ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos

PMR3333 Eletrônica para Engenharia Mecânica 2º semestre de 2019

Nome: _____ Nº USP: _____

Experiência 6 CIRCUITOS COMBINACIONAIS E BARRAMENTOS

Nesta experiência, usaremos a álgebra de Boole para resolver um problema lógico através de um circuito digital. E usaremos mapas de Veitch-Karnaugh para simplificar o circuito antes de implementá-lo. Veremos também três elementos importantes, muito utilizados em sistemas digitais: o decodificador, o multiplexador e o barramento. Veremos como usar multiplexadores para implementar funções lógicas.

Para esta experiência, você deve revisar os tópicos sobre álgebra de Boole e mapas de Veitch-Karnaugh, contidos na seção 12.2.4 do livro-texto ("*Practical Electronics*"). Estude também os seguintes tópicos: multiplexadores (12.3.1) e decodificadores (12.3.2).

- Estude a apostila **com antecedência**. Sua compreensão será avaliada na aula por **ARGUIÇÃO ORAL**.
- Faça os **EXERCÍCIOS** contidos na apostila e tire dúvidas com os professores **com antecedência**.
- Traga para a aula a apostila **IMPRESSA**.

PARTE A TEORIA

6.1 Teorema de De Morgan

Uma das propriedades mais importantes das funções lógicas é expressa pelos dois teoremas de De Morgan,

$$\begin{cases} \overline{A.B.C.\cdots} = \overline{A} + \overline{B} + \overline{C} + \cdots \\ \overline{A + B + C + \cdots} = \overline{A}.\overline{B}.\overline{C}.\cdots \end{cases} \tag{6.1}$$

Repare como esses teoremas são coerentes com a nossa lógica cotidiana. Por exemplo, para entender a primeira igualdade, imagine que seu vizinho afirme que ele "*tem uma Ferrari*" E "*tem um Porsche*". Portanto, ele está mentido se "*não tem uma Ferrari*" OU "*não tem um Porsche*", ou se não tem nenhum dos dois (o que é mais provável). O exemplo para ilustrar a segunda igualdade fica por sua conta.

Uma aplicação imediata desses teoremas é a substituição de portas AND por portas OR, e vice-versa, em circuitos digitais. Delas também derivam duas outras propriedades importantes, conhecidas como *suficiência de NAND* e *suficiência de NOR*.

6.1.1 Suficiência de NAND

Qualquer função lógica pode ser implementada utilizando-se apenas portas NAND. Podemos verificar isso com facilidade, na medida em que uma inversora e uma porta OR podem ser substituídas da seguinte forma:

$$\begin{cases} \overline{A} = \overline{A.A} \\ A + B = \left(\overline{\overline{A}.\overline{B}}\right) \end{cases} \tag{6.2}$$

6.1.2 Suficiência de NOR

Analogamente, podemos substituir inversoras e portas AND por portas NOR, pois

$$\begin{cases} \overline{A} = \overline{A + A} \\ A.B = \left(\overline{\overline{A} + \overline{B}}\right) \end{cases} \tag{6.3}$$

6.2 Implicação lógica

A Tabela 6.1 descreve a função lógica denominada *Implicação*.

Tabela 6.1 Tabela da verdade da função implicação

x	***y***	$x \rightarrow y$
0	0	1
0	1	1
1	0	0
1	1	1

Matematicamente, denotamos essa função por "$x \rightarrow y$" (lê-se *x implica y*), e analisando a tabela verificamos que a implicação equivale à seguinte expressão lógica:

$$(x \rightarrow y) = \overline{x} + y \tag{6.4}$$

A função implicação corresponde a uma proposição do tipo "se *x* então *y*", que pode ser interpretada da seguinte forma. Admitindo que a proposição "se *x* então *y*" seja verdadeira (ou seja, $x \rightarrow y = 1$), podemos afirmar que a condição *y* é verdadeira ($y = 1$) sempre que a condição *x* também é ($x = 1$), mas o inverso não necessariamente é válido. Em outras palavras, a proposição "se *x* então *y*" é válida sempre que *x* for um caso particular da condição *y*.

