Apostila de eletrônica digital da Escola Avançada de Engenharia Mecatrônica

Jorge Luiz Moreira Silva, Celso Massatoshi Furukawa

Resumo – Esta é a apostila de eletrônica digital usada na última edição da Escola Avançada de Engenharia Mecatrônica (EAEM) realizada entre 23 e 30 de julho de 2017 na Escola Politécnica da Universidade de São Paulo. A parte teórica aborda representação numérica em bases não decimais, portas lógicas, diagramas lógicos, álgebra booleana. O curso também inclui uma parte de atividades em laboratório na qual o aluno monta o circuito de um somador completo em *protoboard* com os circuitos integrados OR, AND e XOR. A primeira versão da apostila foi escrita para a EAEM de 2013 e foi submetida a uma atualização na edição de 2016 do evento, resultando na versão que se encontra nas próximas páginas.

Palavras-chave –Eletrônica digital; Sistema binário; Portas lógicas; Álgebra de Boole.

Title – Digital electronics course text of the Advanced Workshop on Mechatronics Engineering

Abstract – This is the course text on digital electronics of the last edition of the Advanced Workshop on Mechatronics Engineering (EAEM, in Portuguese) that was held from July 23 to 30, 2017, at Escola Politécnica of the University of São Paulo, São Paulo, Brazil. Theoretical aspects included are non-decimal number representation, logic gates, logic schematics and Boolean algebra. The course also includes an activity class in laboratory where the student builds the full adder circuit using a protoboard and the OR, AND and XOR gates. The first version of the text was written for the EAEM 2013 and received a major revision for the 2016's edition, resulting on the version available on the following pages.

Keywords – Digital electronics; Binary system; Logic gates; Boolean algebra.

Jorge Luiz Moreira Silva cursa Engenharia Mecatrônica na Escola Politécnica da USP desde 2011. Realizou Duplo Diploma Technische Universität München, Alemanha, onde estagiou no Departamento de Robótica do DLR (Centro Aeroespacial Alemão). Na Escola Politécnica, participou das equipes Partners for the Advancement of Collaborative Engineering (PACE) e do Programa de Educação Tutorial (PET) – Mecatrônica por dois anos. Dentro do PET, foi coordenador da Escola Avançada de Engenharia Mecatrõnica (EAEM) 2013.

Celso Massatoshi Furukawa é engenheiro eletrônico (1987) e mestre em engenharia (1992) pela Escola Politécnica da USP e doutor em engenharia pela Universidade de Tóquio (1996). É docente da Escola Politécnica desde 1989, onde ministra atualmente disciplinas da área de eletrônica nos cursos de graduação em Engenharia Mecatrônica e Engenharia Mecânica. Seus principais campos de interesse são navegação inercial, sistemas embarcados, sistemas abertos, ensino de engenharia.

Versão inicial submetida em 13 de dezembro de 2017. Versão final aceita em 15 de dezembro de 2017. Publicado em 15 de dezembro de 2017. Digital Object Identifier 10.11606/issn.2526-8260.mecatrone.2017.141568

Eletrônica Digital para Mecatrônica

Jorge Luiz Moreira Silva
Prof. Dr. Celso M. Furukawa

PARTE I TEORIA

1 Números Binários, Hexadecimais

Os números são representados de forma posicional, ou seja, os números à esquerda possuem maior peso do que os à direita, por exemplo,

$$7543 = 7 * 10^3 + 5 * 10^2 + 4 * 10^1 + 3 * 10^0.$$

De forma geral, um número inteiro qualquer *A* pode ser representado na base decimal por *n* dígitos de 0 a 9 como $a_{n-1} \ldots a_2 a_1 a_0$, tal que

$$A = a_{n-1} * 10^{n-1} + a_{n-2} * 10^{n-2} + \cdots + a_2 * 10^2 + a_1 * 10^1 + a_0 * 10^0$$

No entanto, números inteiros podem ser representados em qualquer outra base *r* inteira, diferente de dez, sendo representados, de forma generalizada, como

$$\sum_{k=0}^{n-1} a_k * r^k \, .$$

Considerando-se a base 2, os "dígitos" a_k se resumem apenas a 0 ou 1 – são os chamados *bits*. Por exemplo, convertendo o número "vinte e seis" para a base 2 tem-se

$$11010_2 = 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 = 26_{10}.$$

Na expressão acima, indica-se a base em que o número está representado por meio de subscritos.

Como dito, isso pode ser aplicado a qualquer base inteira. Como outro exemplo, na base 7:

$$462_7 = 4 * 7^2 + 6 * 7^1 + 2 * 7^0 = 240_{10}$$

E por que representar em outras bases? Para nós, que possuímos dez dedos nas mãos, é de grande utilidade utilizar a base dez, por ser mais intuitivo. Em processadores ou sistemas eletrônicos, porém, é muito mais conveniente utilizar apenas sinais "ligado" e "desligado", sem intermediários, por ser mais robusto e simples, justificando-se o uso de números em base binária.

Todas as contas em um computador são realizadas em binário. Toda a informação armazenada é binária. Nós apenas vemos números decimais na tela do computador ou calculadora porque eles são convertidos para a base decimal antes de serem impressos.

