Dispositivos Lógicos Programáveis

Maio de 2006

Dispositivos Lógicos Programáveis

Mário P. Véstias

Instituto Superior de Engenharia de Lisboa - ISEL

ÍNDICE

5 Dispositivo PAL

A PAL® é um dispositivo programável cuja marca foi registada 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 actualmente 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, enquanto que 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 em termos de programação, pois a matriz de saída é fixa.

Para ilustrar a arquitectura típica de uma PAL®, consideremos uma configuração exemplo com apenas quatro entradas e quatro saídas (ver figura 14).

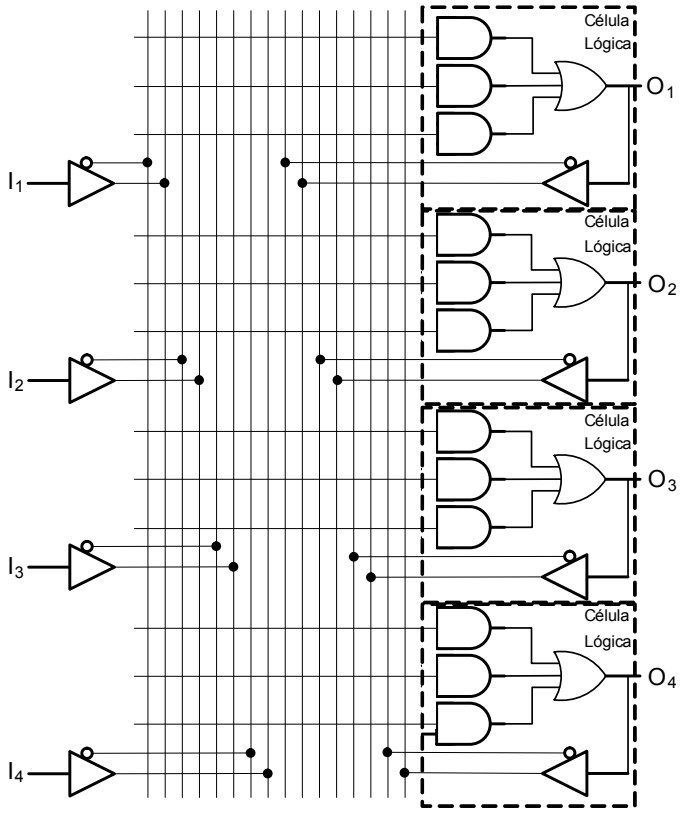


Figura 14 – Exemplo de arquitectura de uma pequena PAL®

O dispositivo da figura 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®.

5.1 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 directamente 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, F_0, F_1, F_2, F_3:

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

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

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

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

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 F_0 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 F_0. 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 F_1 ($A\overline{D} + \overline{B}\overline{C}$) faz parte da expressão lógica de F_0 ($\mathbf{A\overline{D}} + \mathbf{\overline{B}\overline{C}} + \overline{A}BD + \overline{B}\overline{D}$). Assim, a função F_0 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 F_3, F_2 e F_0, apenas teríamos de criar uma nova função que realizaria parte de F_0. Por exemplo, $F_5 = \overline{B}\overline{C} + \overline{A}BD + \overline{B}\overline{D}$ ou $F_5 = A\overline{D} + \overline{B}\overline{C}$.

Depois de simplificar as funções, basta programar a matriz da PAL® (ver figura 15).

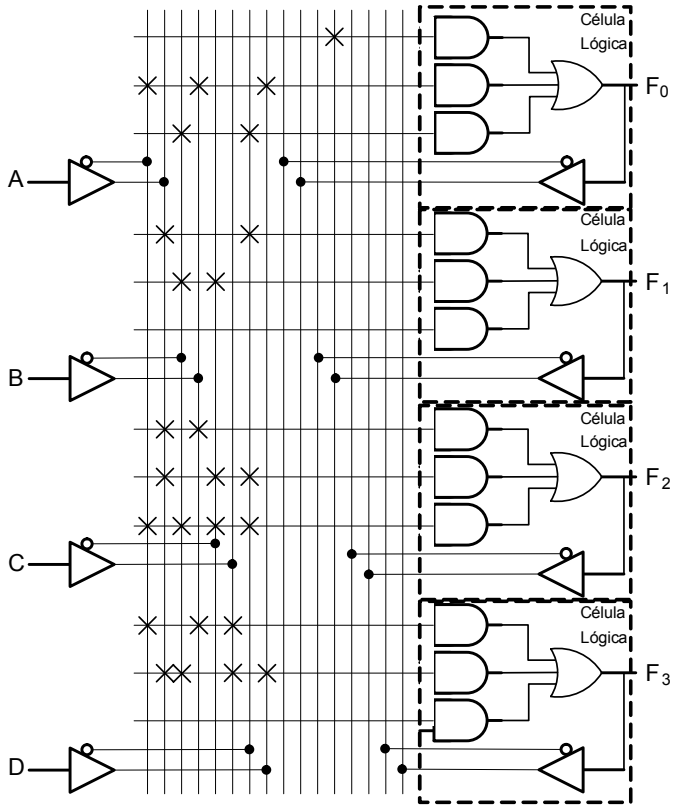


Figura 15 – Implementação das quatro funções na PAL®

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

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.

5.2 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.

5.2.1 Arquitectura 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 16). 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 activa, basta não ligar nenhum sinal à porta AND de controlo;
- um *buffer* de realimentação da saída respectiva 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 bidireccionais, podendo ser usados como entrada, como saída ou como entrada/saída.