O exemplo a seguir mostra uma aplicação prática da função implicação.

6.2.1 Aplicação: testador de matrículas

No século XVII George Boole formulou a álgebra que leva seu nome com o objetivo de construir uma máquina para provar teoremas. Como exemplo, vamos projetar o "testador de matrículas" que já apareceu em uma das nossas listas de exercícios.

Suponha que um estudante (após alguns tropeços durante o curso) está diante da complicada tarefa de fazer a matrícula em cinco possíveis disciplinas, identificadas por *A*, *B*, *C*, *D* e *E*. Após analisar longamente os horários e requisitos, levantou os quatro quesitos abaixo, de q_1 a q_4. O objetivo é matricular o aluno no maior número possível de disciplinas, satisfazendo TODAS as condições a seguir:

q_1: Matricular-se em *A*, *B* ou *C*, ou em mais de uma delas
q_2: Se matricular-se em *C*, então também deve se matricular em *A*.
q_3: Caso se matricule em *D*, então NÃO deve se matricular em *E*.
q_4: Matricular-se em *A* e *B* conjuntamente, ou não se matricular em nenhuma delas.

Vamos definir as variáveis lógicas *A*, *B*, *C*, *D* e *E* de tal forma que $A = 1$ representa "aluno pede matrícula na disciplina *A*" e que $A = 0$ representa "o aluno não pede matrícula em *A*", e analogamente para as demais variáveis. Em termos lógicos, os quesitos podem ser expressos da seguinte forma:

$$\begin{cases} q_1 = A + B + C \\ q_2 = C \rightarrow A = \overline{C} + A \\ q_3 = D \rightarrow \overline{E} = \overline{D} + \overline{E} \\ q_4 = A \odot B = \overline{A \oplus B} = \overline{A}.\overline{B} + A.B \end{cases} \tag{6.5}$$

Repare que o quesito q_2 é do tipo "se *C* então *A*", que se traduz em termos lógicos em ($q_2 = C' + A$). Se o aluno quiser se matricular em *C*, então deverá se matricular também em *A*; mas se não se matricular em *C*, pode ou não se matricular em *A*.

O quesito q_3 é do tipo "se *D* então não-*E*" ($q_3 = D' + E'$). Para termos $q_3 = 1$, o aluno não poderá se matricular em *D* e *E* simultaneamente; as outras combinações de matrícula ou não em *D* e *E* são permitidas.

Definimos então a função *F*(*A*, *B*, *C*, *D*, *E*) que determina se as matrículas solicitadas podem ser aceitas ($F = 1$) ou não ($F = 0$). Como todos os quesitos devem ser satisfeitos para que a matrícula seja válida, temos

$$F(A,B,C,D,E) = q_1.q_2.q_3.q_4 = (A + B + C).(\overline{C} + A).(\overline{D} + \overline{E}).(\overline{A}.\overline{B} + A.B). \tag{6.6}$$

Nesta experiência, iremos montar o circuito que implementar essa função para testar se uma configuração de matrícula é válida ou não. Começamos pelo mapa de Karnaugh final da função *F* (de cinco variáveis) mostrado na Figura 6.1, que pode ser obtido multiplicando-se os mapas dos quesitos de q_1 a q_4 (isso fica por sua conta).

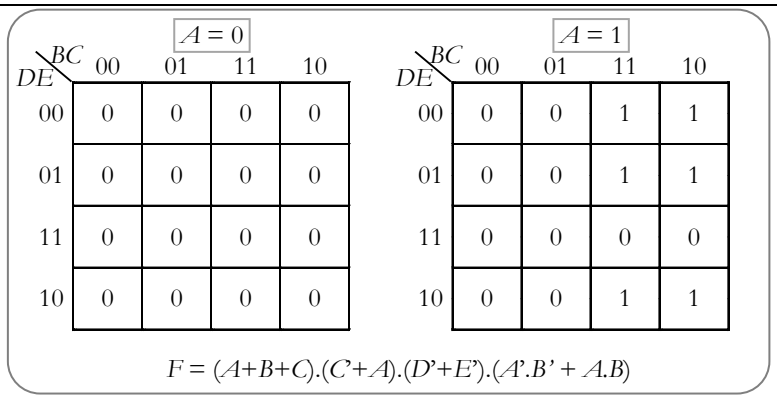


Figura 6.1 Mapa da função de matrícula $F(A, B, C, D, E)$, antes da simplificação

