ARITMÉTICA DE NÚMEROS BINARIOS E SUAS RELAÇÕES COM CIRCUITOS LÓGICOS.

Rafael Pinheiro Leite[1], Prof. Dr. Matheus da Silva Menezes [2]

Resumo: Esse estudo consiste em compreender a aritmética de números binários através da álgebra booleana e sua relação com as portas lógicas. Ao explanar um pouco sobre números binários e as suas operações aritméticas, para que possamos visualizar sua ligação com a álgebra booleana, bem como a relação com as portas lógicas básicas (*OR, AND, NOT*) e suas portas conjuntas (*NOR, NAND*). Através desse embasamento, analisamos de forma analítica dois circuitos compostos, dividindo em pequenos circuitos básicos gerando uma tabela verdade para cada circuito.
Palavras-chave: números binários, álgebra booleana, circuitos lógicos, portas lógicas.

1.INTRODUÇÃO

O surgimento dos números aconteceu pela necessidade dos homens pré-históricos de contar, quando iam caçar, levavam ossos ou pedaços de madeira com a finalidade de fazer um risco para cada animal que caçavam. No momento em que o homem começou a deixar de ser nômade e firmou-se em certos locais, iniciou a criação de animais em pastos, precisando assim de uma nova forma de contagem capaz de controlar seu rebanho. Para isso, o homem pré-histórico começou a utilizar pedras, para cada animal que ia pastar era colocada uma pedra num saco. No fim do dia, cada animal que voltava era retirado uma pedra, assim tornou-se mais fácil saber se um animal tinha se perdido ou morrido. Essas foram as formas mais antigas encontradas para se contar, ao formarem civilizações, cada uma utilizou uma forma diferente [1].

Com o passar do tempo veio a necessidade de uma forma de representar os números simbolicamente, e posteriormente de unificá-los por todo o mundo. O nosso sistema de numeração mais utilizado atualmente usa os algorítmicos indo-arábicos, sendo composto por 10 dígitos, chamado de sistema decimal. Porém, este não é o único sistema utilizado, ainda existem os sistemas com 2, 8 e 16 dígitos, os quais são chamados de sistema binário, octal e hexadecimal, respectivamente. A quantidade de algorítmicos determina qual a sua base. No sistema decimal, utilizamos os dígitos de 0 a 9; enquanto o sistema binário utiliza apenas os dígitos 0 e 1; o octal usa os dígitos de 0 a 7; em contrapartida o sistema hexadecimal utiliza os dígitos de 0 a 9 adicionando cinco letras do alfabeto maiúsculas: A, B, C, D, E e F, que correspondem, nessa base, aos números 10, 11, 12, 13, 14 e 15. [2]

A contagem em outras bases é realizada de forma semelhante a base decimal, iniciando do zero e acrescentando números, até quando não existir outro algorítmico possível. Então, começa a colocar o algarismo significante a esquerda e zera-se os dígitos à direita. Assim, temos a representação das contagens das bases:

- Decimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...
- Binaria: 0, 1, 10, 11, 100, 101, 111, 1000, 1001, ...
- Octal: 0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, ...
- Hexadecimal: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, ...

Com o advento do surgimento dos computadores, a base binaria tornou-se de extrema importância, pois é o sistema utilizado na linguagem digital, correspondendo também a vários caracteres, números, palavras, imagens e sons, além de ser um sistema mais rápido e ter maior facilidade de leitura.

O computador faz o uso do sistema binário por ser composto de circuitos, podemos interpretar que quando não existe passagem de energia no circuito, temos que o número é 0, por outro lado, entendemos há passagem de energia com o número 1.

A interpretação pelo computador dos circuitos é feita utilizando a álgebra booleana, estruturando de uma forma algébrica tornando-se mais fácil captar características básicas das portas logicas e seus conjuntos, além de possibilitar estrutura para as afirmações recebidas do circuito.

A informação recebida pelo computador é armazenada em um bit (binary digit), equivalente a menor unidade de informação, armazenando apenas um valor, 0 ou 1. O computador, mesmo podendo manipular vários bits simultaneamente, geralmente é programado para armazenar as informações em um conjunto de oito bits, chamado de bytes.

Por esse motivo, no presente trabalho abordaremos a relação binaria com circuitos lógicos para realizar calculo aritméticos simples.