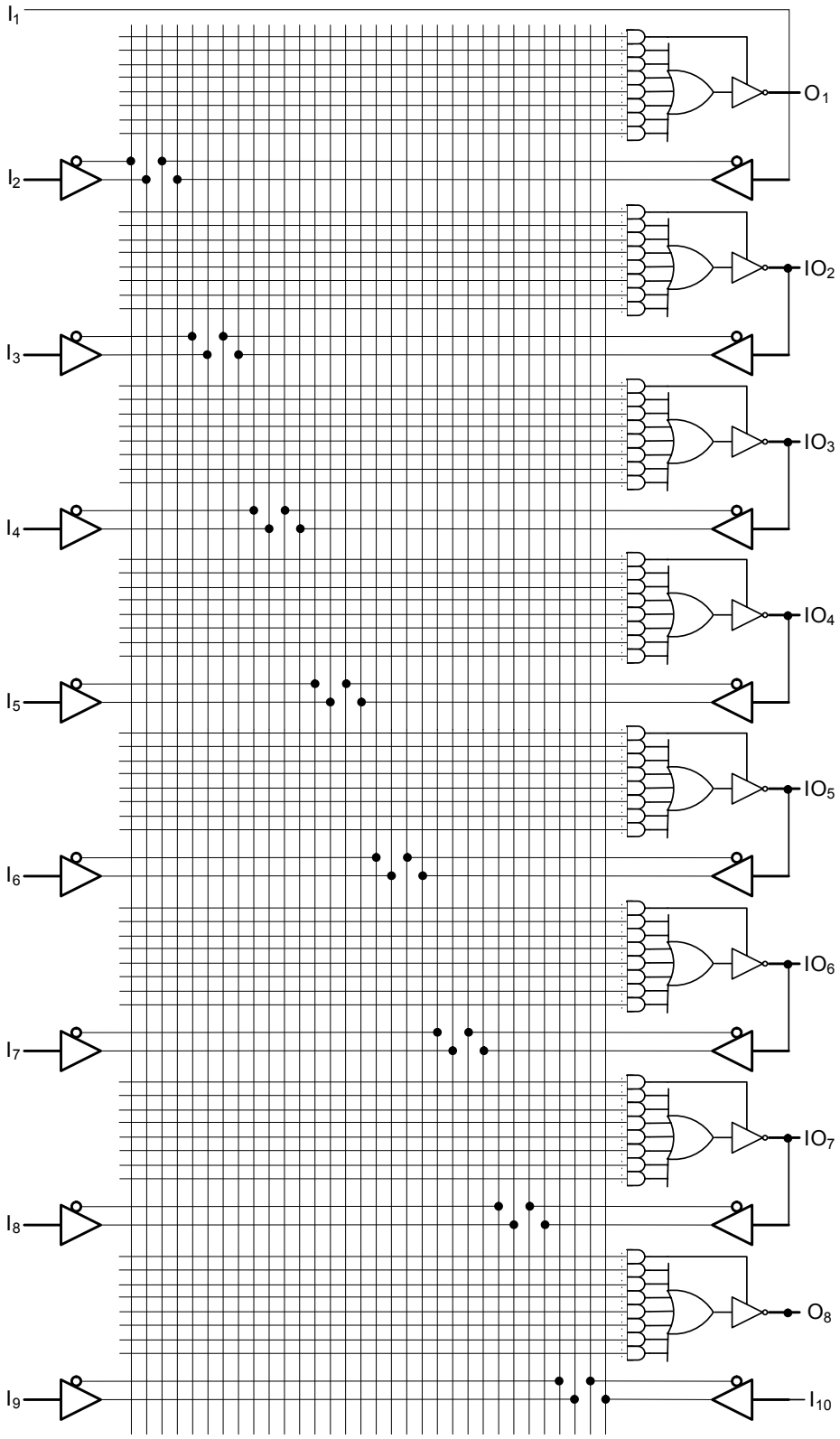


Figura 16 – Arquitectura da PAL® 16L8

Os pinos bidireccinais 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 inactiva;
- pode ser usado como saída. Para tal, basta activar 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, activa-se a porta *three-state*. No caso de se querer receber dados pelo pino, desactiva-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.

NOTA: Uma questão que se coloca nesta altura é a de saber se uma PAL com *m* termos de produto suporta ou não a implementação de todas as funções de *n* entradas (na PAL16L8, temos 16 entradas e 7 termos de produto). De facto, como era de esperar, o número de termos de produto disponibilizados por uma PAL® está longe de suportar todas as funções com o mesmo número de entradas da PAL®. Sabemos que a função que necessita de mais termos de produto é o XOR, sendo que um XOR de *n* entradas requer 2^{n-1} termos de produto (e.g., para 8 entradas, seriam necessários 128 termos de produto!). O que se passa é que, na maioria dos casos, não somos confrontados com funções tão extensas. Na maioria dos projectos de pequena e média complexidade, em que a PAL® é tida como implementação preferencial, na pior das hipóteses surge a necessidade de recorrer à realimentação das funções de saída para implementar uma estrutura AND-OR-AND-OR ... A desvantagem desta realimentação reside na duplicação do atraso de propagação da função lógica.

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):

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

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

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

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

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 17).

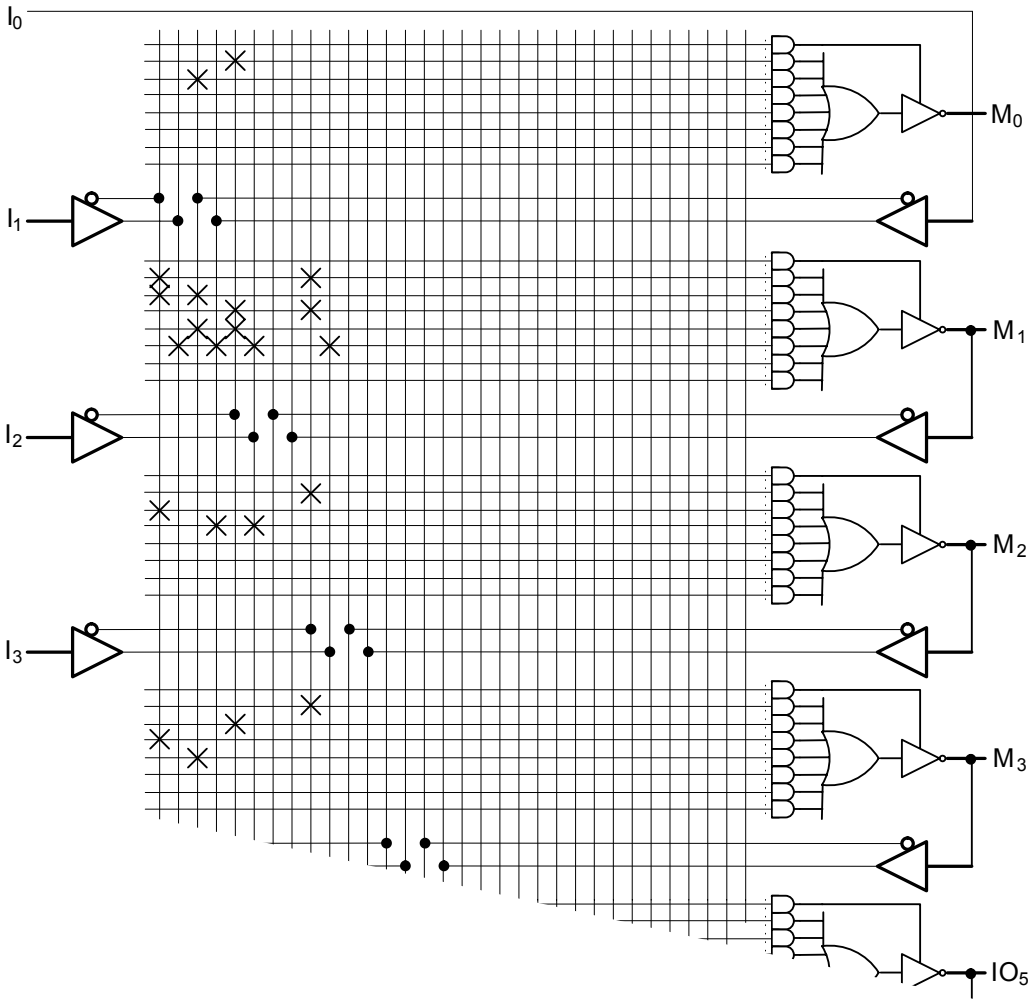


Figura 17 – Implementação do multiplicador de 2-bits na PAL® 16L8

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

5.2.2 Arquitectura de PAL® sequencial PAL16R8

A PAL® 16L8 apenas implementa funções combinatórias porque não tem elementos de memória, como *latches* ou *flip-flops*. Com o objectivo de implementar circuitos sequencias em PAL®, surgiu a primeira geração de PAL® sequenciais com a designação PAL16R8. Este dispositivo programável tem 8 entradas e 8 oito saídas, uma entrada de relógio comum a todos os flip-flops tipo D e uma entrada de controlo comum a todos as portas *three-state* de saída (ver figura 18).

