Universidade Federal Fluminense – UFF
Escola de Engenharia – TCE
Curso de Engenharia de Telecomunicações – TGT

Programa de Educação Tutorial – PET
Grupo PET-Tele

Tutorial PET-Tele

Introdução aos Circuitos Digitais Configuráveis

(Versão: A2019M01D14)

Autores: Alexandre Santos de la Vega (2019)
Tutor: Alexandre Santos de la Vega

Niterói – RJ
Janeiro / 2019

Capítulo 1

Circuitos programáveis

1.1 Introdução

De acordo com a função realizada por um circuito digital, ele pode ser classificado em: circuito fixo (ou invariante no tempo) e circuito variável (ou variante no tempo). Como os próprios nomes já indicam, um circuito fixo realiza uma única função, enquanto um circuito variável pode permitir a realização de um conjunto de funções diferentes.

Costuma-se dizer que circuitos variáveis são programáveis. Embora não esteja errado, o termo programável induz ao pensamento de que o circuito é capaz de executar o que hoje é denominado de um programa de computador. Logo, talvez seja mais indicado dizer que os circuitos variáveis são configuráveis, deixando o termo programável para indicar um tipo específico de circuito configurável. A partir dessa denominação, podem-se definir diversos tipos de configurações e, portanto, diversos tipos de circuitos variáveis.

Pode-se pensar em dividir a configuração dos circuitos variáveis em: externa e interna.

Na configuração dita externa, podem-se agrupar os circuitos que possuem uma estrutura fixa, mas que mudam de função de acordo com as combinações de valores aplicados em algumas de suas entradas. Esse tipo de configuração pode ser usado em circuitos digitais combinacionais e/ou seqüenciais que ocupam uma extensa faixa de complexidade: portas lógicas, funções lógicas, circuitos digitais básicos, blocos funcionais e até mesmo sistemas digitais mais complexos.

Na configuração denominada de interna, podem-se reunir os circuitos que possuem uma estrutura com elementos constituintes fixos, mas que podem sofrer modificações na sua interconexão (roteamento), no conteúdo de informação armazenada (dado) ou em ambos. Esse tipo de configuração é preferencialmente utilizado em circuitos digitais complexos, que podem implementar circuitos combinacionais e/ou seqüenciais, tais como: Dispositivo Lógico Programável (ou *Programmable Logic Device* ou PLD) e processador (microprocessador, microcontrolador e processador de sinal digital).

Naturalmente, sistemas computacionais de complexidade elevada utilizam todos os tipos de circuitos (fixos e variáveis).

De acordo com a definição apresentada acima, a Tabela 1.1 resume a classificação de circuitos digitais de acordo com a função realizada.

A seguir, é apresentada uma introdução sobre o assunto. Deve ser ressaltado que, devido ao fato de apresentarem um conteúdo de alta complexidade e de alta especificidade, não serão abordados os diversos tipos de processadores nem os incontáveis sistemas computacionais de complexidade elevada.

Circuito digital	Fixo		
	Variável	Configuração externa	
		Configuração interna	Troca de roteamento
			Troca de dados
			Troca de ambos

Tabela 1.1: Classificação de circuitos digitais de acordo com a função realizada.

1.2 Circuitos fixos: Decodificador

O Decodificador (*Decoder*) é um bloco com N entradas e $L \leq 2^N$ saídas. Ele também é conhecido pelas seguintes denominações: Decodificador $N-to-2^N$ ($N-to-2^N$ *Decoder*), Decodificador $N-to-L$ ($N-to-L$ *Decoder*), Decodificador de Endereços (*Address Decoder*) e Decodificador de Linha (*Line Decoder*).

O modelo para um decodificador genérico ($L = 2^N$), baseado em portas lógicas, é um único plano com 2^N portas do tipo AND, de N entradas cada. Combinações das N entradas do decodificador, e das suas negações, são adequadamente ligadas às portas AND, a fim de que cada uma delas gere o n-ésimo mintermo $m(n)$ correspondente, para $0 \leq n \leq (2^N - 1)$. Por sua vez, cada mintermo gerado representa uma das L saídas do decodificador.