1.1 Conversão de binário para decimal

Dado um número em base binária, como convertê-lo para decimal? É simples, basta multiplicar cada bit pela respectiva potência de dois e somar tudo no final. Como exemplo, vamos converter o número $(1101011)_2$ para a base decimal:

(Potências) 6 5 4 3 2 1 0
(Bits) 1 1 0 1 0 1 1 $= 1 * 2^6 + 1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$

Note que as potências de dois começam com dois elevado a ZERO para o bit mais à direita

(o chamado bit menos significativo). Cuidado com esse detalhe: é um erro comum começar as potências pelo expoente um, ao invés de zero.

Fazendo a conta acima, obtém-se

$$64 + 32 + 8 + 2 + 1 = 107_{10}\ .$$

1.2 Conversão de decimal para binário

A conversão da base 10 para a base dois é feita seguindo-se um simples algoritmo. Primeiramente, divide-se o número a ser convertido por dois. Guarda-se o resto e divide-se o quociente da conta por dois. Guarda-se o resto da conta, novamente. Repete-se o processo até que o quociente chegue a zero. Quando isso acontecer, a representação binária do número decimal será os restos, de trás para frente. Convertendo 167_{10}, como exemplo:

Figura 1 Algoritmo para transformar base decimal para base binária

Passo	Número	Quociente	Resto
1	167	83	1
2	83	41	1
3	41	20	1
4	20	10	0

Passo	Número	Quociente	Resto
5	10	5	0
6	5	2	1
7	2	1	0
8	1	0	1

Portanto:

$$167_{10} = 10100111_2 = 128 + 32 + 4 + 2 + 1$$

Uma base comum a ser utilizada para representar os números binários é a hexadecimal, por ser mais compacta. Ela possui 16 símbolos: vai de 0 a 9 e depois inclui as letras A, B, C, D, E e F para representar os números de dez a quinze, como mostra a Tabela I.

Tabela I Símbolos da base hexadecimal

Número	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Símbolo	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

A conversão é realizada da mesma forma entre as bases. Na conversão entre a base binária e hexadecimal, porém, existe a possibilidade de se agrupar em grupos de 4 a representação binária, da direita para esquerda, e substituí-los por seu respectivo hexa. Como exemplo, vamos converter o número em binário $(100\ \ 1010\ \ 0000\ \ 1100)_2$ para hexadecimal:

$$\begin{array}{cccc} 4 & A & 0 & 9 \\ 0100 & 1010 & 0000 & 1100 \end{array} = (4A09)_{16} \quad \text{(ou 4A09 h)}.$$

É importante notar que o primeiro grupo à esquerda tem três símbolos (100), mas, para a conversão, basta completá-lo com zeros à esquerda.

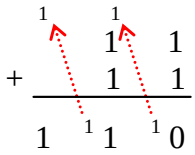
Exercício 1 Faça as seguintes conversões:

a) Converta 100110_2 para base 10.
b) Converta $(117)_{10}$ para base 2.
c) Converta $(B0CA)_{16}$ para base binária.
d) Converta 1000101_2 para base hexadecimal.

2 Soma binária

A soma de números em outras bases é realizada da mesma forma que decimais. A única diferença é que ao invés de se utilizar os dez símbolos decimais (de 0 a 9), são utilizados os respectivos símbolos da base. Para o caso binário, apenas 0 ou 1. Por exemplo, vamos fazer manualmente a soma "11 + 11 = 110" (em decimal, "3 + 3 = 6"). Fazendo-se a soma "coluna a coluna" analogamente ao que é feito ao somar dois números em decimal, tem-se o procedimento ilustrado na Figura 2.

Figura 2 Soma de números em binário



Exercício 2 Faça as seguintes somas em binário:

a) $1001001_2 + 11101_2$
b) $111001_2 + A6_{16}$

3 Portas Lógicas Elementares

Portas lógicas (*logical gates*, em inglês) são circuitos eletrônicos que implementam funções lógicas elementares por meio de sinais elétricos. Os valores lógicos são representados por tensões elétricas padronizadas. Por exemplo, o valor lógico *falso* (ou *zero*, ou *desligado*, ou como você preferir) pode ser associado a uma tensão igual a 0 V, enquanto que o valor lógico *verdadeiro* (ou *um*, ou *ligado*, ou... etc) pode ser associado a 5 V.

Em eletrônica digital, costuma-se designar esses níveis de tensão por

- *L* (*LOW*) - nível lógico baixo,
- *H* (*HIGH*) - nível lógico alto.

Dentro do padrão conhecido como *TTL* (*Transistor-Transistor Logic*), por exemplo, o nível *L* corresponde a uma tensão elétrica entre 0 e 0,8 V, enquanto o nível *H* vai de 2 a 5 V. Note que os níveis lógicos não são representados por tensões exatas, mas sim por faixas de tensão que não se sobrepõem.

Em prol de uma notação mais limpa, vamos representar o nível de tensão *L* pelo símbolo lógico '0' (zero). Ao nível de tensão *H*, associamos o símbolo '1' (um).

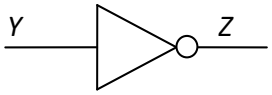
Vamos ver as portas que implementam as três funções lógicas elementares: NOT, AND e OR.

3.1 Porta NOT (inversora, complemento ou negação lógica)

A porta NOT possui uma entrada *Y* e uma saída *Z*, tal que NOT(*Y*) = *Z*. A Figura 3 mostra o símbolo da porta NOT, também conhecida como *inversora*. A saída fornece o nível lógico complementar ao da entrada, ou seja

- se *Y* = 0 então *Z* = 1
- se *Y* = 1 então *Z* = 0

Figura 3 Símbolo da porta lógica NOT (inversora)



A porta NOT implementa eletronicamente a negação lógica – num exemplo simples, dada a afirmação "A luz está acesa", a NEGAÇÃO desta resultará em Verdade quando a luz estiver apagada, e Falso em caso contrário.

