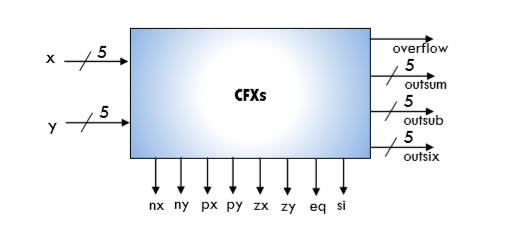


CURSO de Engenharia Informática



Lógica computacional 2017/2018

Elaborado por: Daniel Barreiro nº: 170221079 **Turma:**

Circuito cfxs

Índice

Conteúdo

[Introdução 2](#_Toc499063081)

[Soma 3](#_Toc499063082)

[Half Adder 3](#_Toc499063083)

[Full Adder 3](#_Toc499063084)

[Subtração 4](#_Toc499063085)

[Simétrico 5](#_Toc499063086)

[Função NX/NY 6](#_Toc499063087)

[Função PX/PY 6](#_Toc499063088)

[Função ZX/ZY 7](#_Toc499063089)

[Função EQ 7](#_Toc499063090)

[Função SI 8](#_Toc499063091)

Introdução

Este relatório foi elaborado no âmbito da disciplina de Lógica Computacional com o objetivo de aplicar os conhecimentos adquiridos nas aulas teórico práticas e laboratórios. O circuito a elaborar é destinado a realizar diversas funções sobre números em complementos para dois. Neste relatório iremos explicar a construção de cada função do circuito e das características do mesmo.

Soma

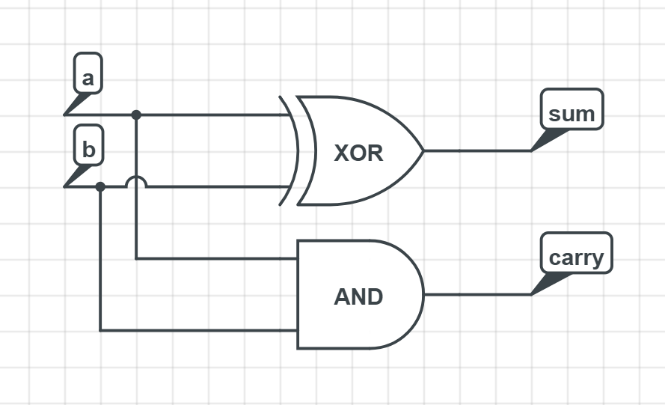
A implementação da soma foi feita através da utilização de um **Half Adder** e **Full Adder**.

Half Adder

O circuito Half Adder tem este nome porque não leva em consideração os resultados de somas menos significativas, ou seja, apena soma 2 bits e não tem o transporte.

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | sum | Carry |
| 0 | 0 | **0** | **0** |
| 0 | 1 | **1** | **0** |
| 1 | 0 | **1** | **0** |
| 1 | 1 | **0** | **1** |

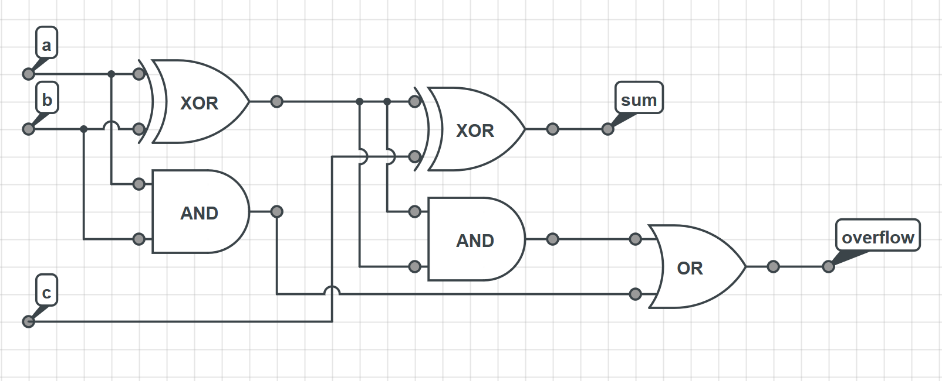
Ao examinar a tabela de verdade podemos ver que a coluna sum poderá ser representada por uma porta **XOR** e a coluna **carry** por uma tabela **AND**, dando assim o seguinte circuito:



Full Adder

O circuito Full Adder soma dois números e a completar o Half Adder tem em conta o valor resultante da soma anterior (‘vai um’). O circuito é composto por 2 Half Adders e uma porta OR que recebe os output do transporte dos dois Half Adder.