Simplificando esse mapa (o que também é por sua conta), chega-se à expressão simplificada,

$$F(A,B,C,D,E) = A.B.\overline{D} + A.B.\overline{E}\,, \tag{6.7}$$

que usaremos para implementar a função F. De imediato, vê-se que a função independe da variável C, o que significa que a matrícula na disciplina C é *opcional*.

6.3 Circuitos combinacionais

Em geral, circuitos combinacionais possuem várias entradas e várias saídas. A cada instante, as saídas dependem única e exclusivamente do valor das entradas.

Existem funções booleanas padronizadas e implementadas em um único circuito integrado (CI), que simplificam o projeto de circuitos lógicos mais complexos. De maneira geral, um sistema digital pode ser definido em termos de grandes blocos. Chamamos esta técnica de *desenvolvimento modular*. Com isso, podemos construir grandes blocos usando CIs específicos, sem que seja necessário detalhá-los.

Nesta experiência, utilizaremos CIs que implementam duas funções padrão importantes.

- 74HC151 – multiplexador (ou seletor) de 8 entradas.
- 74HC138 – decodificador (ou demultiplexador) de 3 para 8 vias

Estes já viveram seus dias de glória, quando computadores eram construídos com milhares de componentes semelhantes, de baixa densidade, contendo poucas centenas de transistores por CI. Nos dias de hoje, componentes contendo milhões de transistores realizam funções extremamente mais complexas. No entanto, o decodificador e o multiplexador continuam sendo conceitualmente importantes no desenvolvimento de circuitos digitais.

6.4 Multiplexador (mux)

O multiplexador (também conhecido por *mux*) tem como função selecionar uma entrada dentre um conjunto de entradas e copiar o valor desta na saída. O funcionamento de um mux é semelhante a uma chave seletora. A saída do mux é igual ao valor da entrada endereçada pelos sinais de seleção. Em geral, um mux com n sinais de seleção tem 2^n entradas para uma saída. Ou, resumidamente, mux 2^n:1.

Nesta experiência utilizaremos o '151, um mux com três sinais de seleção e portanto oito entradas de dados para uma saída (ou seja, 8:1). Em anexo, encontra-se o *datasheet* do 74ALS151 (*Advanced* TTL), cujo funcionamento e a pinagem é compatível com o 74HC151 (CMOS). Neste caso, S_2, S_1 e S_0 são os sinais de seleção, e I_0 a I_7 são as oito entradas de dados. A entrada selecionada por $S_2S_1S_0$ é copiada na saída Y (e o complementar na saída Y').

A entrada E' faz a função de habilitação (*enable*) do circuito, e opera de forma invertida: deve-se fazer $E' = L$ (isto é, 0 V) para que o multiplexador funcione – é o que chamamos de sinal *ativo em zero*. Caso $E' = H$, a saída Y permanece em L independentemente das demais entradas.

6.5 Sinal *ativo em zero*

Os sinais E' e Y' do mux foram batizados com nomes "negados" para indicar que devem desempenhar alguma função quando estiverem em nível L (e não em H, como seria esperado). Assim a saída do '151 é habilitada quando $E' = 0$, ou inversamente quando $E = 1$. Coerente, não?

É muito comum encontrar sinais de habilitação "ativos em zero" em circuitos digitais por motivos de segurança. Isto porque entradas em aberto são interpretadas como nível H em várias subfamílias de circuitos integrados. Assim, se uma linha de habilitação se abrir por um motivo qualquer (quebra de fio ou queima da saída que

a alimenta) não causará a habilitação inesperada do circuito. É por isso que as saídas do decodificador '138 (que veremos a seguir) são negadas, uma vez que geralmente são utilizadas para habilitar outros circuitos.

