Capítulo 6

Unidade lógica e aritmética (ULA)

Neste capítulo, vamos tratar de uma estrutura digital combinacional presente em todas as unidades centrais de processamento, mais comumente chamadas de processador ou CPU. Vamos abordar mais detidamente a unidade lógica e aritmética, a ULA — ou ALU, do inglês arithmetic logic unit (VAHID, 2008).

Esse elemento pode ser utilizado de diversas formas, porém, o mais comum é que seja parte de um processador — e ele costuma ser o elemento central de qualquer processador, pois, em muitas arquiteturas, é o que determina a velocidade com que as informações são processadas.

A ULA é a parte do processador que efetivamente faz o processamento da informação, ou seja, é na ULA que são feitas as operações matemáticas de que precisamos em um produto desenvolvido por meio das técnicas de projeto digital.

Uma ULA tem a complexidade necessária para que o produto que está sendo desenvolvido possa fazer o processamento adequado aos objetivos traçados. Assim, se em um determinado produto for necessária a operação de soma, essa operação deverá estar implementada como uma das funções da ULA. Entretanto, nem sempre é preciso implementar todas as operações matemáticas que serão necessárias, cabendo ao desenvolvedor fazer uma avaliação entre custo e desempenho.

PARA PENSAR

A velocidade de uma CPU é dada pelo tempo de propagação de seus circuitos combinacionais. Em um processador, é comum que o circuito combinacional com maior tempo de propagação seja a ULA. Pense: como o número de bits interfere no tempo de propagação da ULA e, por consequência, na frequência de operação da CPU?

Para exemplificar essa abordagem, vamos trabalhar a hipótese de que nosso produto precisa executar uma multiplicação como uma das etapas para tomar uma decisão no algoritmo. Essa operação matemática de multiplicação poderá ser implementada por meio de um circuito especialista dentro da ULA, ou ser implementada pelo deslocamento de bits e soma condicional. Mas como o desenvolvedor vai tomar essa decisão?

Ele terá diante de si duas opções. A primeira é usar mais elementos lógicos (seja na forma de circuitos integrados ou na forma de área de silício dentro de um circuito mais complexo) e ter um tempo de propagação dentro da ULA mais elevado, para todas as operações da ULA, e implementar o multiplicador. A segunda opção é usar mais ciclos de trabalho do processador para implementar a mesma operação, porém, com um tempo de propagação menor para todas as operações que a ULA vai realizar para executar o algoritmo do produto em questão.

Uma vez que estiver claro para o desenvolvedor o impacto de cada abordagem, ele verificará os requisitos de desempenho do produto – para que este atinja a qualidade que os usuários esperam dele – e tomará sua decisão. Assim, o projeto de uma ULA ou a escolha de qual modelo de ULA será utilizado tem efeito direto no projeto de todo o resto do produto.

1 Construção de uma ULA básica

Uma vez que já temos noção do que é uma ULA, vamos construir uma ULA básica — que, apesar de básica, já atenderá a diversos requisitos para uso em um sistema digital bastante complexo. Para isso, talvez o mais importante seja identificar quais são os requisitos de uma ULA básica, ou seja, o que uma ULA deve apresentar para que seja considerada uma ULA.

Como já comentamos na introdução, o papel de uma ULA é implementar todas as operações lógicas e aritméticas nas informações que serão tratadas pelo produto em desenvolvimento, e é por aí que começamos.

Para implementar a nossa ULA básica, selecionamos algumas das operações mais comuns em qualquer projeto digital:

- soma;
- subtração;
- operação lógica AND;
- operação lógica OR;
- operação lógica NOT; e
- operação lógica XOR.

Para implementar todas as funções citadas, vamos usar alguns blocos básicos, mais simples, de lógica combinacional. Em seguida, vamos juntar esses blocos para formar a nossa ULA básica, a qual

pode ser implementada com 2, 4 ou 8 bits, ou ainda com mais bits de dados.

A nossa ULA terá como entradas de dados as entradas A e B, e como saída de dados a saída S. A função lógica a ser executada pela ULA será definida pelo seletor F. Na forma como vamos implementar a nossa ULA, ela pode ser feita com um barramento de 2 bits ou com um barramento de mais bits de dados — lembrando que o número de bits de dados determina o quão rápido os dados poderão ser manipulados pela ULA em questão.

2 Componentes de uma ULA básica

Uma das formas mais comuns de resolver um problema complexo é dividi-lo em problemas menores, os quais, por sua vez, podem ser novamente divididos, até que sejam simples e fáceis de resolver. Dessa forma, vamos começar a dividir nosso problema pelo número de bits que temos em nossa ULA. Ou seja, em vez de tentar descrever a nossa ULA com todas as entradas e saídas de uma única vez, vamos trabalhar um bit de cada vez; como todos os bits são iguais, basta ligarmos os bits de forma adequada e teremos nossa ULA básica formada.