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | XNOR | Carry |
| 0 | 0 | **1** |  |
| 0 | 1 | 0 |  |
| 1 | 0 | 0 |  |
| 1 | 1 | **1** |  |



Subtração

A implementação da subtração envolve a utilização 2 dois “sub circuitos”, **Half Subtractor** e Full Subtractor.

Half Subtractor

Tal como o Half Adder o Half Subtractor não simula o transporte, a tabela de verdade é a seguinte:

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | suB | BORROW |
| 0 | 0 | **0** | **0** |
| 0 | 1 | **1** | **1** |
| 1 | 0 | **1** | **0** |
| 1 | 1 | **0** | **0** |

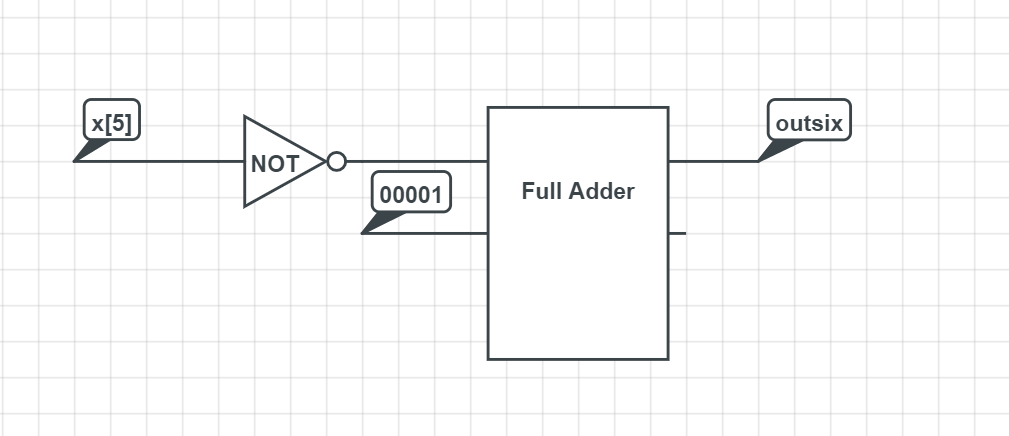
Ao examinarmos a tabela de verdade do Half Subtractor podemos ver que a coluna SUB representa uma porta XOR e a coluna BORROW pode ser representada por um

Simétrico

O output outsix é designado para devolver o simétrico do input x. Para calcular o simétrico de um número binário basta negar todos os seus bits e somar 1 ao número.

**00101 (5) 11010+00001 11011 (-5)**

Para o circuito lógico utilizámos portas **NOT** para negar todo o bit do input x e de seguida utilizando um **FullAdder** somamos 1 ao número binário e assim obtemos o simétrico.



Função NX/NY

A função nx/ny retorna 1 se o input for negativo, para conseguirmos este retorna e uma vez que estamos a utilizar complemento de dois para representação basta nos **retornar o último bit** do número recebido, caso o último bit seja 0 o número é positivo senão é positivo.

|  |  |
| --- | --- |
| NÚmero DECIMAL | NÚmero Binário |
| -8 | **1**000 |
| -7 | **1**001 |
| -6 | **1**010 |
| -5 | **1**011 |
| -4 | **1**100 |
| -3 | **1**101 |
| -2 | **1**110 |
| -1 | **1**111 |
| 0 | **0**000 |
| 1 | **0**001 |
| 2 | **0**010 |
| 3 | **0**011 |
| 4 | **0**100 |
| 5 | **0**101 |
| 6 | **0**110 |
| 7 | **0**111 |