2. FUNDAMENTAÇÃO TEORICA

Esta seção apresenta os conceitos necessários para uma melhor compreensão do tema abordado. A começar por uma apresentação dos números binários, suas operações e conceitos básicos de álgebra booleana e circuitos lógicos.

2.1. Números binários

Mesmo após anos de existência do sistema decimal, muitas pessoas tinham dificuldade em fazer operações aritméticas com esse sistema. Por isso, um novo sistema de numeração foi criado. Primeiramente os nativos da Mangareva, uma pequena ilha situada na Polinésia. O novo sistema facilitou as atividades mais comuns como a de conta peixes, frutas, polvos e outros bens. Posteriormente, graças a esse sistema tornou-se possível grandes avanços tecnológicos e da informática [7].

Como aponta [3], por existir apenas os dígitos 0 e 1, a contagem no sistema binário se torna mais intuitiva, seguindo da seguinte forma: 0, 1, 10, 11, 100, 101, 111, 1000, 1001, e assim sucessivamente.

Para podermos converter um número binário para decimal, temos que escrever o número como o somatório de cada algarismo (a_n), multiplicado por 2^n, onde *n* é a posição do último algoritmo, tendo a seguinte expressão:

$$N = a_n 2^n + a_{n-1} 2^{n-1} + a_{n-2} 2^{n-2} + \ldots + a_2 2^2 + a_1 2^1 + a_0 2^0 \quad (1)$$

A representação de um número *N* em uma base qualquer é dada por $(N)_b$, onde *b* representa a base de representação. Tomando como exemplo o número $(111001)_2$, sua conversão para decimal fica:

$$N = 1.\,2^5 + 1.\,2^4 + 1.\,2^3 + 0.\,2^2 + 0.\,2^1 + 1.\,2^0 = (57)_{10}$$

Para fazermos a conversão inversa, do sistema decimal para binário, dividimos o número por 2, visto que é a base, guardando o resto que sempre será 0 ou 1. Em seguida dividiremos o quociente também por dois e ficamos repetindo o processo até que quociente seja um, então, o número binário será a junção do último resto até o primeiro. Utilizando o exemplo anterior, temos a seguinte conversão exposta na Figura 1: [3]

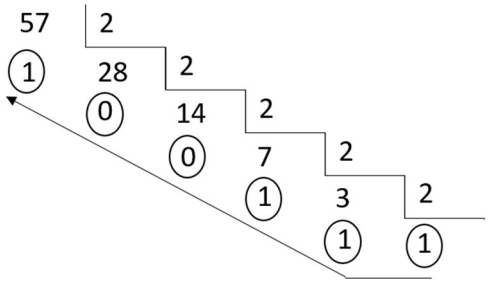


Figura 1. Conversão de decimal para binário. (Autoria própria)

2.1.1. Aritmética Binária

Tomando como base [4], para a aritmética binária a soma entre números binários segue a mesma regra de soma entre números decimais, 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1 porém, ao somarmos 1 + 1, o qual a soma seria 2 em decimal, é representado como 10 em binário. Logo a soma resulta em 0 levando o digito 1 para a próxima coluna de soma, que será levado em consideração na próxima operação de soma. As setas indicam quando um número passa para a próxima coluna, também conhecido em linguagem coloquial pela expressão "vai um", como podemos ver na Figura 2.

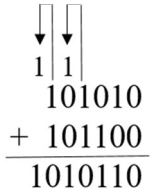


Figura 2. Operação de soma em binário. (Autoria própria)

Do mesmo modo, a subtração também utiliza a regra do sistema decimal, diferenciando apenas quando existe a operação 0 – 1, a qual o resultado é 1, por retirar um digito da próxima coluna. Caso essa retirada seja na última coluna, o número será negativo, a depender de sua estrutura de representação numérica. Um exemplo dessa operação é apresentado na Figura 3.

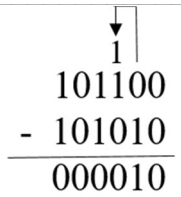


Figura 3. Operação de subtração em binário. (Autoria própria)

A multiplicação binária segue a interpretação dada pela decimal, onde qualquer número multiplicado por 0 é 0, e 1 multiplicado por ele mesmo é 1. Segue também o método de deslocamento e de adição para que assim possa encontrar o produto. Isso pode ser visualizado no exemplo apresentado na Figura 4.