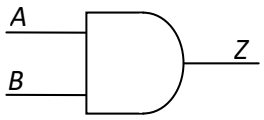
Matematicamente, a função NOT costuma ser indicada de diversas formas diferentes. As mais comuns são

$$\mathrm{NOT}(Y) = Y' = \overline{Y} = /Y . \qquad (1.1)$$

3.2 Porta AND (E lógico)

A porta AND possui duas entradas (*A* e *B*) e uma saída (*Z*). A Figura 4 mostra o símbolo da porta AND e a sua *Tabela da Verdade* (valores de saída tabelados para cada possível combinação de entradas). A notação matemática para se representar uma operação AND é a de um produto.

Figura 4 Símbolo da porta lógica AND e sua Tabela da Verdade



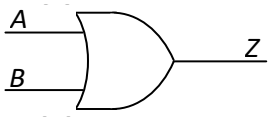
A	***B***	***Z = A.B***
0	0	0
0	1	0
1	0	0
1	1	1

A lógica AND verifica se duas condições são verdadeiras e, se este for o caso, a saída será verdadeira. Por exemplo, um carro só pode ser ligado se possuir um motor E combustível. Caso um dos dois esteja ausente (seja falso), o carro não poderá ser ligado, ou porque simplesmente não tem o motor ou porque não tem fonte de energia para alimentá-lo ou porque não tem os dois.

3.3 Porta OR (OU lógico)

A porta OR também possui duas entradas e uma saída. A Figura 5 mostra o símbolo da porta OR e a tabela da verdade desta. Matematicamente, denota-se a operação com o símbolo '+'.

Figura 5 Símbolo da Porta OR e sua Tabela da Verdade



A	***B***	***Z = A+B***
0	0	0
0	1	1
1	0	1
1	1	1

A porta de saída OR se torna verdadeira se uma OU outra entrada são verdadeiras, ou se ambas o são. Uma aplicação disso seria na aplicação de aprovação de alunos em uma disciplina: caso o aluno tire mais que 5 no trabalho OU na prova ele está aprovado. Ou seja, o aluno só estará reprovado na condição de não passar nem no trabalho nem na prova.

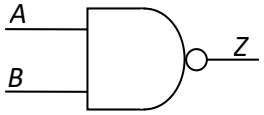
3.4 Portas NAND e NOR

Combinando as três funções lógicas elementares (NOT, AND e OR), pode-se construir outras funções. É o caso, por exemplo, das funções NAND e NOR - combinações bastante simples, mas de extrema importância.

A NAND

Uma porta NAND é uma porta AND seguida por um inversor. Seu símbolo lógico e tabela da verdade estão na Figura 6. Note que o símbolo do inversor foi simplificado, restando apenas a bolinha.

Figura 6 Símbolo lógico e tabela da verdade da porta NAND

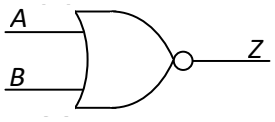


A	*B*	*Z = (A.B)'*
0	0	1
0	1	1
1	0	1
1	1	0

B NOR

Uma porta NOR é constituída por uma porta OR seguida por um inversor, conforme mostra a Figura 7.

Figura 7 Símbolo e tabela da verdade da porta NOR

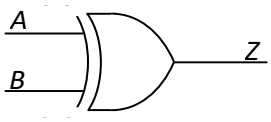


A	*B*	*Z = (A+B)'*
0	0	1
0	1	0
1	0	0
1	1	0

3.5 Porta XOR (OU-EXCLUSIVO)

A função OU-EXCLUSIVO (conhecida como XOR) é uma função lógica que tem várias aplicações. Como seu nome indica, a função XOR difere da função OR por excluir a condição em que ambas as entradas estão em um. A Figura 8Figura 6 mostra o símbolo da função XOR e a sua tabela da ver-dade. Denotamos esta operação com o símbolo '⊕'.

Figura 8 Símbolo da Porta XOR e sua Tabela da Verdade



A	*B*	*Z = A⊕B*
0	0	0
0	1	1
1	0	1
1	1	0

1

Como qualquer função lógica não elementar, a função XOR pode ser construída a partir das funções elementares. Por ora, acredite que

$$A \oplus B = \overline{A}.B + A.\overline{B}. \qquad (1.2)$$

4 Circuito Integrado (CI)

A maioria dos circuitos digitais, desde as portas lógicas vistas nas seções anteriores até os poderosos micro-processadores de última geração, são fabricados em lâminas silício – um material semicondutor. Em um único centímetro quadrado de silício são integrados milhões de transistores e outros dispositivos eletrônicos em escala microscópica, constituindo os chamados circuitos integrados (CI), também conhecidos como *chips*. Por serem muito frágeis, os *chips* são encapsulados em pastilhas de epoxi ou cerâmica, contendo pinos metálicos ligados aos pontos de entrada e saída do circuito.

Figura 9 Circuitos Integrados

5 Diagrama Lógico

Qualquer função lógica pode ser implementada a partir das portas elementares NOT, AND e OR. Pode-se, por exemplo, construir uma porta XOR a partir da expressão 1.2. A Figura 10 mostra o esquema de conexões de um circuito que implementa uma função XOR de duas entradas.