Veja como nomes negados e não negados são usados de forma coerente nos *datasheets* e nesta apostila.

6.6 Implementação de funções lógicas usando multiplexadores

Um mux também pode ser usado para implementar um circuito combinacional qualquer. Observe o diagrama lógico do mux '151 no *datasheet* (Fig. 5 do *datasheet*). Repare que a saída Y nada mais é do que um OR de todos os ANDs possíveis dos três sinais de seleção (ou seja, são os *mintermos* $S_2S_1S_0$), sendo que cada AND recebe ainda a respectiva entrada I_i.

Como exemplo, vamos implementar uma função $F(A_3, A_2, A_1, A_0)$ qualquer de quatro variáveis, cuja tabela da verdade é mostrada juntamente com o correspondente mapa de Karnaugh na Tabela 6.2. Para facilitar a explicação, numeramos as linhas da tabela da verdade de 0 a 15 (do lado esquerdo da tabela).

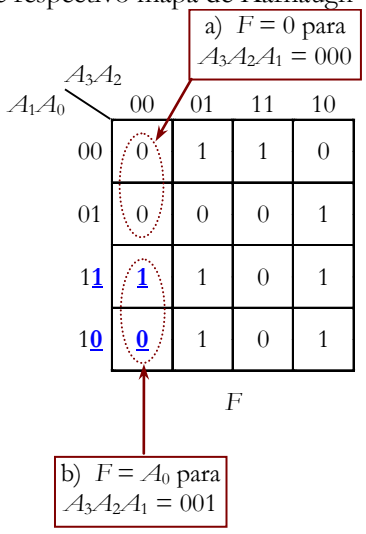
6.6.1 Implementação direta

Podemos implementar essa função usando um multiplexador de dezesseis entradas (16:1), como mostra a Figura 6.2. As variáveis $A_3A_2A_1A_0$ estão conectadas as sinais de seleção do multiplexador e cada entrada de dados I_i está ligada ao nível lógico de saída da função F da i-ésima linha da tabela da verdade. Em outras palavras, utilizamos as variáveis da função F para *endereçar* um dos pinos de entrada do multiplexador, sendo que o pino endereçado deve estar em L ou H de acordo com o desejado para a saída de F.

Por exemplo, a linha 0 da tabela especifica que F deve valer 0 quando $A_3A_2A_1A_0 = 0000$. Então, deve-se fixar em nível L a entrada I_0 do mux da Figura 6.2, que é passado para a saída Y quando $S_3S_2S_1S_0 = 0000$.

Tabela 6.2 Tabela da verdade da função F e respectivo mapa de Karnaugh

	A_3	A_2	A_1	A_0	F
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0



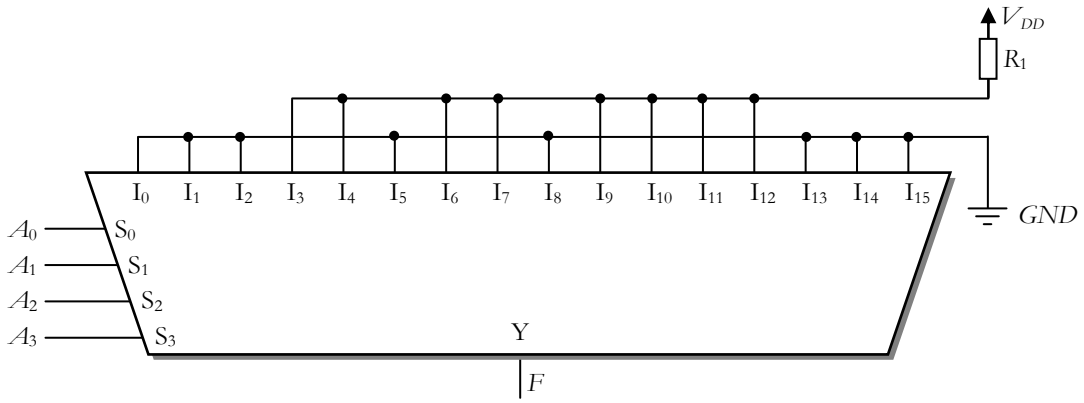


Figura 6.2 Implementação da função F com multiplexador 16:1.