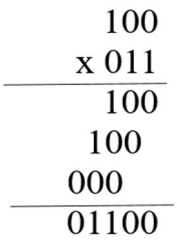


Figura 4. Operação de multiplicação em binário. (Autoria própria)

A divisão é realizada da mesma maneira que no sistema decimal, inclusive os deslocamentos na divisão decimal, porém de uma forma mais simplificada, ao existirem apenas dois valores para o quociente, 0 para quando o divisor for menor que o dividendo e 1 no caso ao contrário, como é exposto na Figura 5.

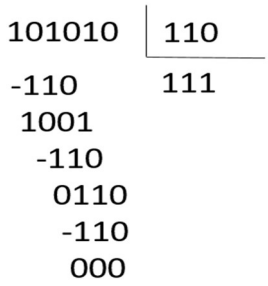


Figure 5. Operação de divisão em binário. (Autoria própria)

2.2. Álgebra booleana

Está seção usaremos como base [5]. A álgebra booleana começou a ser estudada em 1854 por George Boole, o qual ficou conhecida por seu sobrenome. Esse estudo teve grande relevância para a computação atual. A álgebra de Boole preocupa-se em expressar as operações de um circuito na forma de uma operação algébrica. Para isso, utiliza apenas dois valores, 1 ou 0, que também podem ser utilizados como verdadeiro ou falso, ligado ou desligado, respectivamente.

Pelo fato de existirem valores finitos, assim como as condições possíveis na álgebra Booleana, torna-se possível prever as prováveis saídas, dependendo das variáveis utilizadas na entrada (0 ou 1) perante a análise das condições ou características da operação desejada, construindo assim, uma tabela verdade onde são expressos esses valores. Sendo *A* e *B* entradas e *X* a saída, como mostrado na Tabela 1.

Tabela 1. Representação de tabela verdade.

A	B	X
0	0	?
0	1	?
1	0	?
1	1	?

Fonte: Autoria própria (2020)

A tabela anterior demonstra entrada em dois conjuntos lógicos (*A* e *B*) situados nas duas primeiras colunas, sendo a última referente aos valores de saída obtido. Não foram expressos seus valores, pois para tal torna-se necessário saber a operação logica utilizada.

Na álgebra booleana, possuem apenas três operações básicas *AND (E), OR (OU), NOT (NÃO*), além das combinações *NAND (NÃO-E) e NOR (NÃO-OU).*

2.2.1. Operação *OR*

Está operação também é conhecida como adição lógica e representada pela expressão booleana a seguir:

$$X = A+B \tag{2}$$

Lê-se da seguinte forma: *X* é igual a *A* ou a *B*, ou seja, *X* somente assumirá o valor 1, quando pelo menos um dos valores de *A* ou *B* forem iguais a 1. Caso contrário, o valor resultante da operação será igual a zero. Assim, construímos a tabela verdade para a operação *OR*, exposta na Tabela 2.

Tabela 2. Tabela verdade da operação *OR*

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Fonte: Autoria própria (2020)

2.2.2. Operação *AND*

Esta operação é também conhecida como multiplicação logica e representada pela expressão booleana a seguir:

$$X = A.B \tag{3}$$

Lê-se da seguinte forma: *X* é igual a *A* e *B*; ou seja, *X* somente assumira o valor 1, quando os valores de *A* e *B* também forem 1, do contrário, em qualquer outra forma o valor de *X* será igual a 0. Logo construímos a tabela verdade para a operação *AND*, vista na Tabela 3.

Tabela 3. Tabela Verdade da operação AND

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Fonte: Autoria própria (2020)

2.2.3. Operação *NOT*

Esta operação é também conhecida como inversão e representada pela expressão booleana a seguir:

$$X = \bar{A} \tag{4}$$

Lê-se da seguinte forma: *X* é igual ao inverso de *A*; ou seja, *X* assumira o valor oposto de *A*. Sendo a única operação possível com uma única variável de entrada. Assim, construímos a tabela verdade para a operação *NOT*, apresentada na Tabela 4.

Tabela 4. Tabela verdade da operação *NOT*

A	X
0	1
1	0

Fonte: Autoria própria (2020)

2.2.4. Operação *NOR*

A representação da expressão booleana é dada a seguir:

$$X = \overline{A + B} \tag{5}$$

Lê-se da seguinte forma: *X* é igual ao inverso de *A* ou *B*, ou seja, *X* assumira o valor 1, quando os dois valores, de *A* e *B* forem 0, se não, o valor de *X* será igual a 0. Assim, construímos a tabela verdade para a operação *NOR*, exibida na Tabela 5.