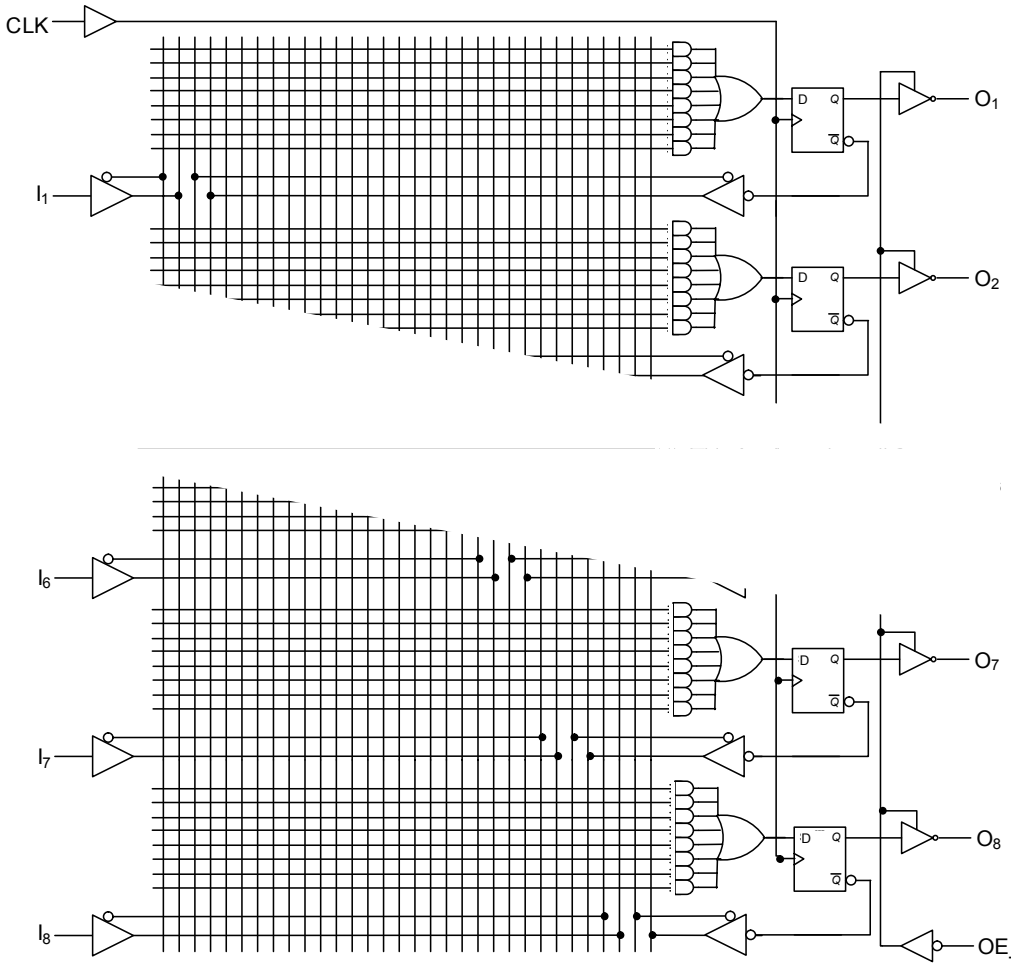


Figura 18 – Arquitectura da PAL® 16R8

As saídas complementares dos flip-flop tipo D são realimentadas para a malha, facilitando a implementação de contadores, de registos 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 sequencias, 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 projecto. 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.

5.2.3 Arquitectura de PAL® sequencial ATF22V10

A ATF22V10 tem as seguintes características (ver figura 19):

- 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 activas a HIGH ou activas a LOW;
- número de termos de produto por macrocélula variável de 8 a 16 termos. Pela figura 19 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 IO_1 tem 8 termos de produto, enquanto que a associada à saída IO_4 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 inactivo. Caso o pino seja configurado como saída, o *three-state* está sempre activo. 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 bidireccional.

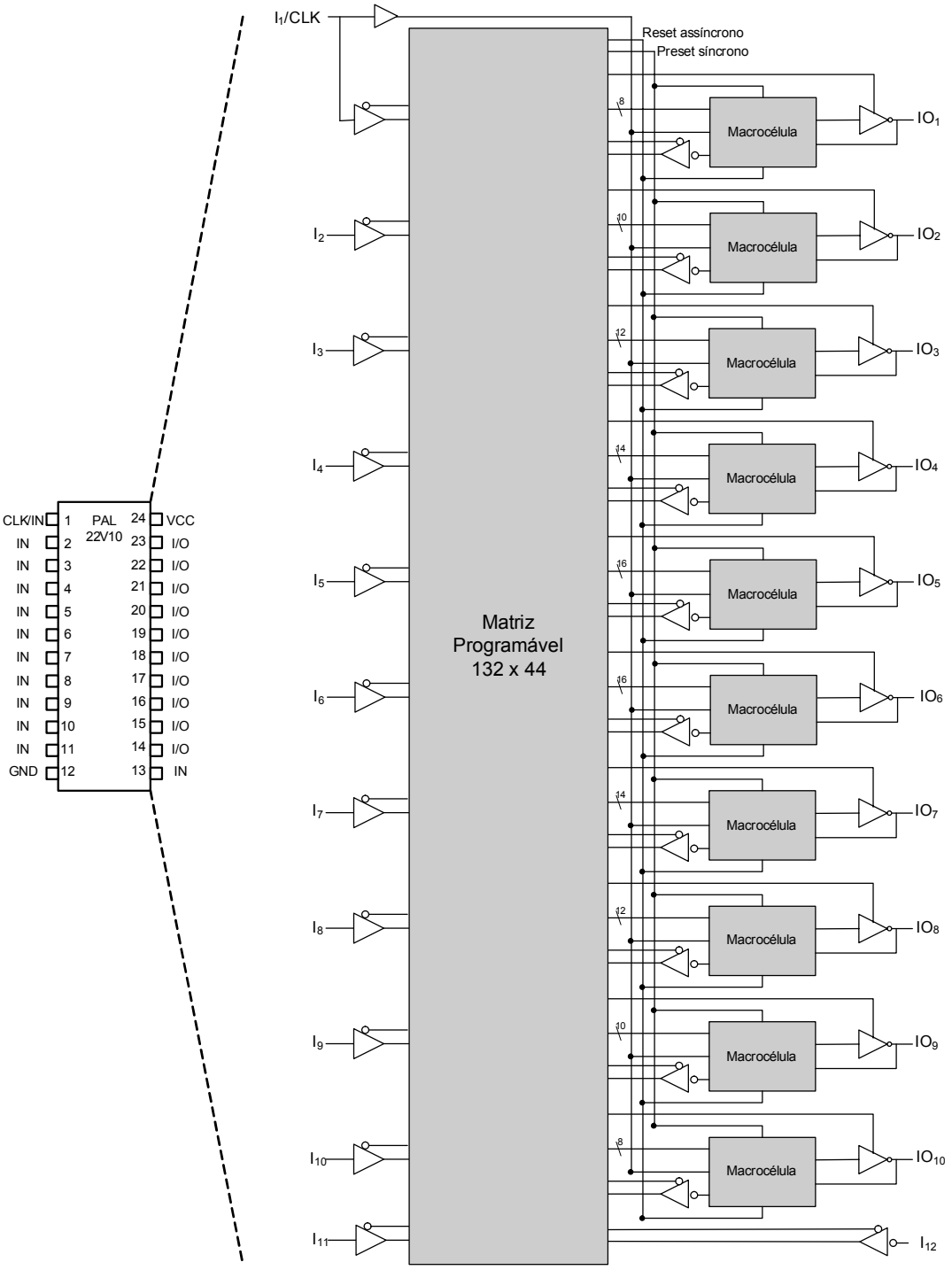
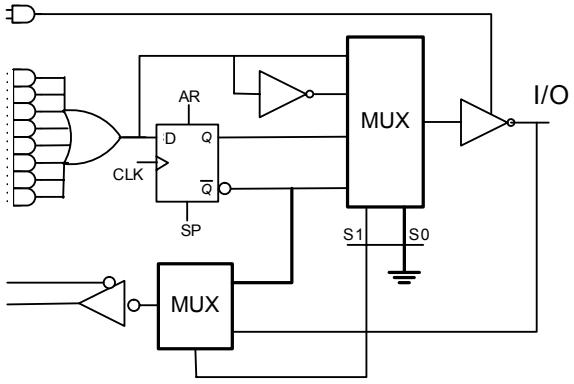