Na implementação de um decodificador com N pequeno, pode-se utilizar diretamente o modelo descrito acima, possivelmente com pequenas adequações de ordem prática. Por outro lado, para N grande, a quantidade de pinos de entrada em cada porta AND demanda o emprego de uma estrutura alternativa. Uma solução comumente empregada é uma combinação, em cascata, de um arranjo paralelo de decodificadores menores com arranjos paralelos de portas do tipo AND adicionais. Por sua vez, nessa estrutura, com N crescente, pode ser necessário que a saída de cada porta AND seja ligada a um número muito grande de outras portas AND. A solução alternativa é usar uma estrutura em árvore (*tree decoder*). Pode-se propor ainda uma solução que utilize a mistura de ambas as alternativas.

Embora o decodificador não seja um circuito configurável, ele é um importante elemento primitivo na modelagem de diversos blocos configuráveis, devido à sua funcionalidade de geração de mintermos.

1.3 Circuitos configuráveis externamente

A seguir, são apresentados alguns exemplos de circuitos configuráveis externamente. São abordadas as configurações de portas lógicas, de funções lógicas, de circuitos digitais básicos, de blocos funcionais e de sistemas digitais mais complexos.

1.3.1 Configuração de portas lógicas

Nesse caso, a configuração externa do dispositivo possibilita a implementação uma porta lógica específica, dentro de um conjunto fixo e reduzido de opções.

Em algumas aplicações, é interessante que o dispositivo implemente apenas uma porta lógica. Isso facilita o roteamento na placa de circuito impresso, uma vez que a porta lógica é localizada diretamente na trajeto do sinal, evitando eventuais desvios de trilhas. Como conseqüência, pode-se ter trilhas diretas e mais curtas, reduzindo resistências e acoplamentos eletromagnéticos das trilhas, o que se traduz em menores tempos de resposta. Além disso, se

for necessário realizar alguma modificação simples no circuito, torna-se mais fácil redesenhar a placa de circuito impresso.

Por exemplo, no início dos anos 2000, a Fairchild Semiconductors[1] fabricava uma família de dispositivos denominada de TinyLogic. Os dispositivos dessa família possuíam características adequadas ao uso em aparelhos portáteis, tais como: baixo custo, dimensão reduzida, baixo consumo e reduzido tempo de resposta. Existiam as seguintes séries: HS (*High Speed*), HT (*High speed Ttl*), UHS (*Ultra-High Speed*) e ULP (*Ultra-Low Power*). Um exemplo de dispositivos da série UHS são o NC7SZ57 e o NC7SZ58. Algumas das características apresentadas por esses dispositivos eram:

- Podiam ser configuradas cinco portas lógicas diferentes, com possíveis negações lógicas nas entradas e/ou na saída. As portas eram: AND, NAND, OR, NOR e XOR/XNOR.
- A saída de um deles era logicamente complementar a do outro.
- No caso da alimentação assumir o valor de tensão nulo, tanto as entradas quanto a saída assumiam um estado de alta impedância.
- A corrente de saída era suficiente para acionar diretamente alguns dispositivos, tais como: LED, acoplador óptico e relé de baixa corrente.
- Todas as entradas possuíam histerese.
- O encapsulamento (SC70) era do tipo SMD (*Surface-Mounted Device*), com dimensões reduzidas (2,00 mm $\times$ 1,25 mm) e organização DIL (*Dual In Line*) de seis pinos, sendo: dois pinos para alimentação (V_{CC} e GND), três pinos de entrada e um pino de saída.

A Tabela 1.2 descreve o comportamento funcional dos dispositivos NC7SZ57 e NC7SZ58. Por sua vez, a Tabela 1.3 descreve as opções de configuração dos dispositivos. Em ambas as tabelas, são feitas as seguintes associações: "1" = $V_{HIGH} = V_{CC}$ e "0" = $V_{LOW} = GND$.

