EXPERIMENTO 3

CIRCUITOS SOMADORES E SUBTRATORES

3.1 Objetivos

Familiarização com a aritmética binária e com a implementação de circuitos somadores binários.

3.2 Introdução

Uma função essencial da maioria dos computadores e calculadoras é a realização de operações aritméticas. Essas operações são realizadas em uma parte específica do hardware conhecida como Unidade Lógica e Aritmética (ULA). Esta unidade é formada por portas lógicas e flip-flops que combinados permitem a realização de somas, subtrações, multiplicações e divisões de números binários. Esses circuitos realizam essas operações em uma velocidade considerada humanamente impossível. Normalmente, uma operação de adição demora menos que 100 ns [TWM07].

A estrutura básica de uma ULA está mostrada na Figura 3.1. O objetivo básico de uma ULA é receber dados binários armazenados na memória e executar operações aritméticas e lógicas sobre esses dados, de acordo com instruções provenientes da unidade de controle. Assim, uma sequência de operações típica de uma ULA pode ocorrer conforme se segue:

1. A unidade de controle recebe uma instrução determinando que um determinado valor na memória deve ser somado ao valor do acumulador;

2. O valor é transferido da memória para o registrador B;

3. Os valores do acumulador e do registrador B são apresentados à lógica de adição que executa a soma e armazena o resultado no acumulador;

4. O resultado pode ser mantido no acumulador para operações subsequentes ou ser transferido para a memória.

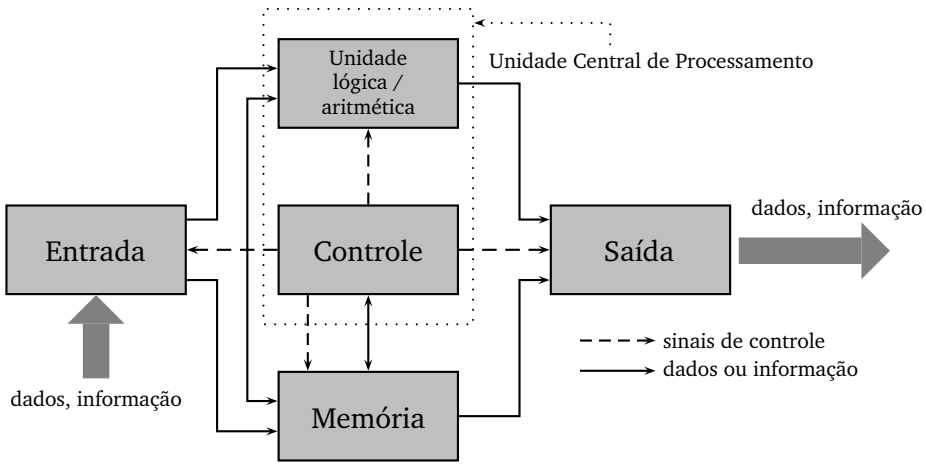

Figura 3.1: Blocos funcionais de uma ULA. Figura adaptada de [TWM07].

A complexidade dos blocos funcionais da ULA mostrados na Figura 3.1 é proporcional à complexidade do sistema em que será utilizada; assim, sistemas simples permitem o uso de ULAs simples e sistemas sofisticados exigem ULAs sofisticadas. Uma vez estabelecido o porte do sistema, existe também o compromisso entre velocidade e preço. Por exemplo, as calculadoras eletrônicas exigem ULAs que permitem operações complexas, porém com velocidade de operação baixa, reduzindo-se o custo; já os computadores de grande porte exigem velocidade de operação elevada, aumentando o custo da ULA.

Neste momento do curso, no entanto, não estamos interessados no estudo detalhado de todos os blocos funcionais que formam uma ULA. Nosso foco, então, será apenas nos circuitos lógicos que realizam as operações aritméticas de soma e subtração.