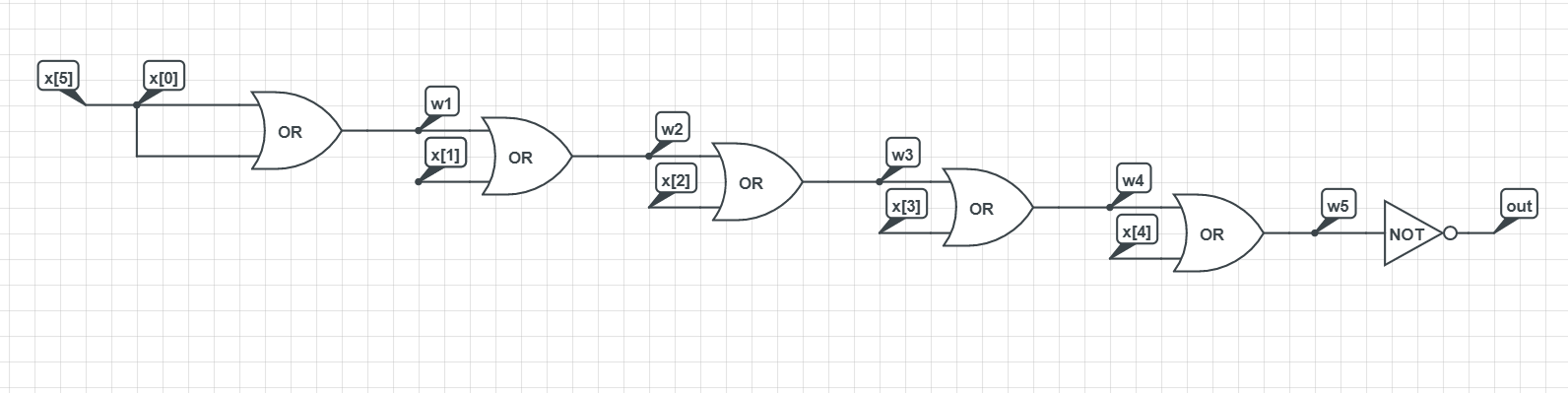
Função PX/PY

A função px/py retorna 1 se o número recebido fosse par. A paridade de um número binário é definida pelo seu **último bit**, se o último bit seja 0 o número é par caso contrário o número é impar.

|  |  |
| --- | --- |
| NÚmero DECIMAL | NÚmero Binário |
| 0 | 00**0** |
| 1 | 00**1** |
| 2 | 01**0** |
| 3 | 01**1** |
| 4 | 10**0** |
| 5 | 10**1** |
| 6 | 11**0** |
| 7 | 11**1** |

Função ZX/ZY

A função zx/zy retorna 1 se o input for zero, para um número binário ser zero todos os seus bits terão de ser 0, verificamos esta condição utilizando um conjunto de portas OR em cascada e uma porta NOT no final, caso algum dos bits seja 1 o output final será 0, caso todo os bits sejam zero o output é 0.



Função EQ

A função eq retorna 1 se o input x e y forem iguais, para verificar esta condição utilizamos a porta XNOR entre pares de bits e de seguida portas AND para valida se existe algum bit diferente entre os dois números. Uma vez que a porta XOR retorna 1 sempre que existe uma diferença entre 2 bits, a porta XNOR (negação da porta XOR) retorna 1 sempre que existe uma igualdade entre 2 bits.