Cabe lembrar que se pode obter um segundo conjunto de portas diferentes, porém logicamente equivalentes ao primeiro, por meio da aplicação dos Teoremas de De Morgan e das seguintes relações: A XOR $B \equiv \overline{A}$ XNOR $B \equiv A$ XNOR $\overline{B}$ e A XNOR $B \equiv \overline{A}$ XOR $B \equiv A$ XOR $\overline{B}$.

Entradas			Saída	
E_2	E_1	E_0	NC7SZ57	NC7SZ58
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

Tabela 1.2: Comportamento funcional dos dispositivos NC7SZ57 e NC7SZ58.

[1] http://www.fairchildsemi.com

Entradas	Saída	
	NC7SZ57	NC7SZ58
$E_0 = 0$	$\overline{E_1}$ NAND E_2	$\overline{E_1}$ AND E_2
$E_0 = 1$	E_1 AND E_2	E_1 NAND E_2
$E_1 = 0$	E_0 NOR E_2	E_0 OR E_2
$E_1 = 1$	E_0 NAND $\overline{E_2}$	E_0 AND $\overline{E_2}$
$E_2 = 0$	$\overline{E_0}$	E_0
$E_2 = 1$	E_1	$\overline{E_1}$
$E_0 = E_1$	E_0 XNOR E_2	E_0 XOR E_2
$E_0 = E_2$	E_0 NAND $\overline{E_1}$	E_0 AND $\overline{E_1}$
$E_1 = E_2$	$\overline{E_1}$ NAND E_0	$\overline{E_1}$ AND E_0

Tabela 1.3: Opções de configuração dos dispositivos NC7SZ57 e NC7SZ58.

1.3.2 Configuração de funções lógicas com multiplexador

Um multiplexador (*multiplexer* ou MUX) digital é um dispositivo com N sinais de entrada E_n, M sinais de controle C_m e 1 sinal de saída S, onde $M \in \mathbb{N}^+$ e $N \leq 2^M$.

Um MUX tem a função de um seletor. A saída S deverá assumir o mesmo valor da entrada E_n, quando o padrão binário presente em $\boldsymbol{C} = [\ C_{M-1}\ C_{M-2}\ \cdots\ C_1\ C_0\ ]$ apresentar o valor $(\boldsymbol{C})_2 = n = [\boldsymbol{C}]_n$. Logo, os padrões binários de $\boldsymbol{C}$ podem ser interpretados como endereços que serão empregados na seleção do sinal de entrada que será copiado para a saída do MUX.

O modelo básico para um MUX, baseado em portas lógicas, é um arranjo padrão AND-OR, composto de um plano com N portas AND de $(M+1)$ entradas, seguido de um plano com 1 porta OR de N entradas. Nesse modelo, cada porta AND realiza a operação

$$AND\ (E_n, [\boldsymbol{C}]_n) = AND\ (E_n, [\ C_{M-1}\ C_{M-2}\ \cdots\ C_1\ C_0\ ]_n) = AND\ (E_n, m(n))\ ,$$

onde $m(n)$ representa o mintermo n. Portanto, enquanto a saída da porta AND endereçada pelo mintermo n apresenta uma cópia da entrada E_n do MUX, as demais portas AND terão saída igual a "0". Por sua vez, a porta OR completa a cópia da entrada selecionada para a sua saída, que é a saída do MUX.

Uma vez que a função básica de um MUX é a de um seletor, pode-se pensar em utilizá-lo para implementar uma função lógica genérica, por meio da configuração adequada das suas entradas (E_n e C_m).

Em uma primeira tentativa, basta interpretar a tabela verdade (TV) de uma função lógica de V variáveis como um processo de seleção de valores "0" e "1", onde os endereços da seleção são os padrões apresentados pelo conjunto de variáveis. Portanto, uma dada função lógica de V variáveis pode ser implementada por um MUX com $M = V$ e $N = 2^M$, conectando-se as M variáveis aos sinais C_m do MUX e fornecendo-se os valores "0" e "1" às entradas E_n do MUX, de acordo com a TV da função alvo.