Sobre o resistor R_1: as entradas fixas em H devem ser conectadas a V_{DD} por meio do resistor de *pull-up* R_1 como mostra a Figura 6.2. A função do resistor é limitar a corrente drenada pelas entradas, protegendo a fonte V_{DD} caso uma dessas entradas entre em curto. Por exemplo, para $V_{DD} = 5$ V, um resistor de 10 kΩ limitará a corrente a 0,5 mA mesmo que uma entrada grampeie em 0 V. E como uma entrada CMOS em H drena apenas 1 µA, o resistor de 10 kΩ causará uma queda de tensão de apenas 0,01 V por entrada ligada a ele.

6.6.2 Redução do multiplexador

Há um modo mais compacto de se implementar essa função. A Tabela 6.3 descreve a mesma função de forma condensada, isto é, a saída é dada em função do bit de seleção A_0. Para isto, agrupamos as linhas da Tabela 6.2 duas a duas (ou equivalentemente, as linhas do mapa K de duas em duas). Desta forma, em cada par de linhas, apenas o valor de A_0 varia – repare nas divisões tracejadas da Tabela 6.2 e os agrupamentos verticais indicados na mapa de Karnaugh. Procedemos então da seguinte forma:

- As linhas 0 e 1 da Tabela 6.2 têm em comum que $A_3A_2A_1 = 000$. Além disso, tem-se $F = 0$ tanto para $A_0 = 0$ como para $A_0 = 1$. Isto está indicado no mapa de Karnaugh pela anotação *a*.
- Portanto as linhas 0 e 1 da Tabela 6.2 resultam na linha 0 da Tabela 6.3, na qual se tem $F = 0$ (independe do valor da entrada A_0).
- Nas linhas 2 e 3 da Tabela 6.2, se tem $A_3A_2A_1 = 001$ e F acompanha o valor de A_0. Ou seja, $F = 0$ se $A_0 = 0$, e $F = 1$ se $A_0 = 1$. Essas duas linhas da tabela estão indicadas pela anotação *b* no mapa K.
- Portanto, a partir das linhas 2 e 3 da Tabela 6.2, tem-se $F = A_0$ na linha 1 da Tabela 6.3.
- Já nas linhas 4 e 5 da Tabela 6.2, tem-se $F = A_0$', resultando na linha 2 da Tabela 6.3. E assim por diante.

Tabela 6.3 Tabela da verdade condensada da função F e respectivo mapa de Karnaugh