Um esquema como este é chamado de *diagrama lógico* – ou, carinhosamente, de *DL*. O circuito utiliza três CIs. O CI 74HC04, por exemplo, contém seis portas NOT em seu interior, sendo que apenas duas delas (indicadas por U1a e U1b) são usadas. As outras portas estão indicadas na Tabela II.

O DL da Figura exemplifica algumas boas normas de documentação de circuitos lógicos:

- Os componentes possuem um identificador (U1, U2, U3) e o respectivo código comercial (74HCxx).
- Múltiplas portas de um mesmo CI são diferenciadas por letras (U1a, U2b, etc).
- Os pinos dos CIs ligados às portas estão numerados
- Os sinais mais importantes são identificados por nomes (*A*, */A*, etc)

Figura 10 Diagrama lógico de um circuito XOR construído com portas elementares

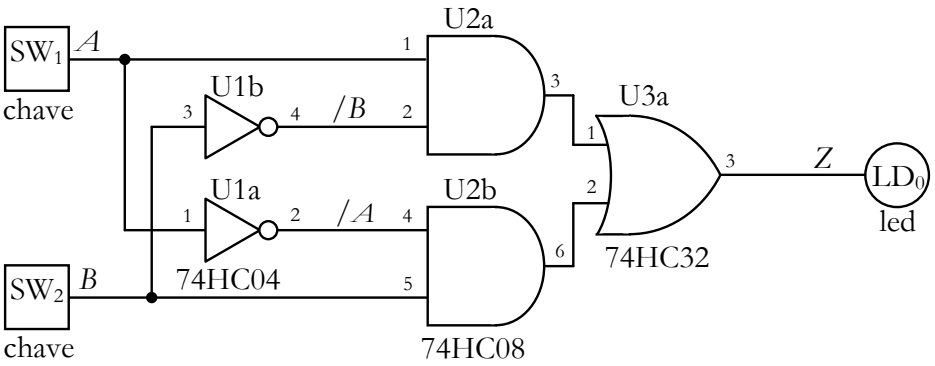


Tabela II Nomenclaturatura do CI's

Código	Tipo de Porta	Número de Portas
74HC04	NOT	6
74HC08	AND	4
74HC32	OR	4

Exercício 3 A Tabela III é a chamada *tabela da verdade* completa do circuito. Ela contém o valor das saídas de cada porta lógica para cada combinação possível de entradas. Complete a tabela da verdade abaixo e mostre que o resultado desta expressão é exatamente igual ao de uma porta XOR.

Tabela III Tabela da verdade completa do circuito da Figura 10

A	***B***	***/A*** (U1 pino 2)	***/B*** (U1 pino 4)	***A./B*** (U2 pino 3)	***/A.B*** (U2 pino 6)	***A./B + /A.B*** (U3 pino 3)
0	0					
0	1					
1	0					
1	1					

Tabelas da verdade completas como essa são muito úteis para localizar defeitos em circuitos lógicos. Por exemplo, se fazemos $A = 0$ e $B = 0$ no circuito e a saída resulta em 1, uma ou mais portas estão com defeito ou pode haver algum mau contato. Nestas condições, podemos verificar o valor da saída de cada porta lógica, em busca das que não reproduzem o valor previsto.

6 Introdução à Álgebra de Boole

As funções lógicas AND e OR constituem uma álgebra, denominada *Álgebra de Boole*. É por esse motivo que são representadas matematicamente pelos habituais operadores algébricos '.' e '+' respectivamente.

A definição formal da álgebra determina que o seu conjunto domínio inclua pelo menos dois elementos: 0 (elemento neutro da operação '+') e 1 (elemento neutro da operação '.'). No caso particular em que o conjunto de domínio é composto por apenas 0 e 1, temos a *Álgebra de Chaveamento*, que constitui a base teórica dos sistemas digitais.

Como toda boa álgebra, a álgebra de Boole pode ser formalmente definida por um conjunto de postulados (ou axiomas), a partir dos quais deriva-se propriedades e teoremas que constituem uma "caixa de ferramentas" para manipular expressões algébricas. A seguir, listamos algumas propriedades básicas, que você pode confirmar analisando as tabelas da verdade das funções elementares.

$$\textit{Elemento neutro} \begin{cases} 1.a = a.1 = a \\ 0 + a = a + 0 = a \end{cases} \tag{1.3}$$

$$\textit{Complemento} \begin{cases} a.\bar{a} = \bar{a}.a = 0 \\ a + \bar{a} = \bar{a} + a = 1 \end{cases} \tag{1.4}$$

$$\textit{Comutatividade} \begin{cases} a.b = b.a \\ a + b = b + a \end{cases} \tag{1.5}$$

$$\textit{Idempotência} \begin{cases} a.a = a \\ a + a = a \end{cases} \tag{1.6}$$

$$\textit{Absorção} \begin{cases} 0.a = a.0 = 0 \\ 1 + a = a + 1 = 1 \end{cases} \tag{1.7}$$

$$\textit{Involução } \left(\overline{\bar{a}}\right) = a \tag{1.8}$$

$$\textit{Distributividade} \begin{cases} (a + b).c = a.c + b.c \\ (a.b) + c = (a + c).(b + c) \end{cases} \tag{1.9}$$

$$\textit{De Morgan} \begin{cases} \overline{(a + b)} = \bar{a}.\bar{b} \\ \overline{(a.b)} = \bar{a} + \bar{b} \end{cases} \tag{1.10}$$

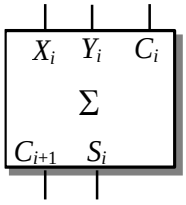
A segunda igualdade da propriedade 1.9 pode causar estranheza, pois mostra que a operação '+' é distributiva sobre a operação '.' na álgebra de Boole.