Com o intuito de diminuir a quantidade de circuito necessária para implementar uma mesma função lógica, pode-se pensar em utilizar um MUX menor. Para isso, deve-se observar que, nas funções lógicas de V variáveis, haverá 2 valores na TV para cada combinação de $(V-1)$ variáveis. Esses valores poderão ser iguais a "00", "01", "10" ou "11", bem como poderão ser associados a "0", "1", V_k e $\overline{V_k}$, onde V_k é a variável desconsiderada no endereçamento. Portanto, pode-se implementar uma dada função lógica de V variáveis por um MUX com $M = (V-1)$ e $N = 2^M$, conectando-se as M variáveis escolhidas aos sinais C_m do MUX e fornecendo-se os valores "0",

"1", V_k ou $\overline{V_k}$, às entradas E_n do MUX, de acordo com a TV da função alvo. A economia é significativa, pois o MUX é reduzido à metade e, mesmo no caso onde o complemento de uma das variáveis não esteja à disposição, é necessário que se acrescente apenas um inversor.

Pode-se obter uma redução ainda maior no tamanho do MUX, em troca da inclusão de um circuito lógico adicional. Para isso, deve-se observar que, nas funções de $V = (K + L)$ variáveis, haverá 2^L valores na TV para cada combinação de K variáveis. Esses valores poderão ser associados a "0", "1" e aos valores provenientes de combinações lógicas das L variáveis desconsideradas no endereçamento. Portanto, pode-se implementar uma dada função lógica de $V = (K + L)$ variáveis por um MUX com $M = K$ e $N = 2^M$, conectando-se as M variáveis escolhidas aos sinais C_m do MUX e fornecendo-se os valores "0", "1" e de combinações lógicas das L variáveis não utilizadas no endereçamento, às entradas E_n do MUX, de acordo com a TV da função alvo. Nesse caso, deve-se fazer um balanceamento entre a redução da quantidade de circuito do MUX e o aumento da quantidade de circuito adicional.

Deve ser ressaltado que, em todos os casos, diversas soluções podem ser propostas, dependendo de quais variáveis serão escolhidas para o endereçamento e da sua ordenação. Dentro desse universo de soluções, algumas delas poderão requerer menos circuitos do que outras.

A título de exemplo, será considerada a função lógica definida por

$$\begin{aligned} f(A,B,C,D) &= \sum m(1,3,5,7,10,11,12,13) = \prod M(0,2,4,6,8.9,14,15) \\ &= \left(\overline{A}\cdot D\right) + \left(A\cdot\overline{B}\cdot C\right) + \left(A\cdot B\cdot\overline{C}\right) \\ &= (A+D)\cdot\left(\overline{A}+B+C\right)\cdot\left(\overline{A}+\overline{B}+\overline{C}\right) \ . \end{aligned} \tag{1.1}$$

A Figura 1.1 ilustra algumas formas de organização do Mapa de Karnaugh da função exemplo, para implementação usando MUX.

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

(a)

AC \ BD	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	1	0	0
10	0	0	1	1

(b)

AD \ BC	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	1	0	1
10	0	1	0	1

(c)

Figura 1.1: Mapas de Karnaugh da função exemplo, para implementação usando MUX.

Para um MUX com $M = 4$ e $N = 16$, pode ser adotada a seguinte configuração:

$$\begin{cases} [\ C_3\ C_2\ C_1\ C_0\ ] = [\ A\ B\ C\ D\ ] \\ \\ [\ E_{15}\ E_{14}\ \cdots\ E_1\ E_0\ ] = [\ 0\,0\,1\,1\,1\,1\,0\,0\,1\,0\,1\,0\,1\,0\,1\,0\ ] \end{cases} .$$

