA STATE1
SETF /M
CLOCKF CLOCK
CHECK /Q2 /Q1   ; VERIFICAÇÃO DE CONGELAMENTO EM STATE1
SETF M

SETF R
CLOCKF CLOCK
CHECK /Q2 /Q1   ; VERIFICAÇÃO DE RESET A PARTIR DO STATE1

```



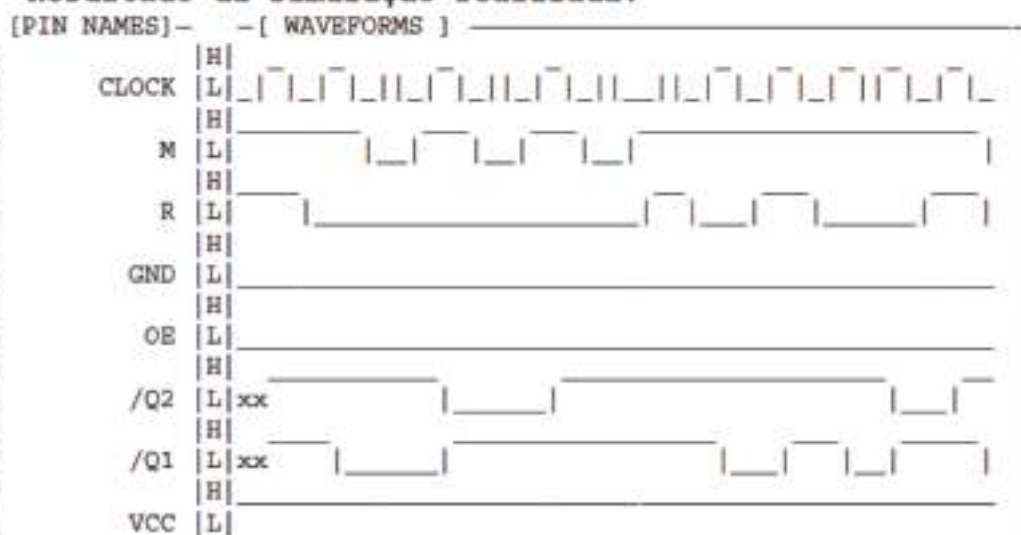
```

SETF /R
CLOCKF CLOCK      ; MUDANÇA PARA STATE2
CHECK /Q2 Q1
SETF R
CLOCKF CLOCK
CHECK /Q2 /Q1      ; VERIFICAÇÃO DE RESET A PARTIR DO STATE2

SETF /R
CLOCKF CLOCK      ; MUDANÇA PARA STATE2
CHECK /Q2 Q1
CLOCKF CLOCK      ; MUDANÇA PARA STATE3
CHECK Q2 /Q1
SETF R
CLOCKF CLOCK
CHECK /Q2 /Q1      ; VERIFICAÇÃO DE RESET A PARTIR DO STATE3
TRACE_OFF

```

Resultado da simulação realizada:



Commands: <ESC>ape <ARROW KEYS> <HOME> <END> <PGUP> <PGDN>



Mapa de ligações resultante (matriz de pontos programáveis)

```

      11 1111 1111 2222 2222 2233
0123 4567 8901 2345 6789 0123 4567 8901

0 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
a
23 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

24 X--- -X-- ---- --X- --X- ---- ---- /R*M*/Q2*/Q1
25 -X-- -X-- ---- --X- --X- ---- ---- /R*/M*/Q2*Q1
26 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
27 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
28 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
29 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
30 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
31 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

32 X--- -X-- ---- --X- --X- ---- ---- /R*M*/Q2*Q1
33 -X-- -X-- ---- --X- --X- ---- ---- /R*/M*Q2*/Q1
34 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
35 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
36 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
37 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
38 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

39 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

40 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
a
63 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

LEGEND: X : FUSE NOT BLOWN (L,N,0) - : FUSE BLOWN (H,P,1)