A última propriedade, De Morgan, é de extrema importância e num primeiro momento pode parecer pouco intuitiva, mas pode ser pensada da seguinte forma. Para a primeira igualdade do 1.10, temos que a negação de um OR resulta em um AND entre as negações, por exemplo, considere a afirmação" Vou ao cinema OU ao teatro" - a NEGAÇÃO dela seria o mesmo que dizer "Não vou ao cinema E NÃO vou ao teatro".

7 Somador Binário Completo

O somador completo – ou *full adder*, soma os três bits de entrada X_i, Y_i e C_i, e gera os bits de saída C_{i+1} e S_i. Em outras palavras, o circuito faz a soma aritmética de três bits, gerando nas saídas um número em binário de 00 a 11 (0 a 3, em decimal). O esquema desse circuito e a sua tabela da verdade estão na Figura 11. A entrada C_i é conhecida como *carry-in* (*vem-um*) e a saída C_{i+1} é chamada de *carry-out* (*vai-um*).

Figura 11 Somador completo e sua tabela da verdade



X_i	Y_i	C_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Você saberia dizer qual é a função lógica correspondente à saída S_i?

$$S_i = X_i \oplus Y_i \oplus C_i \text{. (POR QUE?)} \qquad (1.10)$$

A expressão de C_{i+1} é mais complexa, mas podemos concluir que será igual a 1 sempre que a soma for maior ou igual a dois (10, em binário). Portanto, C_{i+1} deve detectar situações em que pelo menos 2 das entradas sejam iguais a 1. Isto é, C_{i+1} é 1 quando X_i e Y_i valem 1, ou X_i e C_i valem 1, ou Y_i e C_i valem 1. Matematicamente, obtém-se

$$C_{i+1} = X_i.Y_i + X_i.C_i + Y_i.C_i \text{.} \qquad (1.11)$$

Exercício 4 Preencha a tabela da verdade completa do Si e do C_{i+1}. Verifique que essas expressões realmente representam o somador binário.

Tabela IV Tabela da verdade do Si

X_i	Y_i	C_i	$X_i \oplus Y_i$	$(X_i \oplus Y_i) \oplus C_i = S_i$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Tabela V Tabela da verdade do Ci+1

X_i	Y_i	C_i	X_iY_i	X_iC_i	Y_iC_i	$X_iY_i + X_iC_i$	$(X_iY_i + X_iC_i)+Y_iC_i = C_{i+1}$
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

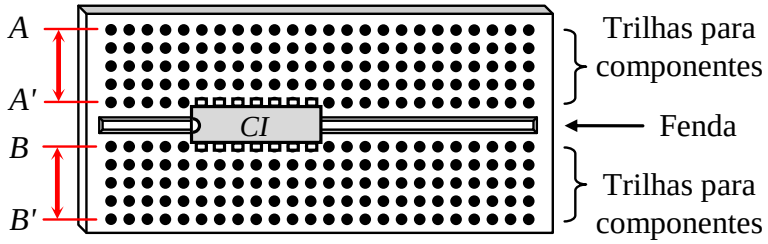
PARTE II PRÁTICA

1 O protoboard

Nesta experiência, usaremos a placa para a montagem e teste de circuitos eletrônicos conhecida como *protoboard* mostrada na ilustrado na Figura 12.

O *protoboard* é uma barra com vários pontos de conexão espaçados de 0,1 polegadas, onde podem ser encaixados os terminais de circuitos integrados e componentes discretos. Uma fenda central divide a barra em duas metades. Em cada metade estão dispostas várias trilhas verticais de cinco furos.

Figura 12 Barra de conexões *protoboard*



As setas indicam como os pontos estão conectados internamente entre si por lâminas metálicas condutoras. Uma trilha vertical de 5 furos na posição (*A-A'* ou *B-B'*) NÃO ESTÁ em curto com outras trilhas paralelas a ela e também NÃO HÁ conexão entre trilhas verticais *A-A'* e *B-B'* de cada lado da fenda .

Para montar os circuitos no *protoboard*, coloque integrados com encapsulamento DIP (*Dual In Paralell*) longitudinalmente sobre a fenda de uma barra de conexões, como mostra a figura. Desta forma, cada pino ocupará um furo de uma trilha vertical isolada das demais, e você terá os quatro furos restantes da trilha para ligar o pino a outro ponto do circuito.