Para um MUX com $M = 3$ e $N = 8$, podem ser adotadas as seguintes configurações:

$$\begin{cases} [\ C_2\ C_1\ C_0\ ] \ = \ [\ B\ C\ D\ ] \\ [\ E_7\ E_6\ \cdots\ E_1\ E_0\ ] \ = \ [\ \overline{A}\ 0\ 1\ A\ 1\ A\ \overline{A}\ 0\ ] \end{cases}$$

e

$$\begin{cases} [\ C_2\ C_1\ C_0\ ] \ = \ [\ A\ B\ C\ ] \\ [\ E_7\ E_6\ \cdots\ E_1\ E_0\ ] \ = \ [\ 0\ 1\ 1\ 0\ D\ D\ D\ D\ ] \end{cases}.$$

Para um MUX com $M = 2$ e $N = 4$, podem ser adotadas as seguintes configurações:

$$\begin{cases} [\ C_1\ C_0\ ] \ = \ [\ A\ B\ ] \\ [\ E_3\ E_2\ E_1\ E_0\ ] \ = \ [\ \overline{C}\ C\ D\ D\ ] \end{cases},$$

$$\begin{cases} [\ C_1\ C_0\ ] \ = \ [\ A\ C\ ] \\ [\ E_3\ E_2\ E_1\ E_0\ ] \ = \ [\ \overline{B}\ B\ D\ D\ ] \end{cases},$$

$$\begin{cases} [\ C_1\ C_0\ ] \ = \ [\ A\ D\ ] \\ [\ E_3\ E_2\ E_1\ E_0\ ] \ = \ [\ (B \oplus C)\ (B \oplus C)\ 1\ 0\ ] \end{cases},$$

$$\begin{cases} [\ C_1\ C_0\ ] \ = \ [\ C\ D\ ] \\ [\ E_3\ E_2\ E_1\ E_0\ ] \ = \ [\ (\overline{A} + \overline{B})\ (A \cdot \overline{B})\ (\overline{A} + B)\ (A \cdot B)\ ] \end{cases},$$

$$\begin{cases} [\ C_1\ C_0\ ] \ = \ [\ B\ D\ ] \\ [\ E_3\ E_2\ E_1\ E_0\ ] \ = \ [\ (\overline{A} + \overline{C})\ (A \cdot \overline{C})\ (\overline{A} + C)\ (A \cdot C)\ ] \end{cases}$$

e

$$\begin{cases} [\ C_1\ C_0\ ] \ = \ [\ B\ C\ ] \\ [\ E_3\ E_2\ E_1\ E_0\ ] \ = \ [\ (\overline{A} \cdot D)\ (A + D)\ (A + D)\ (\overline{A} \cdot D)\ ] \end{cases}.$$

1.4 Circuitos configuráveis internamente

Os circuitos configuráveis internamente podem ser agrupados em uma única classe, denominada de Dispositivo Lógico Programável (ou *Programmable Logic Device* ou PLD). Os PLDs comerciais passaram por uma linha evolutiva de complexidade em relação às estruturas dos circuitos utilizados.

Os blocos básicos configuráveis que deram origem aos atuais PLDs são os seguintes:

- Decodificador (*Decoder*).
- MUX (*Multiplexer*).
- ROM (*Read-Only Memory*) ou conversor de código (*code converter*).
- PLA (*Programmable Logic Array*).
- PAL (*Programmable Array Logic*).

A partir dos blocos básicos configuráveis, e em ordem crescente de complexidade, os PLDs evoluiram sob as seguintes denominações:

- SPLD (*Simple Programmable Logic Device*)
  - PLA.
  - PAL.
  - PLA/PAL com memória: um *flip-flop* do tipo D (DFF) em cada saída.
  - GAL (*Generic Array Logic*).
  - PALCE (*PAL CMOS Electrically erasable/programable device*).
- CPLD (*Complex Programmable Logic Device*).
- FPGA (*Field-Programmable Gate Array*).

Os blocos básicos configuráveis, bem como cada um dos PLDs listados acima, são brevemente abordados a seguir.

Em relação aos blocos básicos configuráveis, serão discutidas apenas as estruturas dos circuitos lógicos. A implementação final de cada circuito depende de diversos fatores e não será aqui apresentada.