Figura 19 – Arquitectura da PAL® ATF22V10

A macrocélula de saída pode ser configurada de acordo com uma de quatro configurações diferentes: saída combinatória activa a HIGH, saída combinatória activa a LOW, saída sequencial activa a HIGH ou saída sequencial activa a LOW. A escolha da configuração é feita de acordo com a aplicação do utilizador e é conseguida através de dois bit de configuração: S0 e S1 (ver figura 20).



S1	S0	Configuração
0	0	Sequencial/activa a LOW
0	1	Sequencial/activa a HIGH
1	0	Combinatória/activa LOW
1	1	Combinatória/activa HIGH

Figura 20 – Arquitectura da macrocélula da PAL® 22V10

A configuração sequencial activa 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 (ver figura 21). Na mesma figura, está ilustrado o circuito lógico equivalente após a configuração da macrocélula

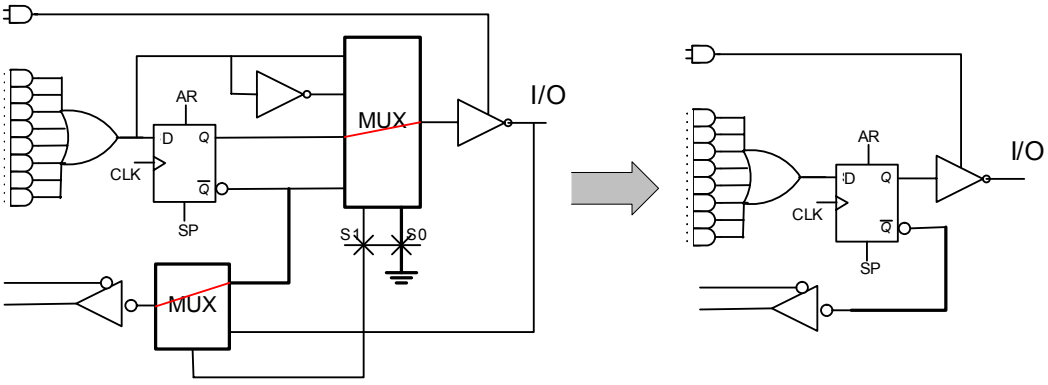


Figura 21 – Configuração sequencial activa a LOW da macrocélula da ATF22V10

Para obter uma saída registada activa a HIGH, o multiplexer encaminha a saída complementada do flip-flop que depois é invertida pelo *three-state* (ver figura 22).

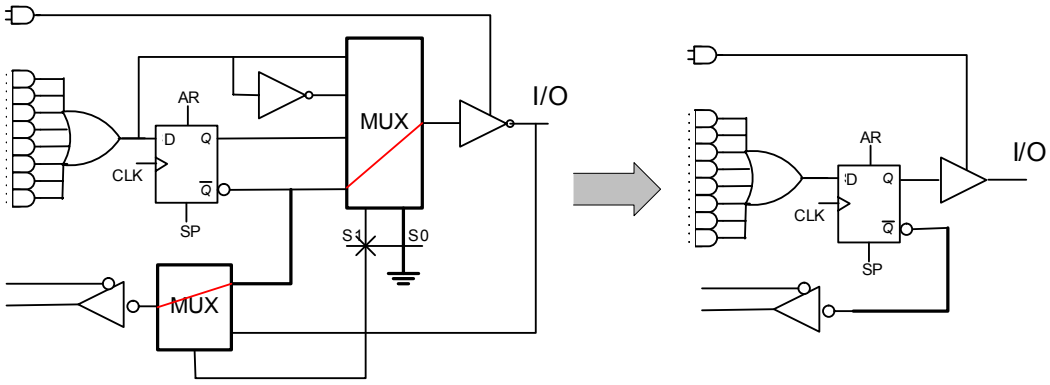


Figura 22 – Configuração sequencial activa a HIGH da macrocélula da ATF22V10

Para ter saídas combinatórias, basta seleccionar as entradas do multiplexer que não passam pelo flip-flop. Para a saída activa a LOW, selecciona a entrada negada e o multiplexer de entrada selecciona o sinal vindo da saída do *three-state* (ver figura 23).

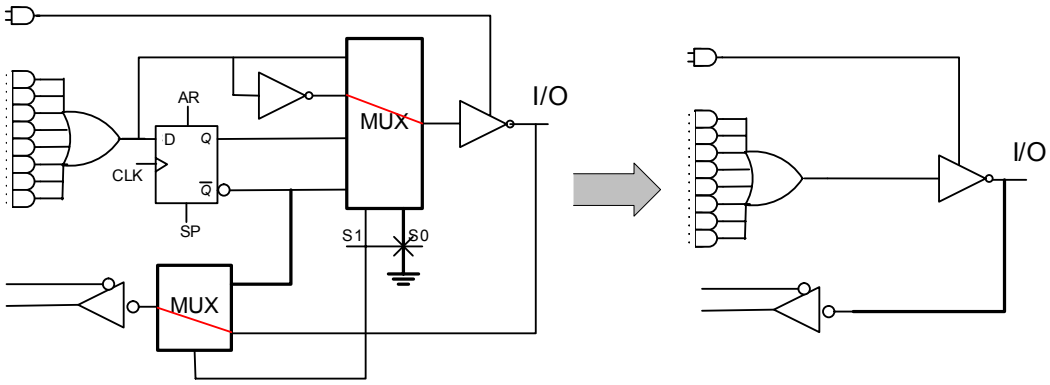


Figura 23 – Configuração combinatória activa a LOW da macrocélula da ATF22V10

Para a saída activa a HIGH, selecciona a entrada não negada e o multiplexer de entrada selecciona o sinal vindo da saída do *three-state* (ver figura 24).