2 Placa XLA

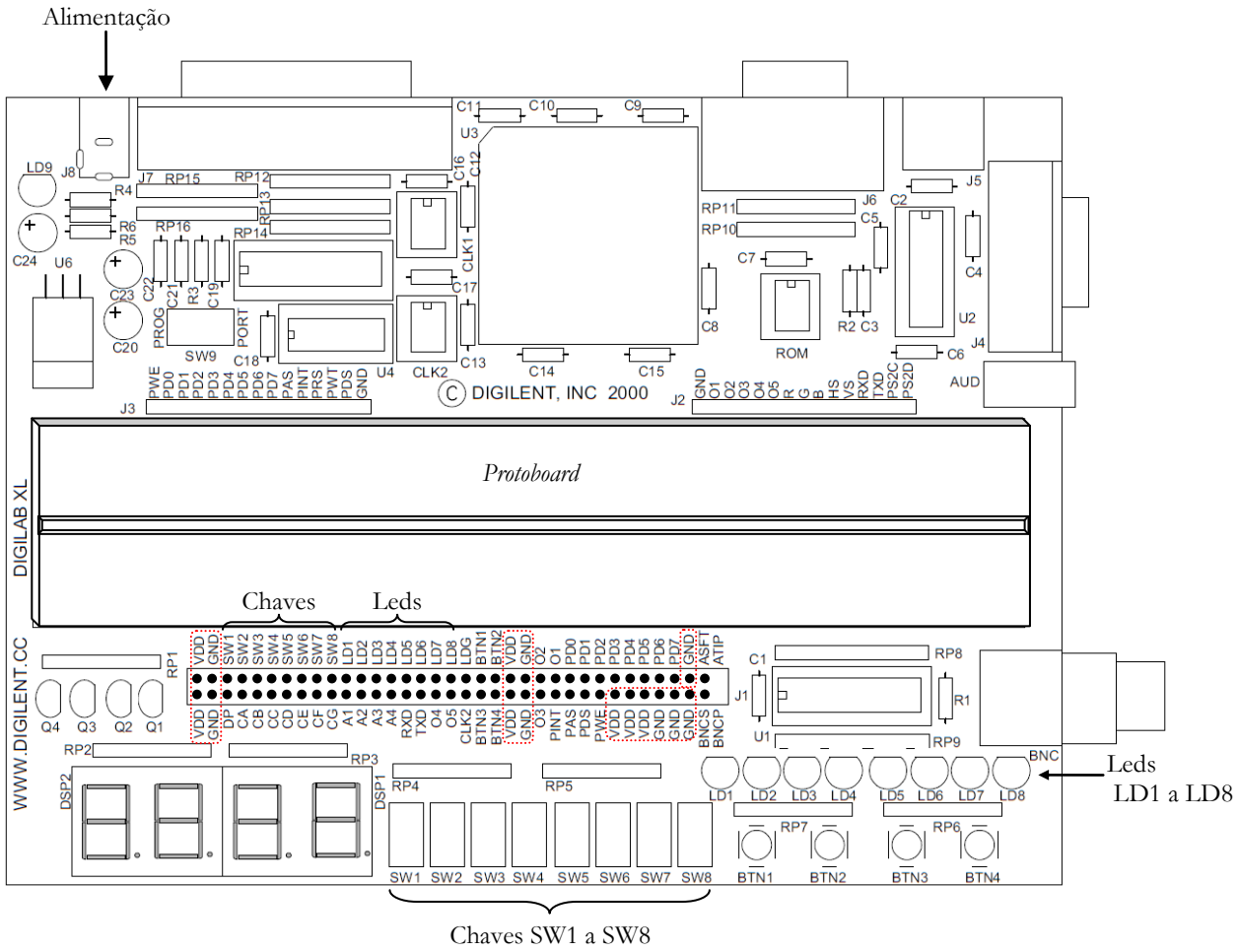
No laboratório, usaremos a placa XLA mostrada na Figura 13 para montar e testar os circuitos digitais. Ela contém vários componentes, mas para esta experiência precisaremos apenas do *protoboard*, das chaves e leds.

A placa contém oito chaves do tipo liga-desliga na parte inferior, numeradas de SW1 a SW8. Na barra de terminais situada logo abaixo do *protoboard*, há um ponto ligado a cada uma delas. Num ponto tem-se 0 V (nível *LOW*) quando a chave correspondente está desligada e 5 V (nível *HIGH*) quando ligada. A chave permanece desligada quando posicionada para frente e é ligada quando acionada para trás.

Os oito leds, numerados de LD1 a LD8, se encontram na parte inferior direita da placa. Para cada led também há um ponto correspondente na barra de terminais. Aplicando-se uma tensão de 5 V a um ponto, acende-se o led correspondente.

Atente também para os pontos “VDD” e “GND” na barra de terminais da Figura 13 (indicados pelas linhas pontilhadas). Em VDD tem-se a tensão de alimentação de 5 V (que chamamos de V_{CC} neste texto) e em GND tem-se 0 V. Use-os para alimentar os CIs.

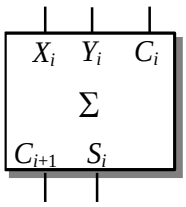
Figura 13 Placa didática XLA



3 Circuito Somador Completo

Nesta experiência, vamos montar no *protoboard* da placa XLA o somador completo visto na parte teórica. Para sua comodidade, copiamos abaixo a tabela da verdade e as equações lógicas do somador.

Tabela 6 Saídas Ci+1 e Si do somador completo



X_i	Y_i	C_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$S_i = X_i \oplus Y_i \oplus C_i, \text{ e} \qquad (1)$$

$$C_{i+1} = X_i.Y_i + X_i.C_i + Y_i.C_i . \qquad (2)$$

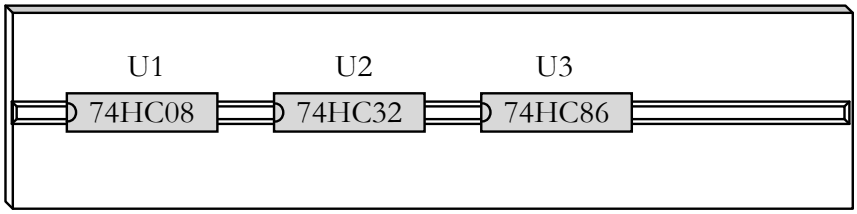
4 Circuitos Integrados (CIs)

Usaremos os seguintes componentes:

- 74HC08: quad AND de duas entradas
- 74HC32: quad OR de duas entradas
- 74HC86: quad XOR de duas entradas

Eles já se encontram inseridos no *protoboard* da placa XLA como mostra a Figura 14.