3.3 Representação de Números Binários

Como a maioria dos computadores e das calculadoras digitais realiza operações tanto com números positivos e negativos, é necessário representar, de alguma forma, o sinal do número ($+$ ou $-$). Existem várias formas de obter essa representação. Uma das maneiras é a representação denominada de sistema **sinal-magnitude**, ou sinal e amplitude. Nesta forma de representação, simplesmente adiciona-se ao número um outro bit denominado de **bit de sinal**. Em geral, a convenção comum é utilizar o bit 0 para número positivos e o bit 1 para números negativos.

Como exemplo, vamos representar os números 27_{10} e -55_{10} na base 2 usando a representação sinal-magnitude:

$$+27_{10} = \underbrace{0}_{+}\ \underbrace{1\,1\,0\,1\,0\,1}_{\text{magnitude}}$$

$$-55_{10} = \underbrace{1}_{-}\ \underbrace{1\,1\,0\,1\,0\,0\,1}_{\text{magnitude}}$$

Notem que, para o primeiro caso, foram necessários 7 bits para representar o número, e no segundo, 8 bits. A quantidade de bits necessária para representar os números em uma sistema digital (por exemplo, a ULA) em geral é fixa. Assim, nos referenciamos a uma ULA de 8 bits, 16 bits, e assim por diante.

Uma outra observação refere-se ao uso do ponto decimal: assim como na base 10, números à direita do ponto decimal representam potências negativas de 2. Assim, em sinal-magnitude o número $0101.11_2 = +5 + 2^{-1} + 2^{-2} = 5.75_{10}$.

Embora o sistema sinal-magnitude seja uma representação direta, os computadores e calculadores normalmente não o utilizam, porque esse sistema requer a implementação de circuitos mais complexos quando comparados a outras alternativas. Um exemplo de complicador é o número 0: ele pode ter duas representações. Por exemplo, usando 1 bit de sinal e 4 de magnitude há duas possibilidades: (0 0000) ou (1 0000).

A maioria dos sistemas modernos usa o **sistema de complemento de 2** para representar números negativos. O complemento de 2 de um número negativo é obtido tomando o complemento de 1 do número (substituição de todos os 0s por 1s e 1s por 0s) e somando 1 na posição do bit menos significativo.

O sistema de complemento de 2 para representação de números com sinal funciona da seguinte forma:

- Se o número for **positivo**, a magnitude é representada por na sua forma direta, e um bit 0 representando o sinal é colocado ao lado do bit mais significativo.
- Se o número for **negativo**, a magnitude é representada na sua forma de complemento de 2, e um bit 1 é colocado ao lado do bit mais significativo.

Um exemplo de representação em complemento de 2: o número -97_{10} = (11100001) em sinal-magnitude é representado em complemento de 2 como (10011111), pois tomando a magnitude (1100001) resulta

$$\text{CP}_2[1100001] = \text{CP}_1[1100001] + 1 = 0011110 + 1 = 0011111.$$

Esse sistema é o mais utilizado para representar números com sinal porque permite realizar a operação de subtração efetuando, na verdade, uma operação de adição. Desta forma, o sistema digital pode usar o mesmo circuito tanto na adição quando na subtração, desse modo poupando hardware. Observe que se realizarmos a operação de complemento de 2 duas vezes, encontramos a magnitude do número binário.

Para visualizar como esse procedimento funciona, basta lembrar que realizar a subtração de $(5 - 4)$ é equivalente a realizar a seguinte operação de adição $(5 + (-4))$. Portanto, para realizar a operação de soma ou subtração que envolva números negativos, basta determinar o complemento de 2 dos números negativos envolvidos e realizar a operação de adição. O procedimento descrito abaixo ajuda na tarefa de realizar operações no sistema de complemento de 2.