Tabela 5. Tabela verdade da operação *NOR*

A	B	A+B	X
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Fonte: Autoria própria (2020)

2.2.5. Operação *NAND*

A representação da expressão booleana é dada a seguir:

$$X = \overline{A \,.\, B} \tag{6}$$

Lê-se da seguinte forma: *X* é igual ao inverso de *A* e *B*, ou seja, *X* assumira o valor 1, quando um dos valores de *A* e *B* forem 0, se não, o valor de *X* será igual a 0. Assim, construímos a tabela verdade para a operação *NAND*, expressada na Tabela 6. [5]

Tabela 6. Tabela verdade da operação *NAND*

A	B	A.B	X
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Fonte: Autoria própria (2020)

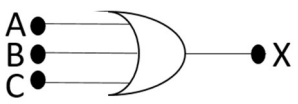
2.3. Circuitos lógicos

Apenas em 1938, o matemático Claude Shannon compreendeu a ligação entre a lógica proposicional e alguns circuitos elétricos, começando a utilizar a álgebra booleana para estruturação de novos estudos na eletrônica.

Desse modo, os circuitos digitais fazem o uso das operações booleanas empregando portas logicas, sendo constituídas de diodos, transistores e resistores, de tal forma que o circuito gere uma saída com o resultado de uma operação algébrica booleana (*OR, AND, NOT*) explicada anteriormente, tendo em vista disso, as portas logicas tem os mesmos princípios e definições das operações booleanas [6].

2.3.1. Porta *OR*

Nos circuitos digitais, portas do tipo *OR* são circuitos que possuem duas ou mais entradas e uma saída *X,* na figura 6 apresentamos um exemplo com 3 entradas *A*, *B* e *C* que passam pela operação booleana *OR*, tendo como saída o resultado *X*

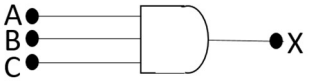


A	B	C	X=A+B+C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Figura 6: Exemplo de representação de uma porta lógica *OR* com três entradas genéricas e sua respectiva tabela verdade. (Autoria própria)

2.3.2. Porta *AND*

Nos circuitos digitais, portas do tipo *AND* são circuitos que possuem duas ou mais entradas e uma saída *X,* na figura 7 apresentamos um exemplo com 3 entradas *A*, *B* e *C* que passam pela operação booleana *OR*, tendo como saída o resultado *X*.

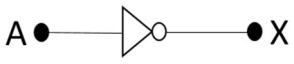


A	B	C	X=A.B.C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Figura 7: Exemplo de aplicação de uma porta lógica *AND* com três entradas genéricas e sua respectiva tabela verdade. (Autoria própria)

2.3.3. Porta *NOT*

As portas do tipo *NOT* se diferenciam por necessitar apenas uma entrada *A* e sua saída *X* é a inversão da variável de entrada, como na operação booleana *NOT,* representado na Figura 8.

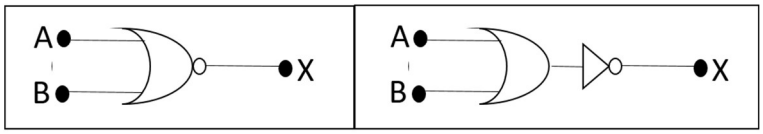


A	X
0	1
1	0

Figura 8: Exemplo de aplicação de uma porta lógica *NOT* com uma entrada genérica e sua respectiva tabela verdade. (Autoria própria)

2.3.4. Porta *NOR*

Nos circuitos digitais, portas do tipo *NOR* são circuitos que possuem duas ou mais entradas e uma saída *X* através da junção das entradas e o inversor como utilizado na operação booleana *NOR,* como exposta as duas possíveis formas de representação na Figura 9.

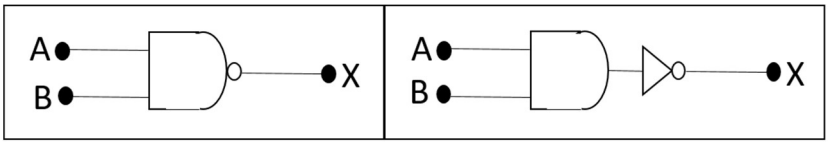


A	B	A+B	X
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Figura 9: Exemplo de aplicação de uma porta lógica *NOR* com duas entradas genéricas e sua respectiva tabela verdade. (Autoria própria)

2.3.5. Porta *NAND*

Nos circuitos digitais, portas do tipo *NAND* são circuitos que possuem duas ou mais entradas e uma saída *X* através da junção das entradas e o inversor como utilizado na operação booleana *NAND* como mostra as duas possíveis formas de representação na Figura 10.