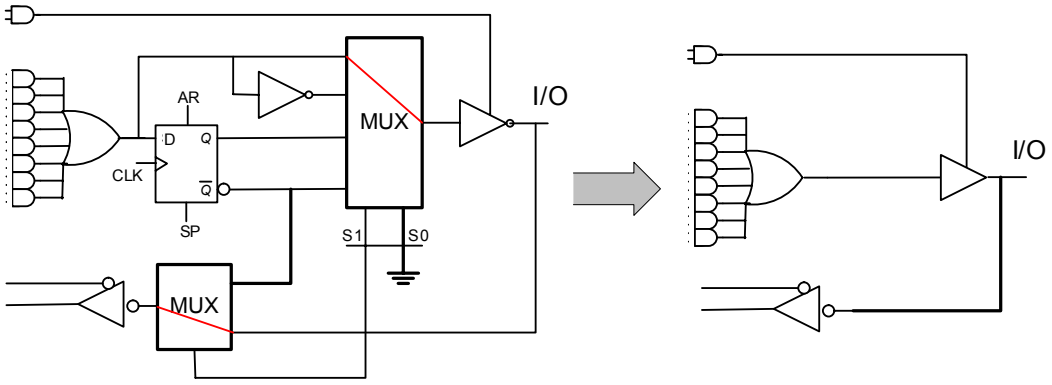


Figura 24 – Configuração combinatória activa a HIGH da macrocélula da ATF22V10

A figura 25 ilustra a arquitectura completa da PAL® ATF22V10.

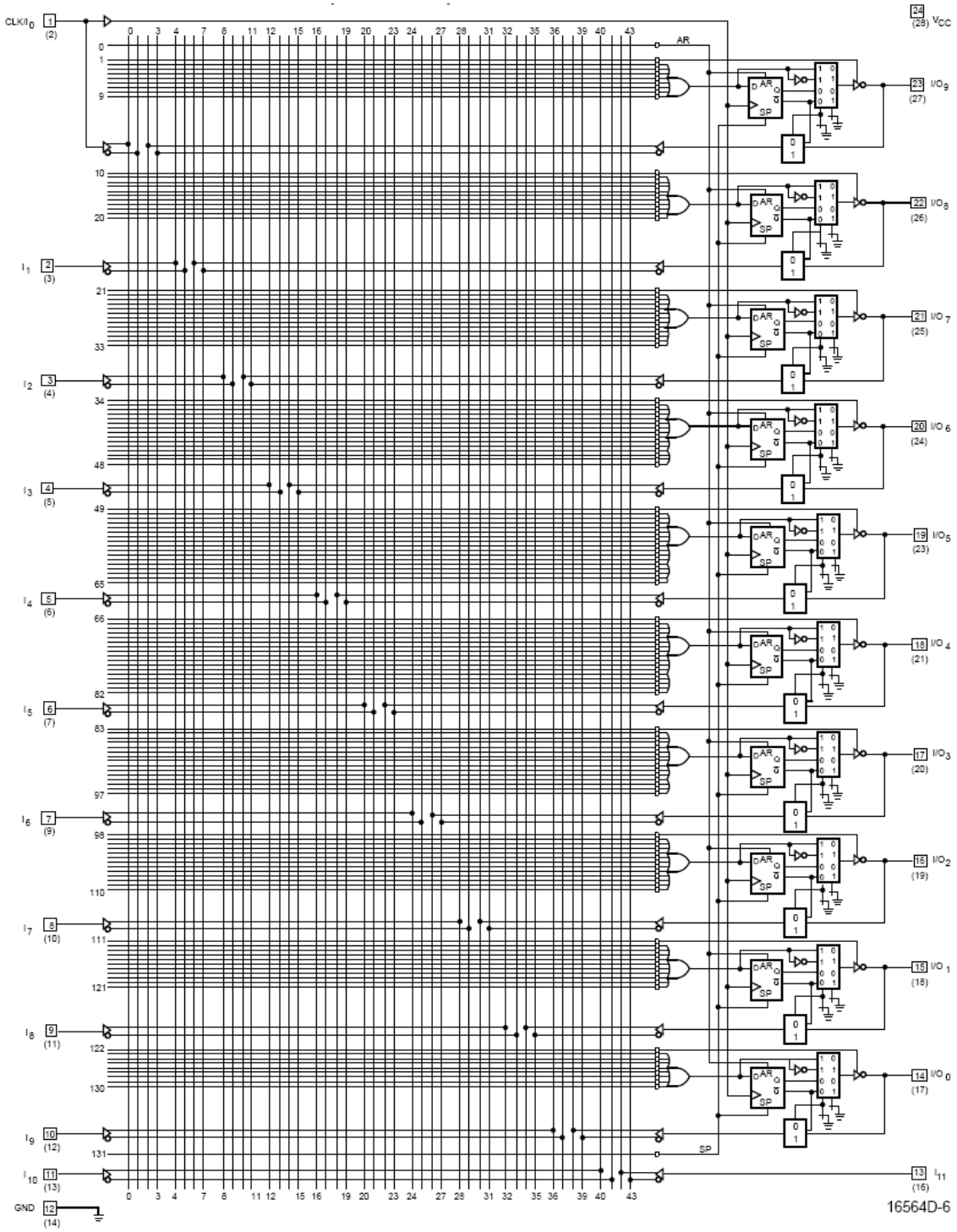


Figura 25 – Arquitectura completa da PAL® ATF22V10 (imagem extraída de [7])

5.2.4 Arquitectura de PAL® sequencial ATF750C/CL

A PAL® ATF22V10 foi um passo em frente na flexibilização da arquitectura 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 optimizar a lógica combinatória com a utilização de flip-flops mais adequados ao problema, etc. Com vista a melhorar todos estes aspectos, 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® ATV750B/BL e ATV750/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 pode 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 activação 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 26 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 directa de relógio. O *preset* de todos os flip-flops é controlado por um sinal síncrono comum a todos proveniente de um termo de produto;