1. Represente os números envolvidos em binário puro;
2. Verifique a quantidade de bits da representação, se necessário complete a sequência de bits com zeros à esquerda de acordo com a quantidade de bits definida para a realização da operação;

3. Identifique os números negativos e determine o seu complemento de 2;

4. Realize a soma binária;

5. Verifique o bit de sinal do resultado. Se for 0, o resultado é positivo; se for 1, o resultado é negativo;

6. Em caso de resultado negativo, se quiser visualizar o resultado, recomenda-se representar o número na forma de sinal-magnitude, assim determine o complemento de 2 do resultado para determinar a magnitude do número negativo obtido (lembrando ao final deste processo que trata-se de um número negativo);

Um procedimento manual rápido para encontrar o complemento de 2 de um número binário é partindo de sua representação positiva: (i) inicie copiando os bits da direita para a esquerda, até encontrar o primeiro bit 1; (ii) copie este bit 1; e (iii) inverta todos os outros bits à esquerda, incluindo o de sinal (que estava com 0). Em outras palavras, inverta todos os bits à esquerda do 1 menos significativo. Verifique.

3.3.1 *Overflow* Aritmético

Ocorre sempre que uma operação aritmética produz um número que necessita ser expresso em mais bits de magnitude do que está disponível. Por exemplo, considere um sistema digital que trabalha com números de 4 bits de magnitude e um bit de sinal. Considere que seja necessário realizar a adição de $+9$ (01001) com $+8$ (01000). Neste caso, tem-se como resultado o número (10001), que representaria o decimal -1, enquanto que a resposta deveria ser $+17$, indicando obviamente um erro no cálculo. Isso ocorre porque para representar a magnitude 17 é necessário mais do que os quatro bits disponíveis, portanto ocorre o transbordamento do "vai-um" ou *overflow* ou transbordamento.

A condição de transbordamento pode ocorrer apenas quando dois números positivos ou dois números negativos são somados, e isso sempre produz um resultado errado. Desta forma, o transbordamento pode ser detectado verificando se o bit correspondente à casa de sinal do resultado tem o mesmo valor dos bits de sinal dos números que estão sendo somados.

Como exemplo de uma condição onde há transbordamento, vamos realizar a operação -18_{10} - 22_{10}, usando 6 bits, incluindo o de sinal. Notem que os 5 bits permitem a representação máxima de -31. Já expressando dos números negativos em complemento de 2, vem:

0	1	1	1			"vai-um"
1	0	1	1	1	0	(-18)
1	0	1	0	1	0	(-22)+
0	1	1	0	0	0	(+24)

3.3.2 Circuitos Somadores e Subtratores

Existem diversos circuitos diferentes para implementar a operação aritmética de soma. Faremos aqui uma bastante usada, a do somador completo. Para realizar uma soma de dois números, vamos imaginar a soma ocorrendo da direita para a esquerda na i-ésima posição, com a eventual presença do "vai-um". Para computar o resultado na posição i, precisamos dos bits dos dois números: A_i e B_i; e também do eventual "vem-um", C_i. São, portanto, três entradas. O resultado em si são dois números: o resultado da soma naquela posição S_i e um eventual "vai-um" C_{i+1} a ser propagado para cada à esquerda. O bit mais à direita, chamado de nenos significativo (LSB - *least significant bit*) não tem "vem um". Como já foi observado, no caso de soma de dois números de mesmo sinal, a presença do "vai-um" na soma mais à esquerda (MSB - *most significant bit*) é um indicador de *overflow*.

	1	1				1		"vai-um"
0	0	1	1	0	0	0	1	(25)
0	0	0	1	0	0	0	1	(09)+
0	1	0	0	0	0	1	0	(34)
+	MSB						LSB	

Consideremos então a seguinte tabela:

A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabela 3.1: Tabela-verdade do somador completo.

A saída S_i nada mais é do que um XOR das três entradas (verifique):

$$S_i = A_i \oplus B_i \oplus C_i.$$

O "vai-um" aparece quando pelo menos duas das três entradas tem valor 1. Portanto,

$$C_{i+1} = A_i\, B_i + A_i\, C_i + B_i\, C_i.$$

O circuito pode então ser implementado com portas lógicas na forma mostrada na Figura 3.2. Na mesma figura, encontra-se um diagrama de blocos do circuito apresentando apenas as entradas e saídas.