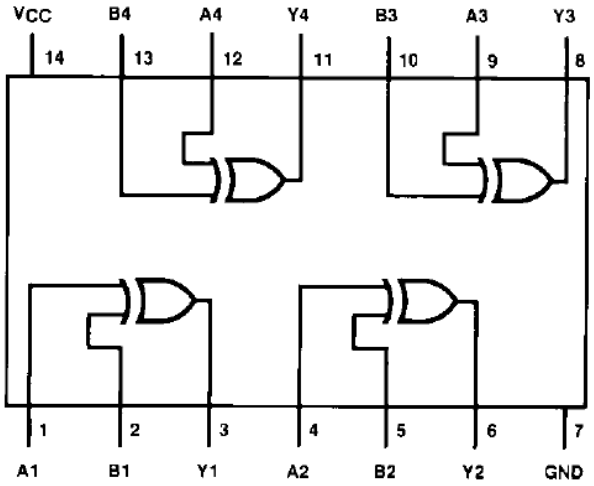
Figura 14 Disposição dos componentes no *protoboard*.



4.1 Bit Si do somador completo

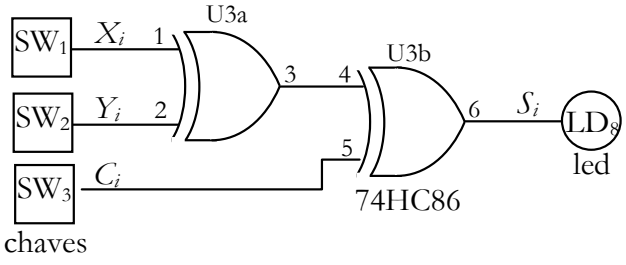
Para implementar o bit S_i, precisamos fazer um OU-Exclusivo (XOR) entre os três bits de entrada (X_i, Y_i e C_i), conforme mostra a expressão (1). Não precisaremos construir o circuito XOR usando portas AND, OR e NOT como vimos na parte teórica porque dispomos no laboratório do CI 74HC86, que contém quatro portas XOR de duas entradas. A Figura 15 mostra a disposição das portas no componente.

Figura 15 74HC86 (XOR)



Como cada porta possui apenas duas entradas, precisaremos "cascatear" duas delas (ou seja, ligar a saída de uma na entrada de outra), como mostra o diagrama lógico da Figura 16.

Figura 16 Diagrama lógico do circuito de S_i



Você deve montar as conexões indicadas no diagrama lógico seguindo o número dos pinos. Por exemplo, "ligar o pino 3 ao pino 4". Vamos aos detalhes (tendo dúvidas, chame um monitor).

- ✓ Localize o CI no protoboard da placa XLA: esta no extremo direito (confira o código impresso nele).
- ✓ Veja a posição de cada pino do CI no desenho da Figura 15.
- ✓ Use os cabinhos no interior da caixa de plástico em sua bancada.
- ✓ ATENÇÃO: os pinos estão numerados em sentido ANTI-HORÁRIO, começando pelo inferior esquerdo.
- ✓ Comece fazendo as conexões de alimentação (elas não aparecem no DL mas são necessárias!)
  - Ligue o pino **14** (superior esquerdo) a um ponto "**VDD**" da barra de terminais
  - Ligue o pino **7** (inferior direito) a um ponto "**GND**" da barra de terminais
- ✓ Complete as demais conexões (os contatos das chaves SW_i e do led LD_8 também estão na barra de terminais).

Feito isso, chame o professor ou monitor para conferir o circuito.

4.2 Teste do circuito de Si

Para energizar a placa XLA e testar o circuito, conecte a fonte da placa em uma das tomadas da bancada e insira o plugue de alimentação no conector de alimentação (canto superior esquerdo).

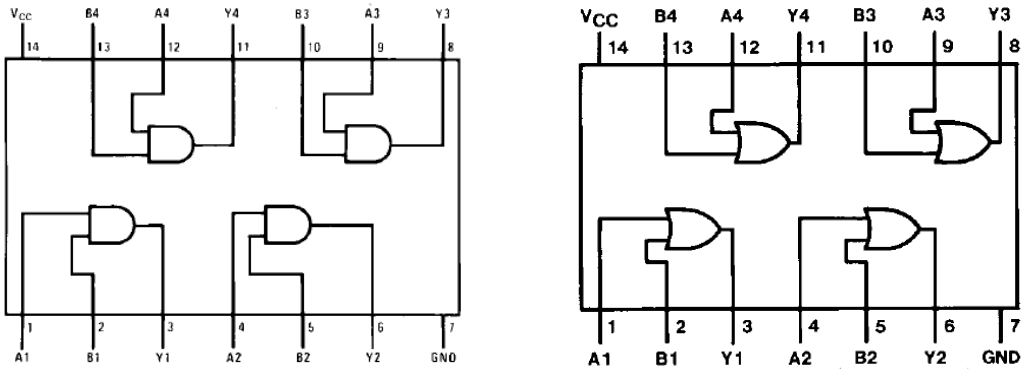
- ✓ Teste as oito configurações de entrada nas chaves SW1, SW2 e SW3 (0 para frente; 1 para trás).
- ✓ Observe o estado do led LD8 (0 = apagado; 1 = aceso).
- ✓ Verifique se o circuito reproduz os valores previstos para Si na Tabela 6. Resumidamente:
  - LD8 apagado se houver um número PAR de chaves em 1.
  - LD8 aceso caso contrário.