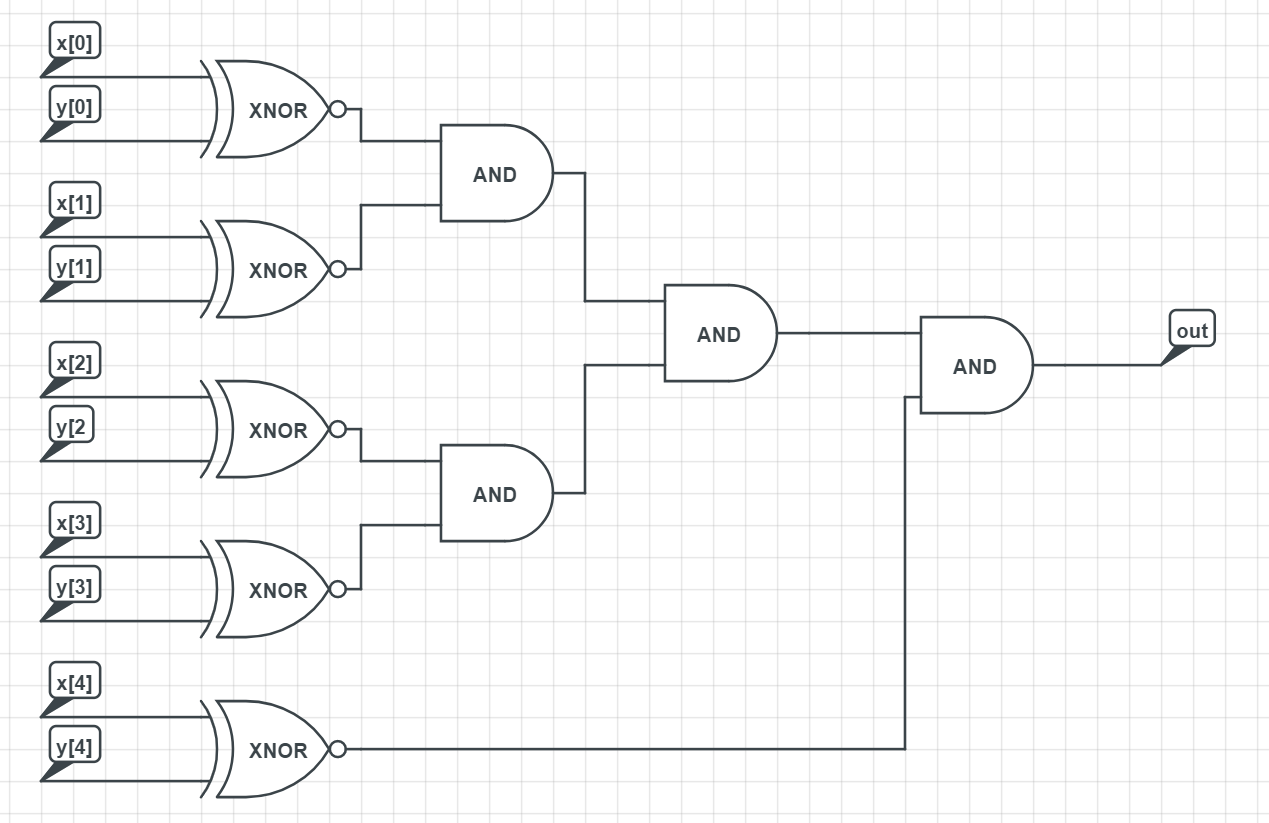
Tabela XOR

|  |  |  |
| --- | --- | --- |
| x | y | XOR |
| 0 | 0 | 0 |
| 0 | 1 | **1** |
| 1 | 0 | **1** |
| 1 | 1 | 0 |

Tabela XNOR

|  |  |  |
| --- | --- | --- |
| x | y | XNOR |
| 0 | 0 | **1** |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | **1** |

A primeira parte do circuito consiste em 5 portas XNOR entre os bits do input x e o input y, de seguidas o output dessas portas XNOR são avaliadas por portas AND que iram avaliar se de facto todos os pares de bit são iguais.



Função SI

A função si retorna 1 se o input x e y forem simétrico. Dois números são simétricos e a soma dos mesmos for 0. O circuito lógico desta função envolve um **FullAdder** para somar os dois inputs e a **função zx/zy** para validarmos se o resultado é 0.