O circuito apresentado é capaz de somar corretamente apenas um bit. Para realizar uma adição de N bits, basta concatenar N blocos, como mostrado na Figura 3.3 para N=4.

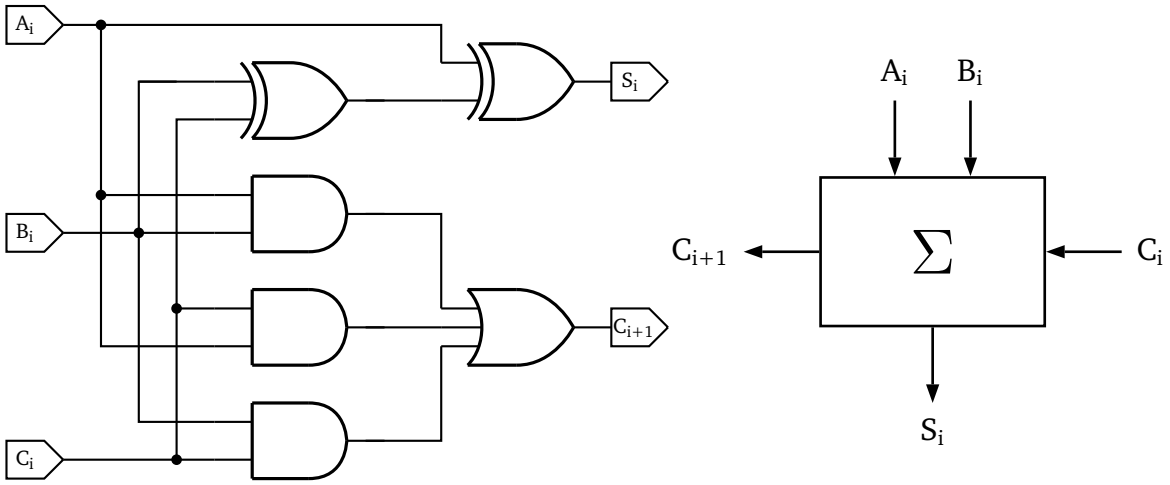

Figura 3.2: Circuito somador completo: implementação com portas lógicas e representação entrada-saída.