No tocante aos PLDs, serão abordadas apenas as seções básicas e a sua organização, dadas a elevada complexidade desses circuitos e a sua variação entre fabricantes.

1.4.1 Blocos básicos configuráveis

Os blocos básicos configuráveis abordados a seguir são fundamentados na geração de mintermos e na construção de formas padrões do tipo SOP (*Sum Of Products*), ou ainda na geração de implicantes minimizados e na construção de formas SOP minimizadas.

Com a exceção do bloco MUX, que apresenta uma única saída, todos os demais blocos podem ser interpretados como blocos multifunções, que geram simultaneamente L diferentes funções, a partir dos N sinais de entrada.

MUX

Conforme definido anteriormente, um multiplexador (*multiplexer* ou MUX) digital tem a função de um seletor e é um bloco com N sinais de entrada E_n, M sinais de controle C_m e 1 sinal de saída S, onde $M \in \mathbb{N}^+$ e $N \leq 2^M$.

A partir da sua definição, um MUX pode ser modelado pela combinação de um decodificador $M-to-N$ com N portas lógicas do tipo AND, e dessas com uma porta lógica do tipo OR. O decodificador recebe os M sinais de controle C_m e gera os N mintermos $m(n)$, onde: $N \leq 2^M$ e $0 \leq n \leq (N-1)$. As N portas AND recebem os N mintermos $m(n)$ e as N entradas E_n, realizando uma seleção. Por fim, a porta OR combina as saídas de todas as portas AND de seleção, gerando a saída única S.

Uma outra forma de visualizar o MUX é a combinação de um "decodificador modificado" com uma porta OR. O decodificador modificado representa a unificação das portas AND do decodificador com as portas AND de seleção.

Uma vez que a estrutura interna de um dado MUX é fixa, deve-se lembrar que a sua reconfiguração, para implementar uma função lógica genérica, é realizada externamente, por meio da configuração adequada das suas entradas (E_n e C_m).

ROM

Um bloco funcional ROM (*Read-Only Memory*) ou conversor de código (*code converter*) é um circuito com N entradas e L saídas, onde cada saída implementa uma determinada relação funcional das N entradas.

Empregando-se uma síntese funcional por meio de uma forma padrão SOP, um bloco de tamanho $N \times L$ pode ser modelado pela combinação de um decodificador $N-to-2^N$ com um plano de L portas lógicas do tipo OR. O decodificador gera todos os 2^N mintermos possíveis, a partir das suas N entradas, enquanto cada uma das L portas OR combina apenas os mintermos que forem especificados para gerar um determinada função, por meio de uma forma padrão SOP.

Se o conjunto de N entradas for interpretado como uma única palavra de um conjunto de códigos de entrada, enquanto o conjunto de L saídas for interpretado como uma única palavra de um conjunto de códigos de saída, o bloco pode ser denominado de conversor de código.

Por outro lado, se as entradas forem pensadas como endereços de registradores e as saídas do conjunto de portas OR forem interpretadas como um dado único, armazenado no registrador endereçado, pode-se dizer que o bloco é uma memória. Uma vez que as conexões são fixas ou, no máximo, reconfiguráveis para um uso fixo, ela é dita uma memória de conteúdo fixo (*Read-Only Memory* ou ROM).

Deve-se destacar que, em um bloco ROM, o plano de portas AND de entrada (decodificador) é uma estrutura fixa, enquanto o plano de portas OR de saída é a parte configurável.

Se o plano de portas OR puder ser reconfigurável, o bloco recebe a denominação de PROM (*Programmable* ROM). Dependendo da forma como a configuração é desfeita (dados apagados) e da forma como ela é refeita (novos dados escritos), diversas denominações são utilizadas para identificar o bloco de memória ROM.

Se a configuração for realizada pelo fabricante, a ROM é denominada de *mask-programmable*. Por outro lado, se ela for realizada pelo usuário, a ROM é denominada de *field-programmable*.

No caso da implementação de vários conjuntos de funções lógicas que dependam das mesmas variáveis (ou de várias ROMs com um mesmo endereçamento), pode-se otimizar o circuito final, utilizando-se um único decodificador e vários conjuntos configuráveis de portas OR.