A	B	A.B	X
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Figura 10: Exemplo de aplicação de uma porta lógica *NAND* com duas entradas genéricas e sua respectiva tabela verdade. (Autoria própria)

Portanto, qualquer circuito lógico, complexo ou não, é possível ser escrito utilizando as três operações básicas booleanas e suas junções, facilitando criar um diagrama com todas as respectivas representações das portas utilizadas.

3. METODOLOGIA

Com o objetivo de facilitar o entendimento dos circuitos lógicos, será realizado uma simulação analítica para exemplificar o comportamento dos tipos de circuitos vistos acima (*OR, AND, NOT, NOR, NAND*).

Iremos analisar a saída de circuitos compostos através da fragmentação do circuito completo em pequenos circuitos lógicos básicos, onde cada operação está enumerada de 1 até 4 nos dois exemplos, possibilitando melhorar o entendimento de cada um e depois o circuito completo.

Iremos analisar dois exemplos (a) e (b), mostrados na Figura 11 e na Figura 12, respectivamente.

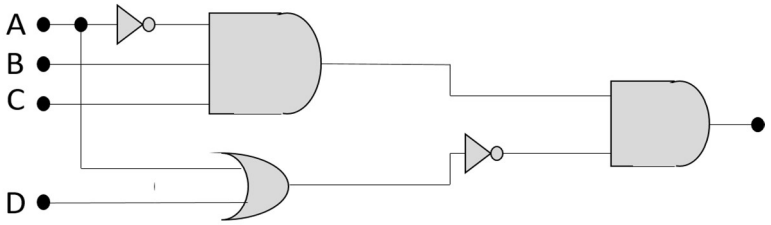


Figure 11. Representação do circuito composto para o problema (a). (Autoria própria)

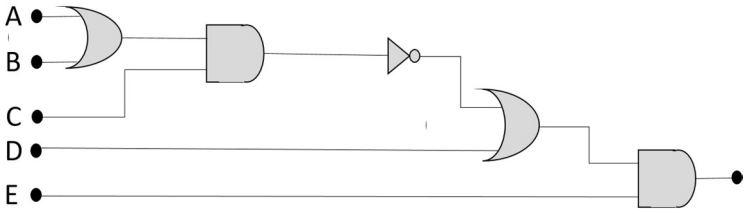


Figure 12. Representação do circuito composto para o problema (b). (Autoria própria)

4.RESULTADO E DISCUSSÕES

Neste tópico apresentamos detalhadamente os resultados parciais e totais dos problemas propostos.

4.1 Problema (a)

Neste problema temos um circuito composto por 4 entradas *A, B, C, D* e operações do tipo *AND, NOR e NOT*. A Figura 13 representa esquematicamente cada etapa do processo que será explicado na sequência.

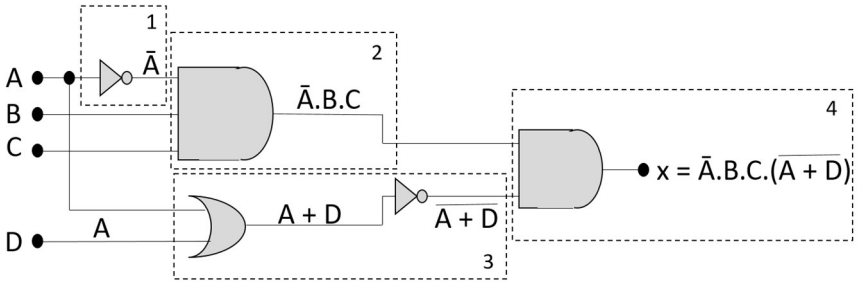


Figura 13. Representação esquemática para solucionar o problema (a). (Autoria própria)

No circuito 1, temos a entrada *A*, que passa por uma porta *NOT*, tendo seu valor alterado para $\bar{A}$. No circuito 2, temos as entradas $\bar{A}$*, B, C*, por se tratar de uma porta *AND* a saída será $\bar{A}$*.B.C*. O circuito 3 é do tipo *NOR*, um *OR* seguido de um *NOT*, com a entrada de *A* e *D* e saída $\overline{A + D}$. Finalmente no circuito 4 temos uma a *AND* com entradas $\bar{A}$*.B.C* e $\overline{A + D}$, resultando na saída total da composição aritmética dada por $X = \bar{A}.B.C.(\overline{A + D})$. A tabela verdade desse circuito composto é mostrada na Tabela 7.