	A_3	A_2	A_1	F
0	0	0	0	0
1	0	0	1	A_0
2	0	1	0	A_0'
3	0	1	1	1
4	1	0	0	A_0
5	1	0	1	1
6	1	1	0	A_0'
7	1	1	1	0

A_1 \ A_3A_2	00	01	11	10
0	0	A_0'	A_0'	A_0
1	A_0	1	0	1

F

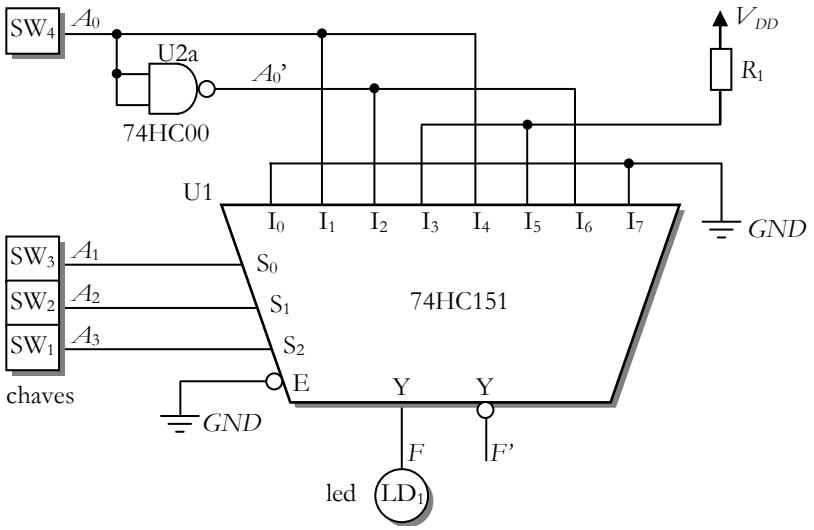


Figura 6.3 Implementação reduzida da função F com um multiplexador 8:1

Seguindo esse raciocínio, completamos a Tabela 6.3 e o correspondente mapa K. O circuito da Figura 6.3 implementa a função F usando um mux 8:1. Desta forma, podemos implementar o circuito utilizando apenas um mux '151 e um inversor. No lugar do inversor, usamos uma porta NAND (74HC00), como aplicação da propriedade de *Suficiência de NAND* – isso nos permite ter menos componentes em estoque (economia...).

A Figura 6.3 exemplifica mais algumas boas normas de documentação:

- NÃO se desenha o mux como um chip '151 (isto é, pinos de 1 a 8 embaixo e de 16 a 9 em cima). A ideia não é representar a construção física do circuito, mas sim a sua lógica de funcionamento. Assim, procure desenhar entradas à esquerda ou em cima, e saídas à direita ou embaixo.
- Dentro do símbolo lógico do mux, indicam-se os nomes genéricos dos pinos, seguindo a nomenclatura sugerida no *datasheet*.
- Do lado de fora, indicam-se os nomes dos sinais do circuito.
- Use nomes NEGADOS para sinais ATIVOS EM ZERO.
- Costuma-se destacar entradas e saídas ativas em zero nos símbolos lógicos desenhando-se inversores (bolinhas) junto aos pinos. É o caso da entrada no pino 7: no símbolo, o nome do sinal está do lado de dentro *após* o

inversor e por isso aparece como *E* (sem negação) – do lado de fora, antes do inversor o nome do sinal seria *E*' (negado). Analogamente, o pino 6 tem um inversor para indicar que é uma saída ativa em zero (*F*').

Nota: no diagrama lógico, incluímos chaves (símbolos quadrados), um led (símbolo redondo) para ilustrar como o circuito poderia ser montado no *protoboard* do laboratório. Adote essa prática quando for fazer os diagramas pedidos no pré-relatório e no relatório.

6.7 Decodificador

Como seu nome indica, a função desse circuito é decodificar um endereço em binário de *n* bits, ativando um e apenas um dentre os dispositivos conectados a suas 2^n saídas (veja a figura 5-11 do livro-texto).

O integrado '138 é um decodificador com três bits de endereço ($n = 3$) e oito saídas. Observe o *datasheet* em anexo. Trata-se do 74LS138 (TTL), mas funcionamento e a pinagem é compatível com o 74HC138 (CMOS) que usaremos no laboratório.

As entradas A_2, A_1 e A_0 constituem um endereço de três bits, onde A_2 é o bit mais significativo, que permite selecionar uma das oito saídas, O_0' a O_7'. Note ainda que as saídas são negadas: por exemplo, se tivermos $A_2A_1A_0 = L\ H\ H$ (isto é, 011), a saída O_3' irá a *L* e as demais ficarão em *H*.

Existem três entradas para habilitar o circuito, E_1', E_2' e E_3, tal que a habilitação (*ENABLE*) ocorre com

$$ENABLE = \overline{E_1'}.\overline{E_2'}.E_3 . \qquad (6.8)$$

Como E_1' e E_2' são *ativos em zero*, teremos *ENABLE* = *H* quando E_1' = *L*, E_2' = *L* e E_3 = *H* (lembre-se: E_1' e E_2' são os *nomes* das entradas, que são *ativas em zero* e por isso estão negadas na expressão acima). Caso *ENABLE* seja igual a *L*, todas as saídas ficarão desativadas e mantidas em nível *H*, independentemente do valor de $A_2A_1A_0$. Essas três entradas facilitam o *cascateamento* desse componente, isto é, quando se deseja construir um decodificador de 16 saídas a partir de dois CIs '138, por exemplo.

