Henrique Coutinho Layber e Renan Moreira Gomes

IMPLEMENTAÇÃO DE UMA UNIDADE LÓGICA E ARITMÉTICA COM PORTAS LÓGICAS BÁSICAS

Professor Dr. João Paulo de Almeida

Universidade Federal do Espírito Santo

Ciência da Computação

Vitória, Espírito Santo, 2018

Henrique Coutinho Layber e Renan Moreira Gomes
Universidade Federal do Espírito Santo
Vitória 2018

Sumário

Conceituando a ALU

Unidade Lógica Aritmética (*Arithmetic Logic Unit – ALU*)é uma unidade lógica aritmética é um elemento central em uma CPU (Unidade de Processamento Central). Ela executa operações lógicas e aritméticas básicas diversas em dados de entrada, mas podemos juntá-las e usar das basicas para formar ALUs mais complexas. Iremos explicar passo a passo como projetar uma ALU que opere sobre dados de 8 bits, e como usar a mesma para montar uma ALU de 16 bits, como uma calculadora básica.

Especificação

Nossa ALU final irá conter as seguintes operações em cima das entradas A e B de 16 bits:

- `ADD` soma (`A + B`);
- `SUB` subtração (`A – B`);
- `XOR` bit-a-bit (`A XOR B`);
- `OR` bit-a-bit (`A OR B`);
- `AND` bit-a-bit (`A AND B`);
- `SHL` Shift left (`A << 1` desloca os bits de A para a esquerda);
- `SHR` Shift right (`A >> 1` desloca os bits de A para a direita).

Henrique Coutinho Layber e Renan Moreira Gomes
Universidade Federal do Espírito Santo
Vitória 2018

Além dos operandos A e B de 8 bits, nossa ALU irá conter uma entrada OP de 3 bits que indica a operação a ser realizada com os operandos. A ALU irá seguir a seguinte tabela para a operação OP:

A ALU inicial também terá um `carry` de entrada (Cin), que permitirá depois concatenar duas ALUs de 8 bits para formar uma ALU de 16 bits, e as saídas de ambas deverão funcionar da seguinte forma:

Valor de OP	Saída
000	ADD Soma (A + B)
001	SUB subtracão (A - B)
010	XOR bit-a-bit (A XOR B)
011	OR bit-a-bit (A OR B)
100	AND bit-a-bit (A AND B)
101	SHL Shift left (A << 1)
110	SHR Shift right (A >> 1)
111	Não usada

Tabela a: Valor de OP e respectiva operações

- Uma saída C indicando `carry` (1 em C indica que houve `carry`) (no caso de subtração, indica `borrow`, no caso da operação `SHL`, o carry recebe o bit mais significativo sendo deslocado; no caso de `SHR`, recebe o bit menos significativo, em soma, indica Overflow);
- Uma saída Z que indica se todos os bits da saída são iguais a `0` (`1` em Z indica que todos os bits da saida S são `0`);
- Uma saída S de 8 bits com o resultado numérico da operação, e dois dígitos de display de 7 segmentos para mostrar o resultado.[1]

1 Não será feito display hexadecimal para a ALU de 16 bits, pois a sua implementação é mais complexa.

Observe na imagem abaixo, todas as entradas e saídas que serão utilizadas como base para a definição da ALU:

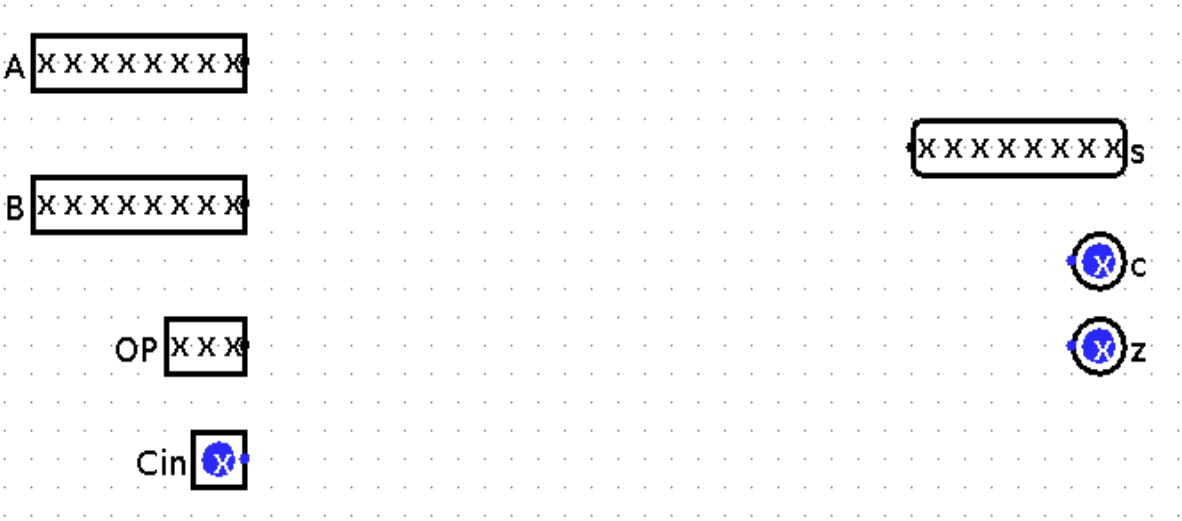

Illustration a: Entradas e saídas da ALU

Cada circuito da ALU

É muito mais simples construir um circuito para 8 bits do que 12 deles de uma só vez. É por isso que buscamos modularizar o máximo possível – e para a resolução de problemas (bugs). Somente alguém insano faria tudo de uma só vez. É claro que poderíamos poupar algumas comparações (e consequentemente performance ou dinheiro) por fazer sem a modularização, mas como o objetivo da nossa ULA é fazer apenas uma das operações de cada vez, não haverá tanto impacto.

Comparação OR bit-a-bit (A OR B):

A comparação OR é um circuito de relativamente simples implementação, principalmente por ele ser uma operação bit-a-bit. Então é um simples caso de aplicar a operação a cada bit separadamente para expandir ele para uma operação de 8 bits e então juntar em uma saída de 8 bits:

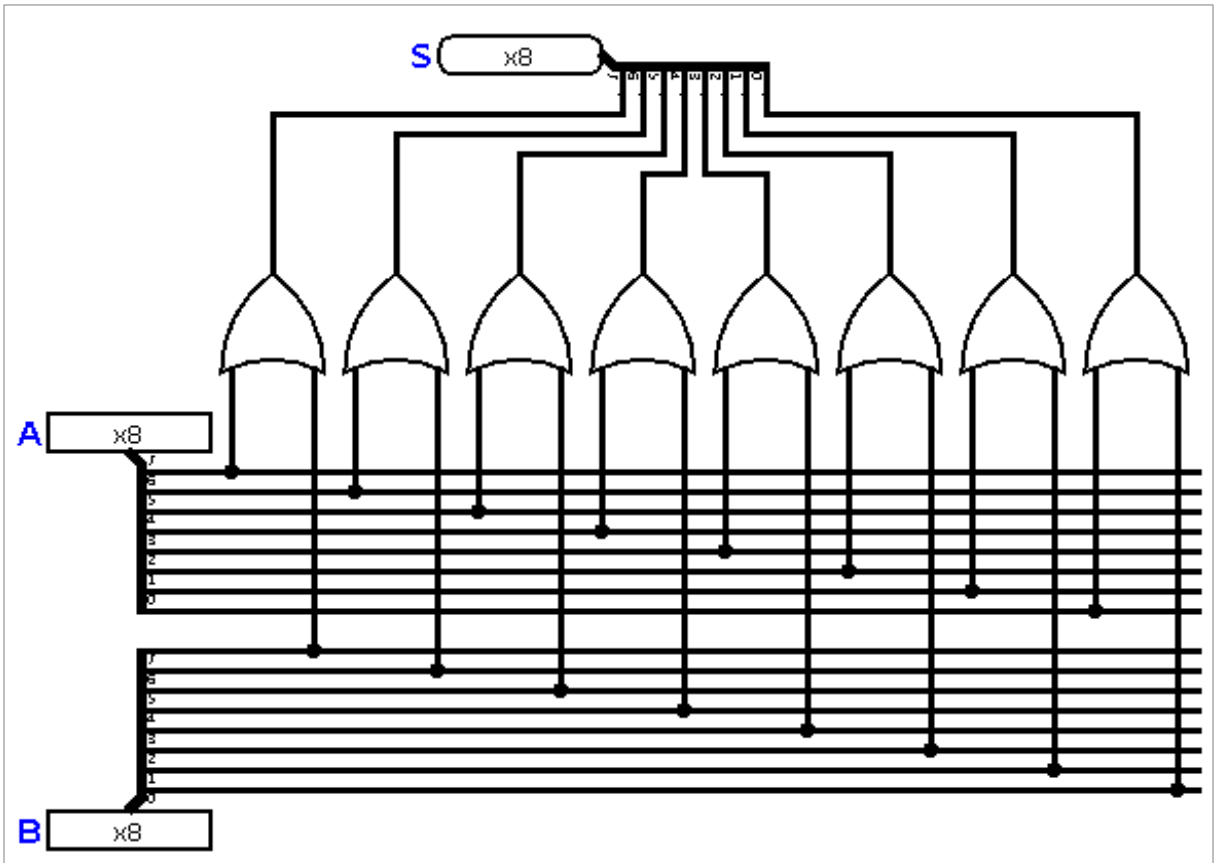

Illustration b: OR 8bits

Henrique Coutinho Layber e Renan Moreira Gomes
Universidade Federal do Espírito Santo

Comparação AND bit-a-bit (A AND B)

Tal como a comparação OR, a comparação AND também é aplicada bit-a-bit, então podemos aplicar o mesmo método para essa operação e o resultado devera ser A AND B (8bits):