Tabela 7. Tabela verdade do circuito composto do problema (a)

A	B	C	D	$\bar{A}$	$\bar{A}BC$	$\overline{A + D}$	X
0	0	0	0	1	0	1	0
0	0	0	1	1	0	0	0
0	0	1	0	1	0	1	0
0	0	1	1	1	0	0	0
0	1	0	0	1	0	1	0
0	1	0	1	1	0	0	0
0	1	1	0	1	1	1	1
0	1	1	1	1	1	0	0
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0

Fonte: Autoria própria (2020)

4.2 Problema (b)

Neste problema, temos um circuito composto por 5 entradas *A, B, C, D, E* e operações do tipo *OR, NAND e AND.* A Figura 14 representa esquematicamente cada etapa do processo que será explicado na sequência.

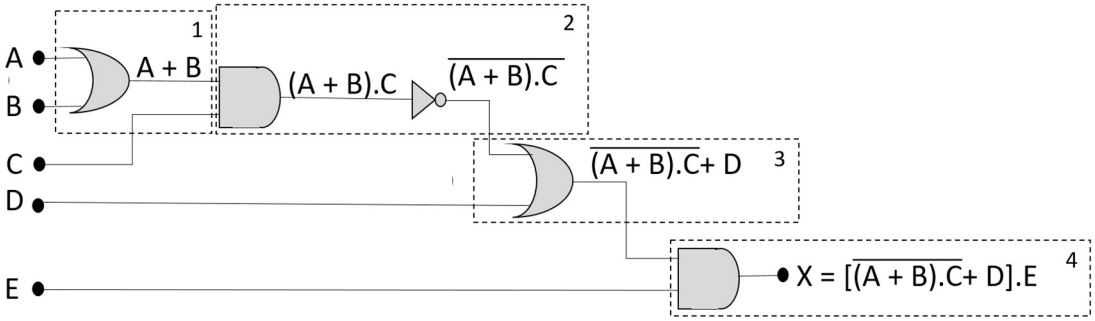


Figura 14. Representação esquemática para solucionar o problema (b). (Autoria própria)

O circuito 1 é composto por duas entradas, *A* e *B* que passam por uma porta do tipo *OR* com a saída *A+B*. No circuito 2, temos as entradas *A+B* e *C*, como é uma porta *NAND*, uma *AND* seguida de uma *NOT*, a saída será $\overline{(\boldsymbol{A} + \boldsymbol{B}).\boldsymbol{C}}$. O circuito 3 é uma porta do tipo OR com entradas $\overline{(\boldsymbol{A} + \boldsymbol{B}).\boldsymbol{C}}$ e D com saída $[\overline{(\boldsymbol{A} + \boldsymbol{B}).\boldsymbol{C}}]$+D. Por fim, no circuito 4 temos uma porta AND com as entradas $[\overline{(\boldsymbol{A} + \boldsymbol{B}).\boldsymbol{C}}]$+D e E gerando uma saída total da composição dada por X = $\{[\overline{(\boldsymbol{A} + \boldsymbol{B}).\boldsymbol{C}}]+D\}.E$. A tabela verdade desse circuito composto é exposta na Tabela 8.

Tabela 8. Tabela verdade do circuito composto do problema (b)

A	B	C	D	E	A+B	$\overline{(A+B).C}$	$\overline{(\overline{(A+B).C}+D}$	X
0	0	0	0	0	0	1	1	0
0	0	0	0	1	0	1	1	1
0	0	0	1	0	0	1	1	0
0	0	0	1	1	0	1	1	1
0	0	1	0	0	0	1	1	0
0	0	1	0	1	0	1	1	1
0	0	1	1	0	0	1	1	0
0	0	1	1	1	0	1	1	1
0	1	0	0	0	1	1	1	0
0	1	0	0	1	1	1	1	1
0	1	0	1	0	1	1	1	0
0	1	0	1	1	1	1	1	1
0	1	1	0	0	1	0	0	0
0	1	1	0	1	1	0	0	0
0	1	1	1	0	1	0	1	0
0	1	1	1	1	1	0	1	1
1	0	0	0	0	1	1	1	0
1	0	0	0	1	1	1	1	1
1	0	0	1	0	1	1	1	0
1	0	0	1	1	1	1	1	1
1	0	1	0	0	1	0	0	0
1	0	1	0	1	1	0	0	0
1	0	1	1	0	1	0	1	0
1	0	1	1	1	1	0	1	1
1	1	0	0	0	1	1	1	0
1	1	0	0	1	1	1	1	1
1	1	0	1	0	1	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	1	0	0	0
1	1	1	0	1	1	0	0	0
1	1	1	1	0	1	0	1	0
1	1	1	1	1	1	0	1	1