6.8 Barramento *tri-state*

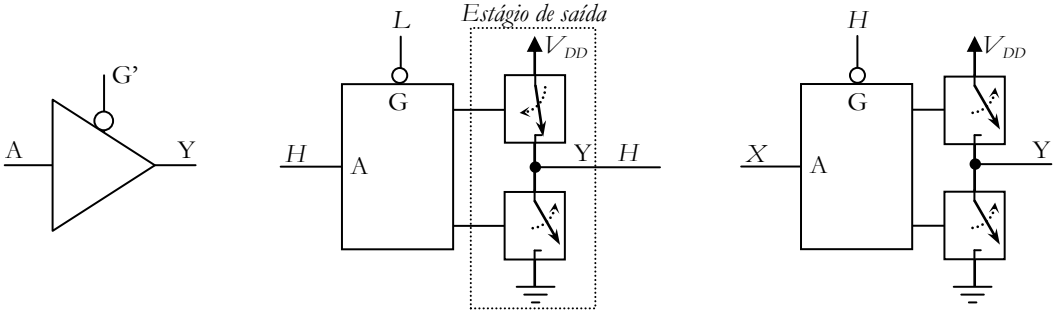
Um barramento (ou *bus*) nada mais é do que um único condutor elétrico no qual são conectadas as saídas de vários dispositivos digitais. No entanto, a cada instante apenas uma saída deve estar eletricamente ligada ao barramento enquanto que as demais devem ser mantidas em alta impedância, caso contrário pode ocorrer um curto-circuito entre uma saída em nível *H* (5 V) e outra em nível *L* (0 V).

Portas lógicas capazes de desconectar eletricamente seu estágio de saída são conhecidas como *tri-state*. Ao lado dos dois estados já conhecidos, *L* e *H*, o terceiro estado indicado pelo nome é chamado de *alta impedância* (ou *high-Z*), isto é, eletricamente isolado.

6.8.1 Seguidor *tri-state* '125

Um exemplo de circuito com saída *tri-state* é o seguidor '125, também conhecido como *buffer tri-state*. No laboratório, usaremos o 74HC125 que contém quatro portas *tri-state* com sinais de ativação individuais (*datasheet* em anexo).

A Figura 6.4a mostra o símbolo lógico do seguidor. Como o nome indica, o seguidor copia o nível lógico da entrada *A* na saída *Y*. O estado de alta impedância de saída é comandado pelo sinal de habilitação *G'*. Repare que o sinal é *ativo em zero* e por isso seu nome é *G'* (*G* negado). A saída está habilitada quando *G'* = *L*, e está em alta impedância quando *G'* = *H*. Mais uma vez, atenção aos detalhes: o sinal se chama *G'* **antes** do inversor na figura *a*, e o mesmo sinal se chama *G* **depois** do inversor nas figuras *b* e *c*.



a) Símbolo lógico b) Saída ativa em nível *H* c) Saída em alta impedância

Figura 6.4 Seguidor *tri-state* '125

O estágio de saída do seguidor é composto por dois *transistores* em série, numa configuração conhecida como *totem-pole*. A Figura 6.4b mostra, por exemplo, o estágio de saída ativo (*G'* = *L*) e em nível *H*: nesta situação, o

transistor inferior se encontra aberto e o superior está fechado para conectar a saída *Y* a V_{DD}. Por fim, a Figura 6.4c mostra o que acontece quando o estágio de saída é desabilitado (*G'* = *H*): os dois transistores se abrem, deixando a saída *Y* em alta impedância.

O diagrama elétrico do circuito se encontra no *datasheet* ("Circuit Diagram", figura da esquerda) e temos que admitir que é um tanto difícil entender como funciona. Mas repare que o estágio de saída (no extremo direito do circuito) é composto por dois transistores MOSFET, sendo o superior canal P e o inferior canal N, constituindo um *inversor CMOS*.