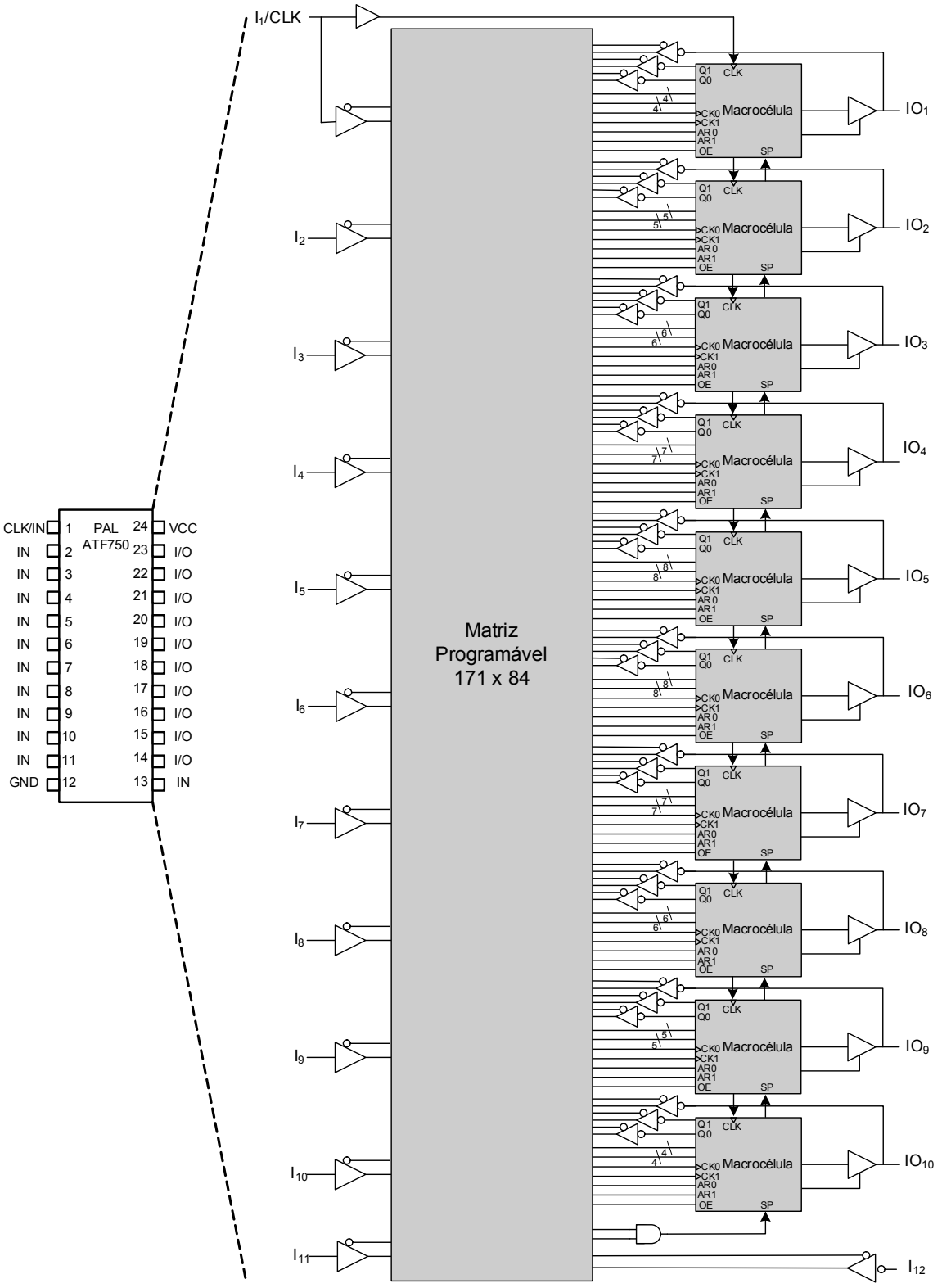


Figura 26 – Arquitectura completa da PAL® ATF750C/CL

- 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 inactivo. Caso o pino seja configurado como saída, o *three-state* está sempre activo. 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 bidireccional. Os pinos de saída incluem ainda um controlo para programação da polaridade.

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 26 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 IO_1 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 selecciona a origem do sinal de relógio aplicado a cada um dos flip-flops (ver figura 27).

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 selector *Sel1* configurável.

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

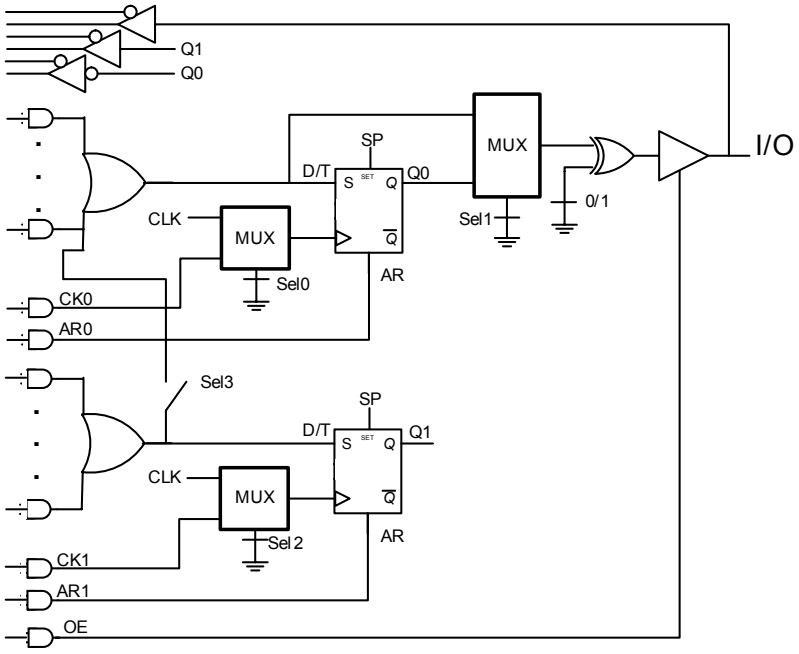


Figura 27 – Arquitectura da macrocélula de saída da PAL® ATF750C/CL

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 selector *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 27), 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:

a) Saída combinatória;
b) Saída combinatória mais um flip-flop;
c) Saída sequencial;
d) Saída sequencial mais um flip-flop;
e) Utilização de dois flip-flops com pino de I/O usado como entrada;
f) 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 que o flip-flop Q0 pode ser interno ou estar associado a um pino.

a) Configuração com saída combinatória

Esta é a configuração mais simples, em que não são utilizados flip-flops (ver figura 28).

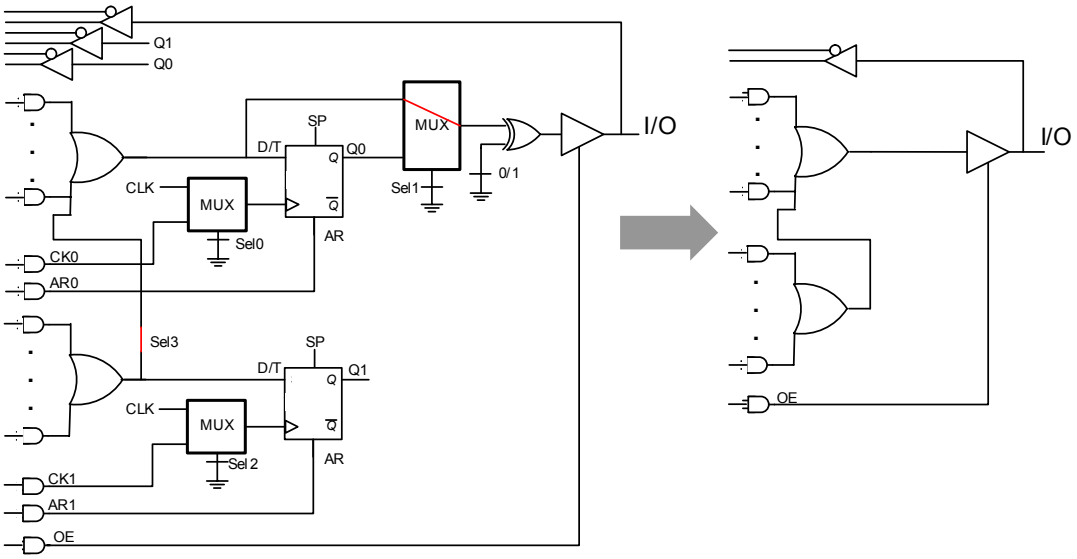


Figura 28 – Configuração com saída combinatória

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.

b) Configuração com saída combinatória mais um flip-flop

Nesta configuração, é produzida uma saída combinatória com o termo de soma superior e o termo de soma inferior é suado juntamente com o seu flip-flop para produzir uma variável Q1 registada (ver figura 29).