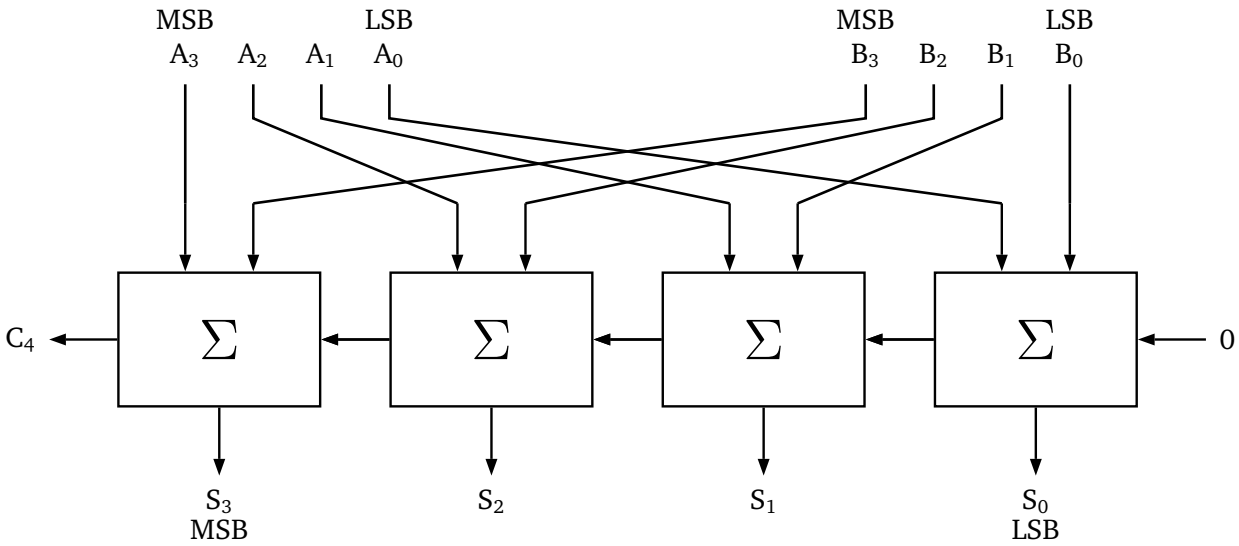

Figura 3.3: Circuito somador para entradas de 4 bits usando somadores completos.

3.4 Pré-Relatório

3.4.1 Pesquisa bibliográfica

Realize uma pesquisa bibliográfica sobre as diferentes configurações de circuitos somadores existentes. Faça uma breve explicação sobre a teoria envolvida em cada um deles, além de uma comparação entre essas diferentes configurações, destacando em cada caso as vantagens e desvantagens. DICA: Pesquisar sobre um circuito denominado **"*carry* antecipado"** (*look-ahead carry*).

3.4.2 Projetos e Simulações

Nesta seção são descritos os circuitos que devem ser projetados e simulados. Na etapa de simulação o aluno pode utilizar o software de sua preferência, desde que forneça as formas de de onda às entradas e saída. Sugerimos utilizar softwares com funcionalidade similar à encontrada na sala SS.

Em seu pré-relatório, devem ser apresentados:

- O nome do software utilizado (fabricante, versão, sistema operacional);
- Os diagramas de blocos e esquemáticos dos circuitos projetados;
- Um plano de validação contendo quais as saídas esperadas para as entradas escolhidas;
- O código VHDL de cada circuito projetado.

Nos projetos os alunos devem apresentar todas as informações adicionais que forem necessárias para perfeita compreensão do projeto realizado.

3.4.2.1 Projeto e Simulação 1

Projetar e simular um circuito que permita realizar o complemento de 1 de um número de 3 bits (incluindo o bit de sinal). Esse circuito deve possuir ainda uma entrada seletora (SEL) que permita especificar quando se deve realizar essa operação de complemento.

Desta forma, quando a operação de complemento não for desejada, o circuito deve fornecer na saída exatamente o número de entrada (ver Figura 3.4). Assim, a função realizada por esse circuito depende do valor da entrada de seleção SEL:

- se SEL = 0, a função selecionada é a IGUALDADE e Z = A;
- se SEL = 1, a função selecionada é o COMPLEMENTO DE 1 e Z = $CP_1[A]$.

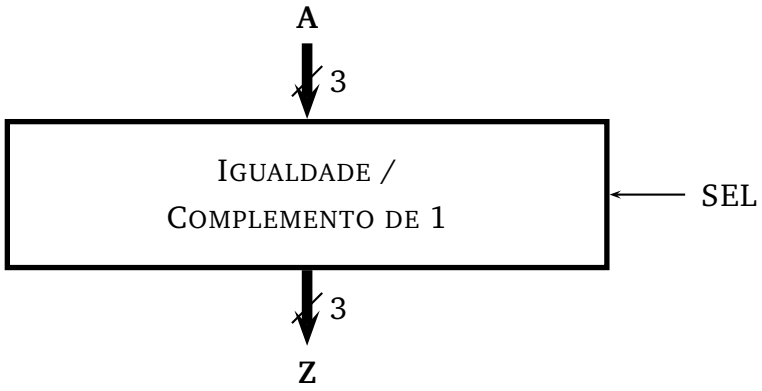

Figura 3.4: Circuito seletor de Igualdade/Complemento de 1.