Fonte: Autoria própria (2020)

5. CONSIDERAÇÕES FINAIS

O presente estudo teve como principal finalidade analisar a relação entre os números binários, álgebra booleana e circuitos lógicos básicos. Entendemos que este estudo é importante, pois o mesmo faz a ligação entre a matemática na base binária e a construção de circuitos eletrônicos que estão presentes em várias tecnologias que utilizamos diariamente.

Uma das principais aplicações deste conteúdo diz respeito ao contexto computacional, que a partir de circuitos básicos (*OR, AND, NOT*) e suas portas conjuntas (*NOR, NAND*), os mesmos podem ser agregados em formas compostas para os mais variados fins envolvendo uma grande gama de aplicações mais complexas.

Nesse estudo, apresentamos os números binários e posteriormente verificando o comportamento das operações algébricas nessa base numérica, relacionando com a álgebra booleana. Boole facilitou essas operações binárias ao fazer o uso da lógica, criando um sistema simplificado tornando mais fácil a manipulação das variáveis, além da existência uma tabela verdade para cada operação, onde é possível conferir o resultado. O estudo de Boole possibilitou a criação de novas tecnologias empregando a ideia da sua álgebra nos circuitos, elaborando assim as portas logicas.

O foco do presente trabalho foi expor a relação binária da álgebra booleana com os circuitos lógicos, através de uma explanação analítica de cada componente dos problemas. Para esta problemática, apresentamos dois exemplos de circuitos compostos, que consistem em vários circuitos básicos agregados. Podemos observar que não importa o tamanho do circuito, pois podemos fazer a análise de cada circuito básico separadamente e ir progredindo as entradas de acordo com seus resultados de saída parcial, que serão as entradas do próximo circuito, utilizando a estratégia "dividir para conquistar".

Como o estudo tem grande relevância para a computação, é importante que essas relações sejam mais aprofundadas com a ampliação dos circuitos lógicos e de circuitos integrados. Acreditamos que esses conceitos

básicos poderiam ser vistos em estudos de cálculo numérico para fazer a ligação entre a matemática binária e sua transformação e utilização em circuitos eletrônicos sem, contudo, perder o enfoque da disciplina em si. Salientamos que essa sugestão é válida se a carga horária comportar essa alteração.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] COSTA, Cleomar Luiz Da. A História Da Matemática Como Estímulo Ao Ensino-Aprendizagem. 2016. Dissertação (Mestrado). Universidade Federal de Goiás.

[2] MIYASCHITA, W. Y. Sistemas de Numeração: Como funcionam e como são estruturados os números. Bauru, 2002.

[3] MARTINES, Viviane Mantovani. Sistema de numeração binário: dos computadores a sala de aula. 2019. Dissertação (Mestrado). Universidade de São Paulo.

[4] MARTÍNEZ, Héctor Antonio Villa. Sistemas de numeración y aritmética binaria. 2016. Programa de Ciencias de la Computación. Universidad de Sonora.

[5] SANTOS, Vilton Ricardo dos; OLIVEIRA, Cassius Gomes de. A álgebra boolena presente nos circuitos lógicos. Ciências exatas e tecnológicas. Aracaju. V. 3. N. 3. P. 63-72. 2016.

[6] VIEIRA, Fábia Magali Santos. Álgebra booleana. Educação & Tecnologia, [S.l.], v. 5, n. 1, maio 2012. ISSN 2317-7756. Disponível em: <https://www.periodicos.cefetmg.br/index.php/revista-et/article/view/341/356>. Acesso em: 20 de dezembro de 2019.

[7] SAMPEDRO, Javier. Um sistema binário inventado na Polinésia séculos antes de Leibniz. **El País**, Madri, 16 de dezembro de 2013. Disponível em: < https://brasil.elpais.com>. Acesso em: 17 de dezembro de 2019.