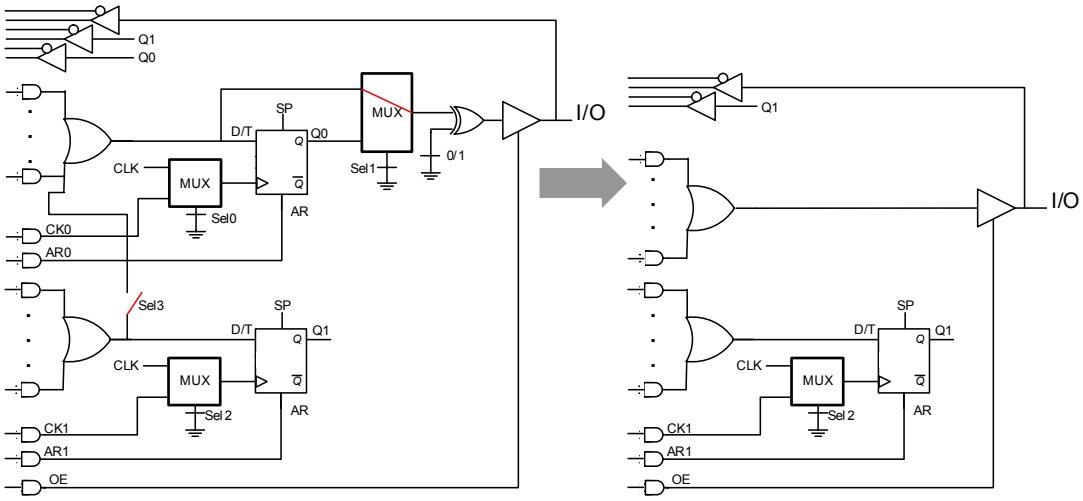


Figura 29 – Configuração com saída combinatória mais um flip-flop

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.

c) Saída sequencial

O modo mais simples de gerar uma saída sequencial é utilizando o flip-flop superior que tem um caminho directo 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 directo para uma saída, tendo de ser realimentado para a matriz para poder aparecer numa saída combinatória (ver figura 30).

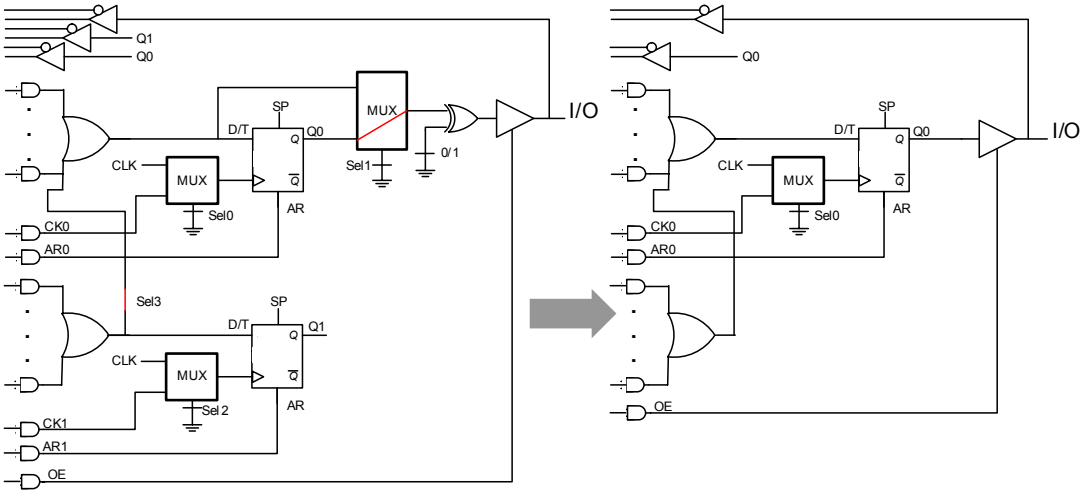


Figura 30 – Configuração com saída sequencial

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

d) 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 figura 31).

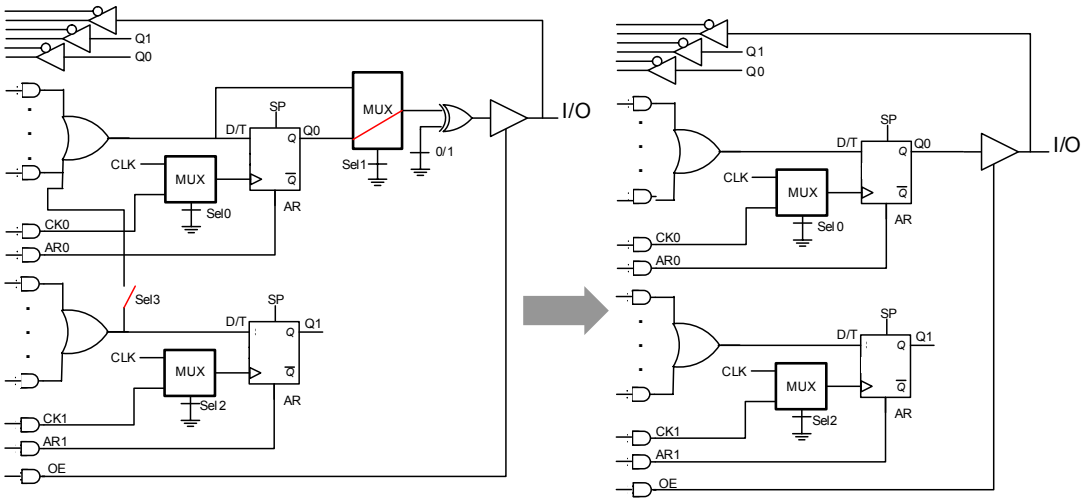


Figura 31 – Configuração com saída sequencial mais um flip-flop interno

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

A ATF750C/CL tem a vantagem de ter realimentações directamente das saídas dos flip-flops. Como tal, os flip-flops podem ser usados sem estarem associados a qualquer pino I/O, libertando o pino I/O para poder ser usado como entrada (ver figura 32).

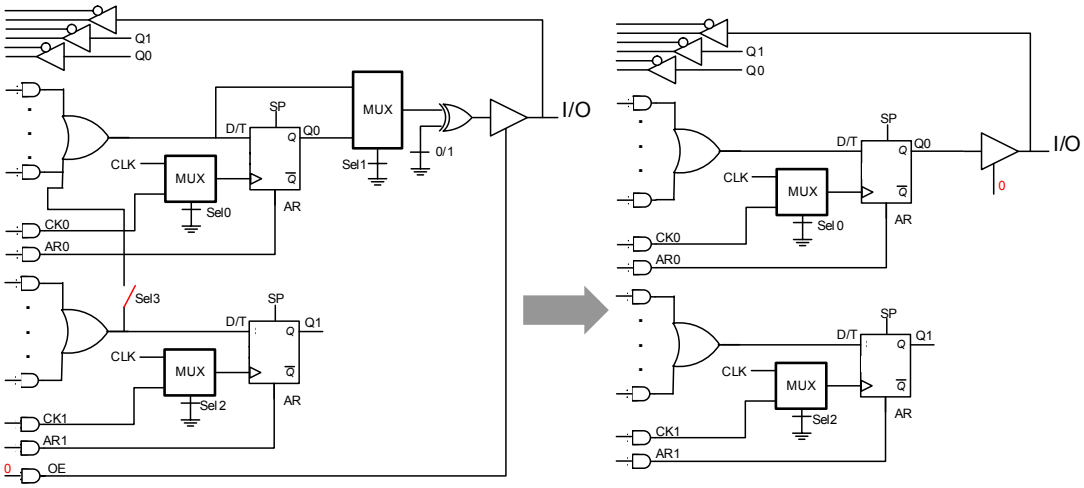


Figura 32 – Configuração com dois flip-flops internos e o pino usado como entrada combinatória

Repare que nesta configuração o *three-state* de saída está sempre inactivo (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.

f) 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 respectivo (ver figura 33).