3.4.2.2 Projeto e Simulação 2

Projetar e simular um circuito que seja capaz de detectar uma condição de *overflow* (estouro de capacidade) para ser usado com um circuito somador de números com sinal, de três bits, codificados na forma de complemento de 2.

3.4.2.3 Projeto e Simulação 3

Projetar e simular um circuito SOMADOR/SUBTRATOR de três bits (incluindo o bit de sinal) para números com sinal, codificados na forma de complemento de 2, dado pelo bloco funcional mostrado na Figura 3.5. O circuito deve ter ainda uma saída (E) que indica as situações em que ocorreu um estouro de capacidade (*overflow*). Desta forma, a função realizada por este circuito, que depende do valor da entrada de seleção SEL, pode ser resumida da seguinte forma:

- se SEL = 0, a função selecionada é a SOMA e S = A + B;
- se SEL = 1, a função selecionada é a SUBTRAÇÃO e S = A + CP_2[B].
- A saída E indica a condição de Overflow.
- A saída C_3 é o *carry* da operação, que pode ser utilizado para expandir a capacidade da operação através da associação com outros blocos somadores.

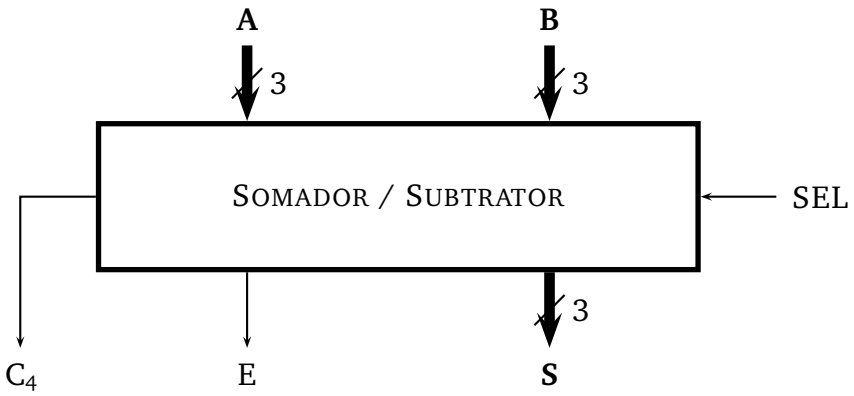

Figura 3.5: Circuito Somador/Subtrator de três bits.

DICAS:

- O bloco lógico funcional desse circuito deve ser, primeiramente, detalhado na forma de um circuito lógico utilizando grandes blocos funcionais. Em seguida, esse circuito lógico deve ser detalhado usando os blocos funcionais disponíveis individualmente na forma de portas lógicas.
- Utilize os circuitos projetados nas Seções 3.4.2.1 e 3.4.2.2 como parte do seu projeto.

- Cuidado na etapa de verificação do circuito. Prepare tabelas contendo os valores de entrada e saída e siga-as, verificando se o circuito fornece a saída correta em todos os casos.

OPCIONAL: O circuito projetado irá fornecer na saída os valores negativos em sua forma de complemento de 2. Altere o seu projeto de forma que na saída os números negativos não estejam nesta forma, mas representados pela notação sinal-amplitude.

3.5 Roteiro Experimental

Implemente em ambiente de simulação os circuitos projetados nas Seções 3.4.2.1, 3.4.2.2 e 3.4.2.3 seguindo as orientações do professor em sala. Em todos os projetos deve-se realizar a seguinte sequência:

1. Apresente os diagramas de blocos de seu sistema completo, em termos de entradas e saídas.
2. Se possível, refine o detalhamento do diagrama para conter sub-blocos funcionais, também representados em termos de entradas e saídas.
3. Repita o processo anterior até chegar ao nível de descrição em termos de portas lógicas.
4. Implemente o código VHDL do circuito todo.
5. Apresente formas de onda que validem o funcionamento de seu projeto.
6. Observe, também, a ocorrência de resultados inválidos. Caso existam, em que condições eles ocorrem?