É importante salientar que não vamos entrar em detalhes sobre como cada bloco é construído internamente, como cada porta lógica é otimizada etc. A ideia, neste capítulo, é construir uma visão sistêmica de como se constrói um circuito mais complexo a partir de elementos mais simples.

3 Operações lógicas

Dentre as operações que selecionamos para implementar em nossa ULA básica, podemos destacar:

- operação lógica AND;

- operação lógica OR;
- operação lógica NOT; e
- operação lógica XOR.

Com exceção da operação NOT, as demais são a saída binária entre cada bit das entradas A e B. Ou seja, no circuito que vamos criar para fazer cada bit de nossa ULA, vamos ter uma porta AND, uma porta OR e uma porta XOR – cada uma com duas entradas, uma ligada na entrada A e outra na entrada B –, além de uma porta NOT, que vamos ligar apenas na entrada A. Vamos ligar as saídas posteriormente.

4 Operações aritméticas

Além das operações lógicas, há duas operações aritméticas que vamos implementar na nossa ULA: soma e subtração. Para executá-las, vamos utilizar os circuitos somador completo e subtrator completo. Ambos os blocos possuem as entradas A, B e Te, além das saídas S e Ts.

No caso do somador, Te e Ts correspondem ao sinal de carry, que é o famoso "vai 1", sendo Te o "vai 1" de entrada e Ts, o "vai 1" de saída, para o próximo bit do somador.

No caso do subtratator, Te e Ts correspondem ao sinal de borrow, que é o "empresta 1" do circuito de subtração, sendo que o Te é o "empresta 1" vindo do bit menos significativo e que o Ts vai ligado ao Te do próximo bit mais significativo.

Não vamos, aqui, entrar em detalhes sobre a construção desse bloco, já que os capítulos 4 e 5 são dedicados a eles.

5 Multiplexador (MUX)

Agora que já descrevemos os circuitos que efetivamente realizam as operações, precisamos de uma forma de selecionar, na saída de cada bit, qual das operações será refletida na saída.

O circuito que implementa essa função lógica é chamado de multiplexador, ou MUX. Resumidamente, sua função é, dado um código de seleção, refletir na saída uma das entradas.

Para implementar o nosso MUX, precisamos selecionar uma de seis opções de operações disponíveis. Isso vai nos exigir ao menos 3 bits de seleção. Porém, com 3 bits de seleção, temos disponíveis oito possibilidades.

Como forma de aproveitar melhor os recursos disponíveis, podemos acrescentar mais duas operações lógicas:

- reset (em que zeramos todos os bits da saída); e
- cópia A (em que copiamos a entrada A na saída).

Como forma de simplificar a representação da nossa tabela-verdade, vamos fazer uma ligeira modificação na representação tradicional de tabelas-verdade.

Tabela 1 – Tabela-verdade do MUX

F2	F1	F0	S
0	0	0	E0
0	0	1	E1
0	1	0	E2
0	1	1	E3
1	0	0	E4
1	0	1	E5
1	1	0	E6
1	1	1	E7

Nessa tabela, podemos notar que, de acordo com o código colocado nas entradas de seleção F0, F1 e F2, estamos selecionando qual das entradas de dados entre E0 e E7 será refletida na saída. Assim, podemos usar o MUX para selecionar qual das operações lógicas vamos ter na saída da nossa ULA simplificada.

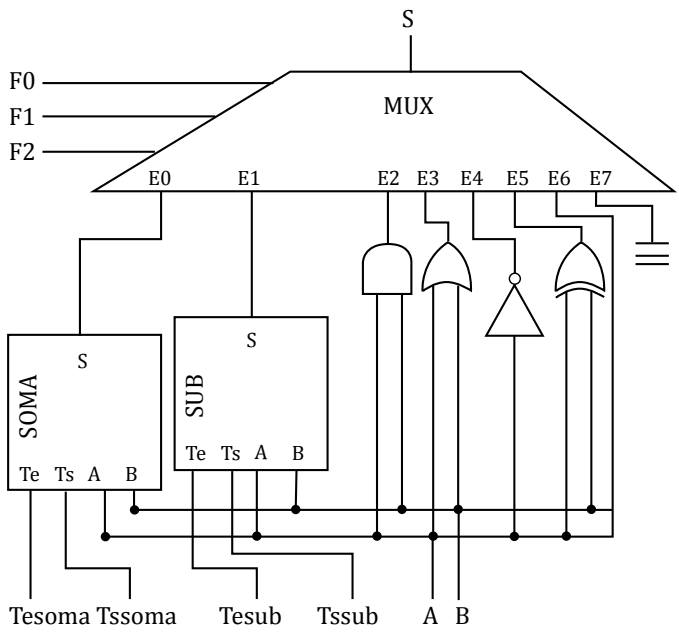
Para que fique mais claro, vamos trocar as entradas do MUX da tabela 1 pelas operações que pretendemos realizar com essas seleções, conforme expresso pela tabela 2.