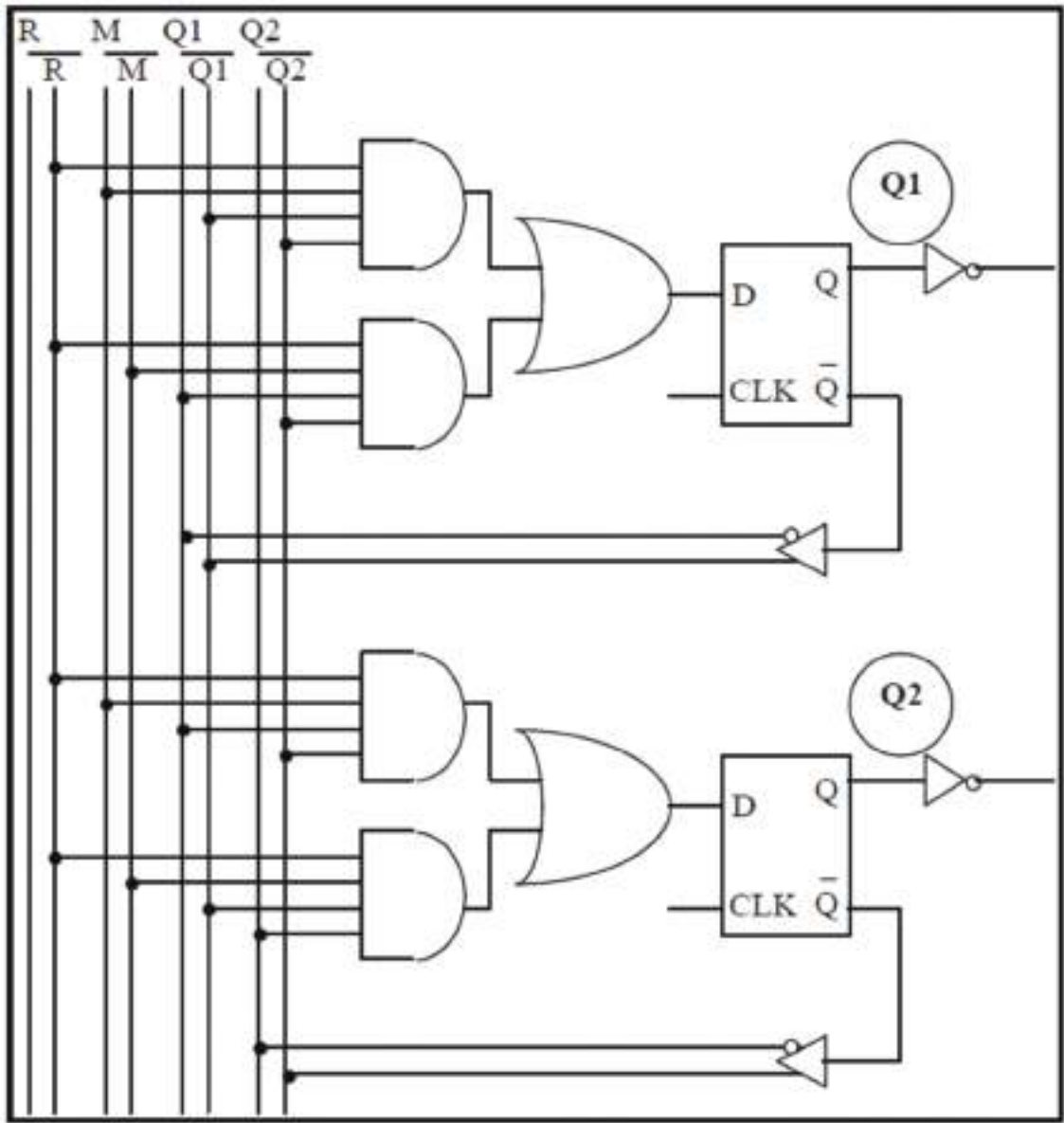
NUMBER OF FUSES BLOWN - 112

SECURITY FUSE XX

```



Circuito resultante



Bibliografia

PADILHA, António e outros, *Electrónica Digital*. McGrawHill. (s.d.).

PADILHA, António, *Sistemas Digitais*. McGrawHill. (s.d.).

PEREIRA, A. Silva; ÁGUA, Mário; BALDAIA, Rogério, *Sistemas Digitais, 12.º Ano. Curso Tecnológico de Electrotecnia e Electrónica*. Porto Editora. (s.d.).