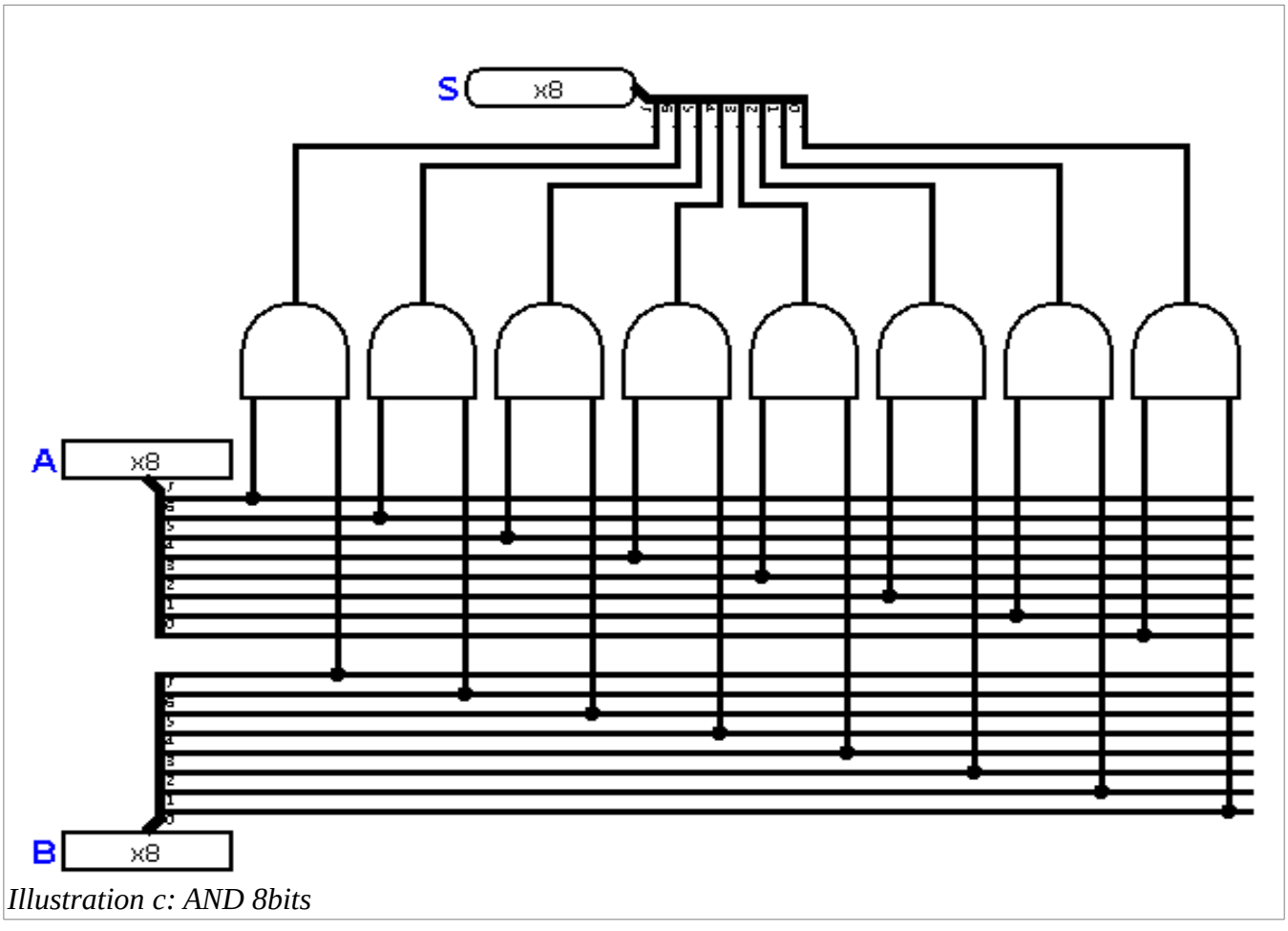

Illustration c: AND 8bits

Henrique Coutinho Layber e Renan Moreira Gomes
Universidade Federal do Espírito Santo
Vitória 2018

Comparação XOR bit-a-bit (A XOR B)

Como a comparação XOR (diferente de) não é uma porta básica (sendo elas a AND, OR e NOT) então é necessário defini-la e então modularizar e usar para fazer o XOR para 8 bits. Apór fazer o XOR de 1 bit, pode-se considerar como uma porta básica e então ela também é aplicada bit-a-bit, levando usar a mesma técnica:

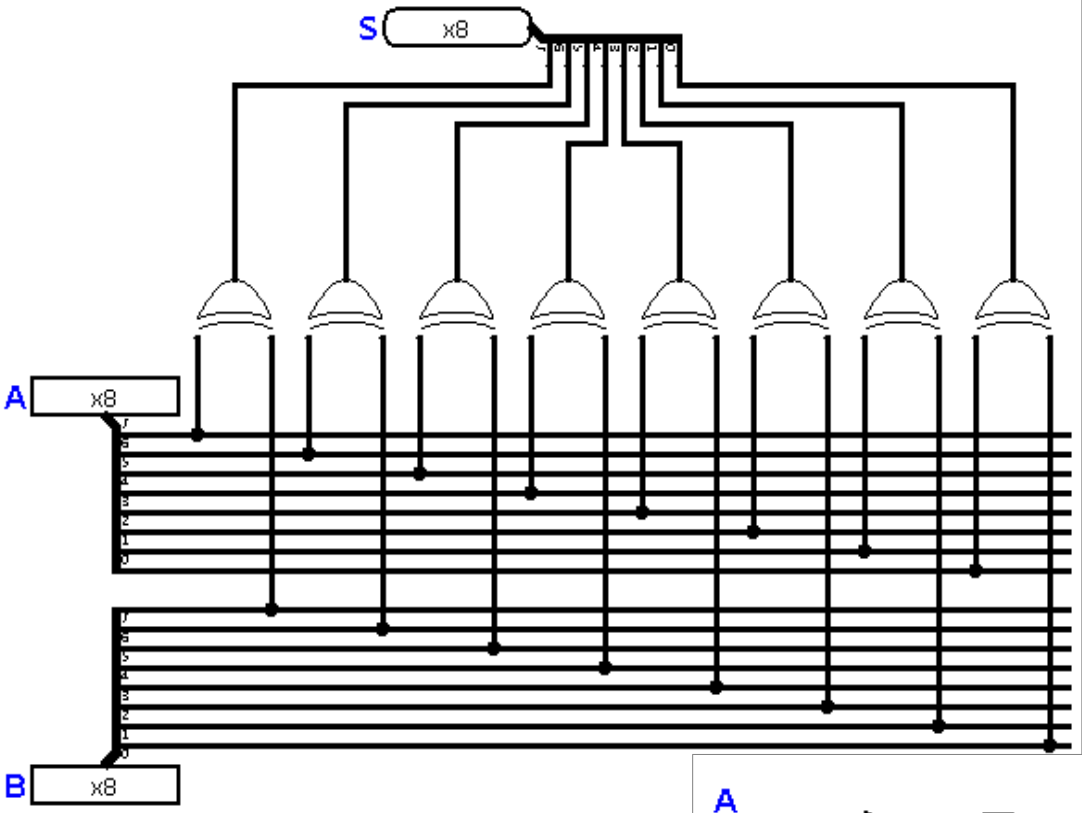

Illustration d: XOR 8bits

Cada sinal "D" é um XOR de 1 bit, representado na illutration e.

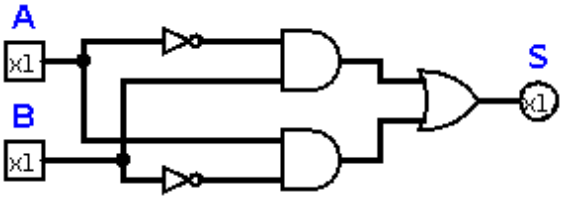

Illustration e: XOR 1 bit

Note que não é preciso ser complexo para a modularização ser útil.

ADD soma (A + B)

Para realizar a soma de 8 bits, devemos criar inicialmente um somador completo de 1 bit, observe a imagem abaixo:

Com esse circuito de soma (somador completo) somos capazes de criar um somador completo de 8 bits, ou seja, podemos utilizar um módulo de circuito de soma de 1 bit para construir um somador de 8 bits. Basta utilizarmos o Cout do primeiro no Cin do segundo, assim sucessivamente até chegar ao somador de 8 bits. Observe a imagem abaixo:

E assim, finalizamos a operação ADD da nossa ULA.

Henrique Coutinho Layber e Renan Moreira Gomes
Universidade Federal do Espírito Santo
Vitória 2018

SUB subtração (A – B)

Nós faremos a subtração de 8 bits semelhante ao somador de 8 bits, porém com algumas diferenças. O Cin será constante 1 e todas as entradas de B serão negadas, pois o B será representado em complemento a dois. Assim realizamos a soma de A com B em complemento a dois, ou seja, subtraimos A de B. Observe a imagem a seguir:

E assim, finalizamos a operação SUB da nossa ULA.

XOR bit a bit (A XOR B)

A operação XOR, o “ou exclusivo” pode ser implementado de diversas maneiras. Observe a imagem a seguir de XOR de 1 bit utilizando apenas portas lógicas básicas:

Basta usar 8 XORs de 1 bit para representar um XOR de 8 bits (bit a bit). Observe:

E assim finalizamos nosso XOR de 8 bits.

OR bit a bit (A OR B)

AND bit a bit (A AND B)

SHL Shift left (A << 1)

SHR Shift right (A >> 1)