Tabela 2 – Tabela de funções da nossa ULA

F2	F1	F0	S
0	0	0	Soma
0	0	1	Subtração
0	1	0	AND
0	1	1	OR
1	0	0	NOT
1	0	1	XOR
1	1	0	A
1	1	1	RESET

Dessa forma, podemos juntar todos os circuitos que já foram discutidos neste capítulo, e teremos uma representação da nossa ULA básica na forma de circuito digital, conforme expresso pela figura 1.

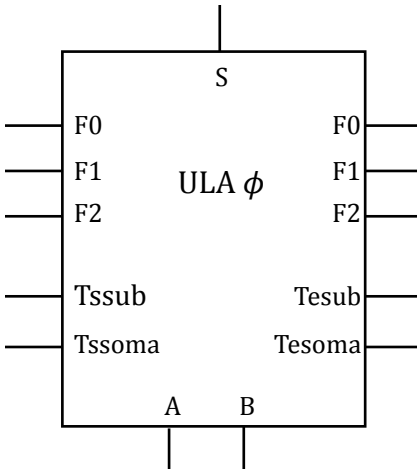
Figura 1 — Diagrama em blocos de um bit da ULA básica



No circuito da figura 1, vamos ter quatro entradas — Tesoma, Tesub, A e B — e três saídas — Tssoma, Tssub e S —, além das entradas F0, F1 e F2, que selecionam qual função da ULA está ativa. Ligamos essas entradas e saídas internamente no nosso bit da ULA, conforme descrito na figura 1.

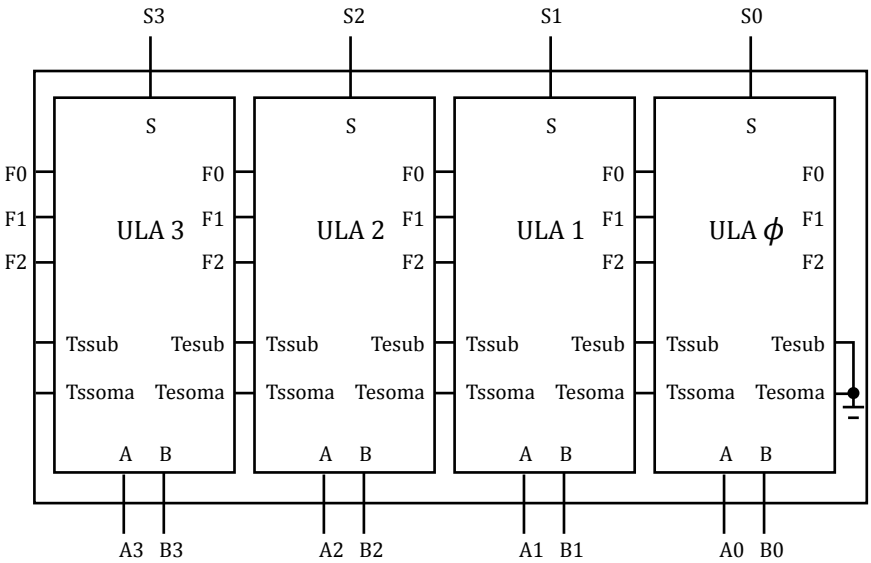
Podemos simplificar a representação de cada bit da nossa ULA com o símbolo da figura 2, a seguir.

Figura 2 — Símbolo de 1 bit da nossa ULA básica



Se juntarmos dois circuitos de 1 bit da nossa ULA, teremos uma ULA de 2 bits. Por outro lado, se juntarmos quatro circuitos, como na figura 3, a seguir, teremos uma ULA de 4 bits.

Figura 3 — ULA de 4 bits



Assim, implementamos uma ULA que pode ser ampliada, a fim de executar as operações descritas neste capítulo, para quantos bits forem necessários.

Considerações finais

A descrição que fizemos da nossa ULA básica certamente não é a forma mais eficiente de implementar esse circuito; porém, ilustra bem o que é uma ULA e como podemos implementar esse elemento, que é de fundamental importância para a implantação de circuitos digitais mais complexos. Além disso, a ULA é um dos componentes básicos dos quais precisamos para implementar computadores e quaisquer outros produtos que sejam desenvolvidos utilizando as técnicas de projeto digital, e que tenham de manipular informações de forma bastante compacta.

Referências

VAHID, Frank. **Sistemas digitais**: projeto, otimização e HDLs. Tradução: Anatólio Laschuk. Porto Alegre: Artmed, 2008.