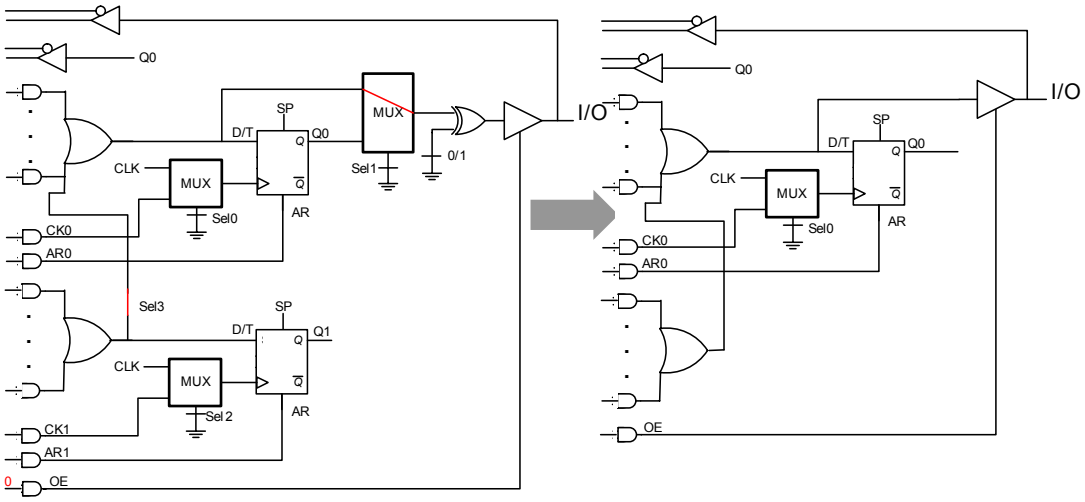


Figura 33 – Configuração com saída combinatória registada num flip-flop interno

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.

5.3 Configuração das PAL® com a linguagem CUPL

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.

5.3.1 Configuração da PAL® ATF22V10 usando CUPL

Recordando a macrocélula da ATF22V10 (ver figura 34), 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 flip-flops e está sempre associado ao pino 1 da PAL®.

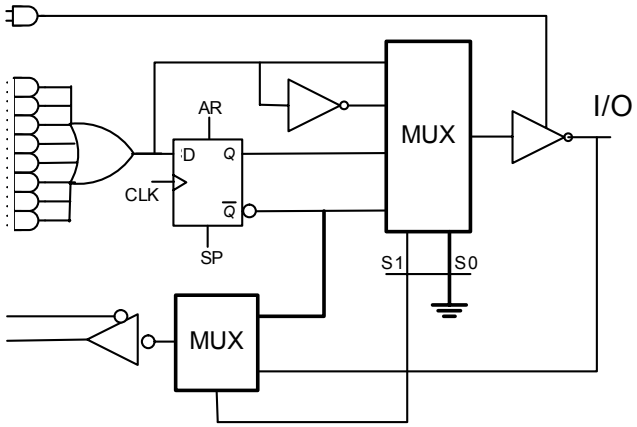


Figura 34 – Macrocélula da ATF22V10

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-se especificar 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.

5.3.2 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 35.

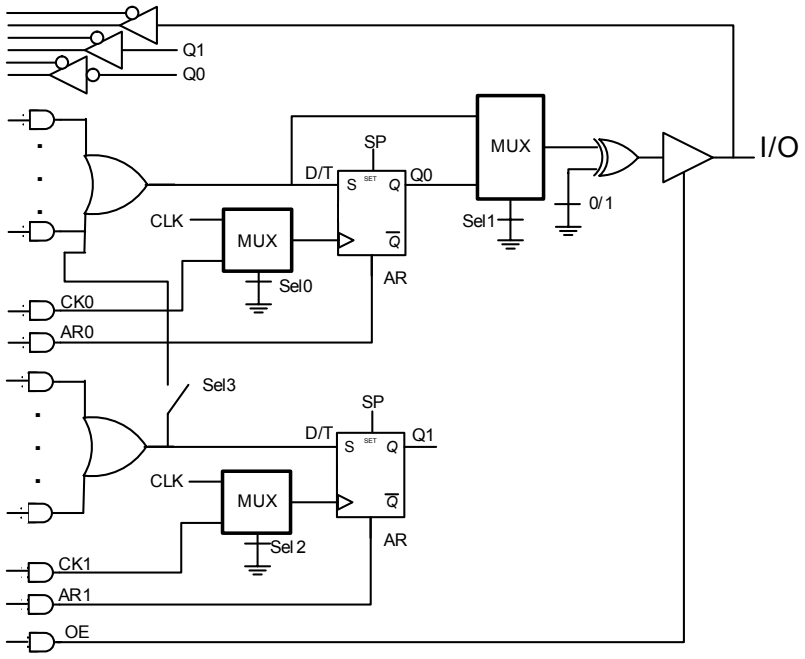


Figura 35 – Macrocélula da ATF750X

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 tabela 5).

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 5 – 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.

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 figura 36).

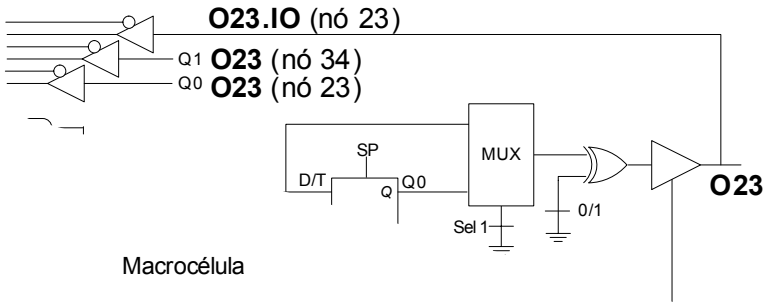


Figura 36 – Identificação dos pontos de realimentação

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.

NOTA: Se um pino de I/O for usado como entrada, esta é identificada apenas com o nome do pino. A extensão .IO é usada apenas para realimentação do valor de um pino de saída registado.

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 figura 37).

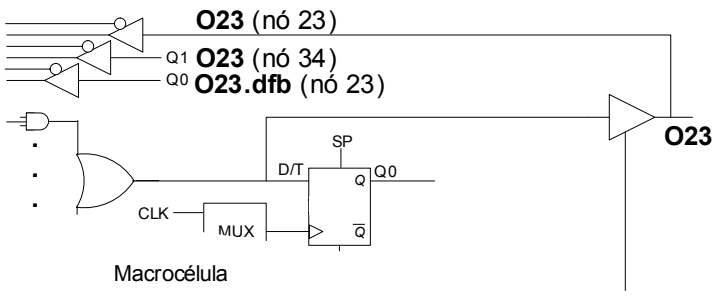


Figura 37 – Identificação do ponto de realimentação com flip-flop e saída combinatória

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 directamente 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 directamente 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-se usar 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 directo 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 directamente, enquanto que 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, respectivamente, 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.