PLA

Um PLA (*Programmable Logic Array*) é um bloco com N entradas e L saídas. Quando a quantidade de sinais de entrada (N) é elevada, o emprego de um decodificador padrão necessita de uma quantidade muito mais elevada (2^N) de portas AND, com N entradas cada. Além disso, é provável que a maioria dos mintermos gerados pelo decodificador não seja utilizada. Tais situações motivam o uso de uma estrutura de "decodificador simplificado e programável", onde é utilizado um conjunto menor de portas AND e as suas conexões com os sinais de entrada são configuráveis. Deve-se notar que, diferentemente do caso do decodificador padrão, onde as 2^N portas AND geram todos os possíveis mintermos (implicantes padrões), é utilizado um arranjo minimizado e configurável de portas AND, que são capazes de gerar implicantes minimizados. Por sua vez, assim como no caso do bloco ROM, as conexões entre as portas AND, que geram os implicantes, e as portas OR, que geram as saídas, também são configuráveis. Logo, o PLA é um bloco com estrutura duplamente configurável (entrada/AND e saída/OR).

Valores típicos para os parâmetros de um PLA são os seguintes:

- Entradas (N): 10 a 20.
- Portas AND (implicantes): 30 a 60.
- Portas OR (L): 10 a 20.

PAL

Um PAL (*Programmable Array Logic*) é um bloco com N entradas e L saídas.

A possibilidade de configuração das conexões entre as portas AND, que geram os implicantes, e as portas OR, que geram as saídas, tal como acontece nos blocos ROM e PLA, só é vantajosa nos casos onde os mesmos implicantes são utilizados para sintetizar diferentes funções de saída.

Os casos onde devem ser implementadas diferentes funções lógicas (L), com um número elevado de variáveis (N), mas que apresentam mínima dependência das mesmas variáveis, ainda motivam o uso de um "decodificador simplificado e programável". Porém, a configuração entre as portas AND dos implicantes e as portas OR das saídas não se faz necessária, pois cada saída necessita de diferentes implicantes. Nesses casos, basta que cada porta OR possua uma conexão fixa com um conjunto próprio de portas AND.

Seguindo o modelo acima, nos blocos PAL, o estágio de entrada (AND) é configurável enquanto o estágio de saída (OR) é fixo.

Tipicamente, podem ser encontrados blocos PAL com 10 a 16 entradas, porém com os mais variados arranjos de saídas e de portas AND para cada saída.

1.4.2 SPLD

SPLD sem memória

Os dispositivos PLA e PAL originais são considerados SPLDs sem memória.

PLA/PAL com memória

Os circuitos básicos configuráveis não se utilizam de elementos de memória. Logo, eles são capazes de implementar apenas circuitos combinacionais.

Para possibilitar a implementação de circuitos seqüenciais simples, foi criada uma extensão para os blocos PLA e PAL. Em cada sinal de saída do bloco foi acrescentado um *flip-flop* do tipo D (DFF).

Cada DFF pode ser pensado como um registrador de um *bit*. Por sua vez, o conjunto de DFFs em paralelo pode ser interpretado como um registrador com entrada paralela e saída paralela. Dessa forma, esses novos PLDs foram denominados de PLA e PAL com registrador (*registered* PLA/PAL).

GAL

O GAL (*Generic Array Logic*) foi uma evolução direta do bloco PAL com registrador.

Além da adição de DFFs nas saídas um bloco PAL, foram incluídos ainda portas lógicas, multiplexadores e *buffers* com controle de *tristate*.

De uma forma geral, o GAL é organizado em duas seções: um plano de portas AND e um plano de macrocélulas. O plano de portas AND representa a seção de entrada configurável de um PAL. Em uma PAL, cada conjunto de portas AND é conectado a uma porta OR, que gera uma das saídas do dispositivo. Em uma GAL, cada conjunto de portas AND é conectado a uma macrocélula ou OLMC (*Output Logic MacroCell*).