Ao terminar de testar, desligue a fonte: tire fonte da tomada.
NÃO DESMONTE O CIRCUITO. Vamos usá-lo novamente.

4.3 Bit Ci+1 do somador completo

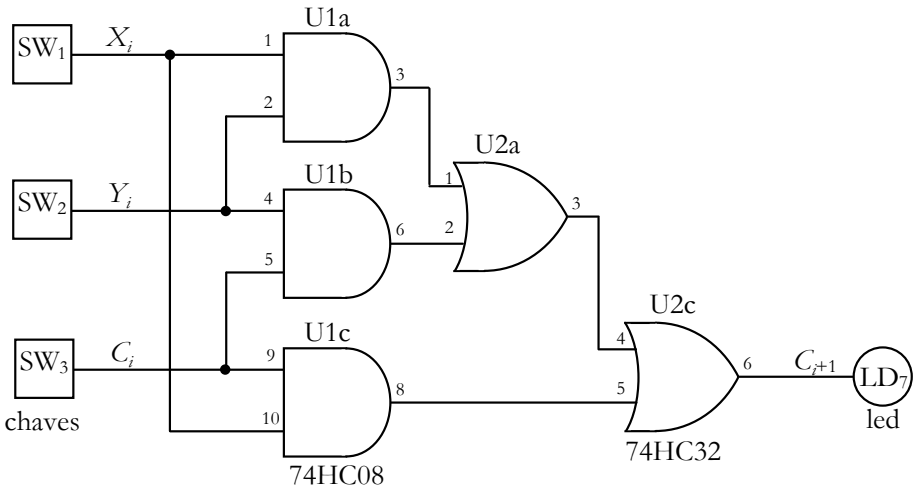
Para implementar o bit C_{i+1} seguindo a expressão (2), usaremos as portas AND do CI 74HC08 e as portas OR do 74HC32. A Figura 17 mostra a pinagem desses componentes.

Figura 17 74HC08 (AND, à esquerda) e 74HC32 (OR, à direita)



Como as portas do 74HC32 possuem apenas duas entradas e a expressão (2) contém um OR entre três produtos, teremos que cascatear também duas delas. A Figura 18 mostra o diagrama lógico do circuito resultante.

Figura 18 Diagrama lógico do circuito de C_{i+1}



Monte o circuito no *protoboard* (tendo dúvidas, chame um monitor):

- ✓ Localize os CIs: o '08 (U1) é o da esquerda e o '32 (U2) é o do meio (confira os códigos impressos neles).
- ✓ Veja a posição dos pinos nos desenhos da Figura 17.
- ✓ LEMBRE-SE: os pinos estão numerados em sentido ANTI-HORÁRIO, começando pelo inferior esquerdo.
- ✓ Comece fazendo as conexões de alimentação (elas não aparecem no DL mas são necessárias!)
  - Ligue os pinos **14** dos dois CIs em "**VDD**" (na barra de terminais)
  - Ligue os pinos **7** em "**GND**" (na barra de terminais)
- ✓ Deixe as conexões com as chaves SW_i por último; complete as demais (o contato do led LD_7 também está na barra de terminais).

Vamos usar as mesmas chaves para fornecer entradas simultâneas aos dois circuitos do somador (S_i e C_{i+1}). Para isso, interligue os sinais X_i, Y_i e C_i do circuito da Figura 18 (C_{i+1}) aos pinos correspondentes do circuito da Figura 16 (S_i). Ou seja

- [U1, pino 1 ou pino 10] com [U3, pino 1]
- [U1, pino 2 ou pino 4] com [U3, pino 2]
- [U1, pino 5 ou pino 9] com [U3, pino 5]

Quando terminar a montagem, chame o professor ou monitor para conferir o circuito.

4.4 Teste do somador completo (Ci+1 e Si)

É chegada a hora da verdade.

Conecte a fonte da placa em uma tomada da bancada e insira o plugue de alimentação no conector de alimentação (canto superior esquerdo).

- ✓ Teste as oito configurações de entrada nas chaves SW1, SW2 e SW3 (0 para frente; 1 para trás).
- ✓ Observe os leds LD7 e LD8 (respectivamente C_{i+1} e S_i).
- ✓ Verifique se o circuito SOMA EM BINÁRIO os três bits de entrada, conforme previsto na Tabela 6.

Se tudo deu certo, PARABÉNS! Pode ser o começo do seu futuro projeto de supercomputador.

5 Finalização

- ✓ **Placa XLA e fonte** A placa foi colocada no plástico antiestático e dentro da na caixa **com sua fonte**?
- ✓ **Cabinhos** A caixa deve conter 30 cabinhos de cores variadas. Ela está completa?
- ✓ **Perdidos** Há componentes ou cabinhos caídos no chão?
- ✓ **Empréstimos** Usou alguma coisa de outra bancada? O quê? Foi devolvido?
- ✓ **Limpeza** A bancada está limpa?

(Antes da Glória, Muito Trabalho Duro...)