Uma macrocélula genérica é um conjunto formado por uma porta OR, acrescida de um DFF, de uma porta lógica XOR e de alguns multiplexadores.

Apesar das macrocélulas serem blocos fixos, elas podem ser configuradas externamente por sinais de controle, que são aplicados às portas lógicas e ao multiplexadores.

Um grau de flexibilidade extra ainda foi conseguido com a possiblidade de realimentação de um sinal da macrocélula para o plano de portas AND de entrada. O sinal realimentado pode ser a saída do DFF, a saída da macrocélula ou a saída de uma macrocélula adjacente.

Finalmente, na saída de cada macrocélula existe um *buffer* com controle de *tristate*, que gera uma das saídas do dispositivo. Caso ele seja desabilitado, a sua saída por ser usada como entrada.

PALCE

O PALCE (*PAL CMOS Electrically erasable/programable device*) foi um PLD com arquitetura similar ao GAL.

1.4.3 CPLD

A evolução natural dos SPLDs foi a associação de várias dessas estruturas, juntamente com complexas interconexões configuráveis e interfaces para entrada/saída de dados (I/O *drivers*), em um único dispositivo denominado de PLD Complexo (*Complex Programmable Logic Device* ou CPLD).

Nesses dispositivos, foi incorporado um acesso para programação e teste, denominado de JTAG, que foi definido pelo *Joint Test Action Group* e é especificado pelo padrão "IEEE 1149.1".

Em alguns CPLDs, foram utilizados conjuntos de GALs ou conjuntos de PLAs.

Em um caminho intermediário entre os SPLDs e os FPGAs (abordados a seguir), foram adotados conjuntos de PLDs formados por subconjuntos de estruturas básicas. Tais estruturas básicas podem ser interpretadas como arranjos "PAL+OLMC" bem mais complexos. Nesse caso, a estrutura PAL foi trocada por uma pequena memória (*LookUp Table* ou LUT), capaz de sintetizar qualquer função lógica de poucas variáveis. Por sua vez, a macrocélula passou a incorporar elementos úteis à otimização de soma/subtração e à conexão de registradores, para uso em circuitos seqüenciais mais complexos. Esse tipo de CPLD apresenta um esquema de interconexão ainda mais intrincado, com conexões locais e conexões globais. Conseqüentemente, esse tipo de CPLD é, por vezes, chamado de FPGA "simplificado".

1.4.4 FPGA

Procurando atender a projetos que exigem complexidade e desempenho mais elevados, os PLDs evoluíram para o denominado FPGA (*Field-Programmable Gate Array*).

De uma forma bem simplificada, a arquitetura básica de um FPGA é uma matriz de blocos programáveis, localmente intercaladas por roteamentos programáveis.

Os blocos programáveis são conjuntos de estruturas básicas. Entre FPGAs diferentes, as estruturas básicas podem variar de complexidade. Porém, em essência, elas possuem os mesmos elementos constituintes. A fim de sintetizar qualquer função lógica de poucas variáveis é empregada uma pequena memória (*LookUp Table* ou LUT). Encontram-se também DFFs, multiplexadores e, por vezes, somadores. Além disso, são incorporados elementos úteis à otimização de soma/subtração e à conexão de registradores, para uso em circuitos seqüenciais mais complexos.

Além dos blocos programáveis, são também incluídos outros blocos funcionais, destinados a atender às demandas de complexidade e desempenho: blocos SRAM, blocos DSP e blocos PLL. Os blocos SRAM (*Static Random Access Memory*) visam atender às demandas de memória para uso da aplicação. Os blocos DSP (*Digital Signal Processing*), destinados a facilitar a implementação de operações de DSP, são constituídos de multiplicadores paralelos, estruturas

MAC (*Multiply And Accumulate*) e registros de deslocamento (*shift registers*). Os blocos PLL (*Phase Locked Loop*) são utilizados no gerenciamento dos sinais de relógio (*clock*), tais como: multiplicação de *clock*, divisão de *clock*, deslocamento de fase e filtragem de *jitter*.