6.8.2 Exemplo: barramento de um bit

O circuito da Figura 6.5 mostra um barramento *F* de um bit com oito sinais conectados, de D_0 a D_7.

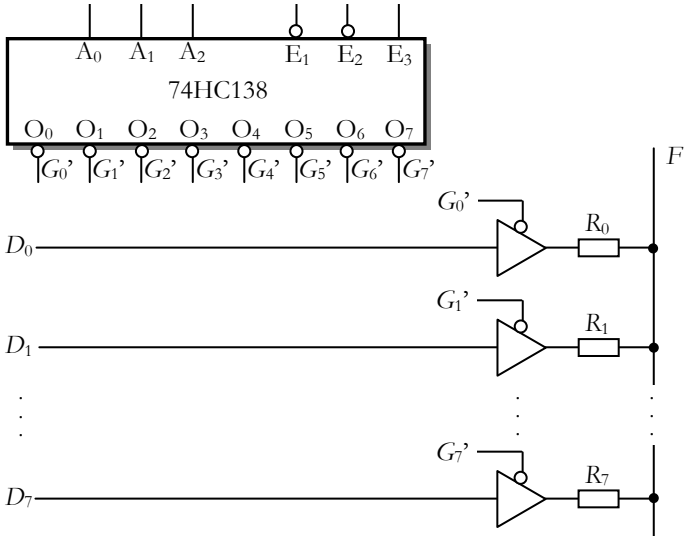


Figura 6.5 Barramento de 1 bit

O decodificador 74HC138 garante que a cada instante apenas um dos seguidores *tri-state* tenha sua saída habilitada e possa impor uma tensão no barramento *F*. Note que isto constitui um mux simplificado.

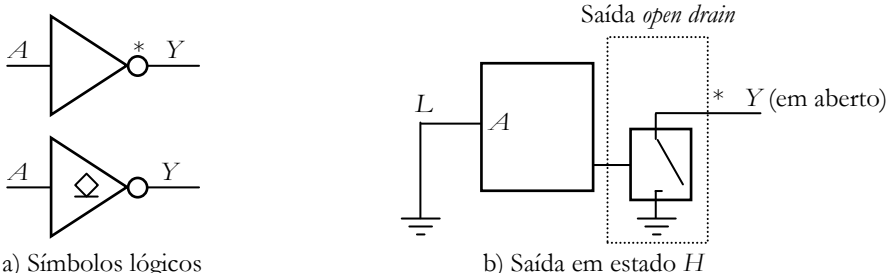
Os resistores R_i são colocados em série com as saídas dos seguidores por segurança: servem para limitar a corrente caso duas ou mais saídas venham a ser ativadas em níveis lógicos diferentes por algum problema. Em geral empregam-se resistores de baixo valor, entre 100 e 470 **Ω**.

6.9 Saídas *open drain*

Há circuitos digitais em que o estágio de saída não apresenta os dois transistores em *totem-pole* descritos na seção anterior, mas apenas o transistor inferior. Ou seja, é possível levar a saída ao nível LOW fechando-se a chave ligada ao terra, mas devido à ausência do transistor superior não é possível fazer a saída ir a HIGH por si só, sem usar componentes externos.

No caso dos circuitos CMOS, esse tipo de saída é chamado de **dreno aberto** ou ***open drain***. Nota: é comum encontrar também o termo *coletor aberto* (ou *open collector*), mas esse nome é mais apropriado para circuitos fabricados com uma tecnologia mais atinga, conhecida como TTL (veremos isso com mais detalhes quando estudarmos os diferentes tipos de transistores).

Como exemplo, veja o circuito do 74HC05 no *datasheet* em anexo. Esse componente contém seis portas inversoras *open drain*. A segunda página do *datasheet* contém o diagrama elétrico da porta. A Figura 6.9a mostra o símbolo lógico do inversor '05 e as duas formas usadas para se indicar uma saída *open drain*: o sinal '*' ou o losango com o traço inferior.



a) Símbolos lógicos b) Saída em estado *H*

Figura 6.6 Inversor *open drain* '05

Na situação mostrada na Figura 6.9b, o inversor tem sua entrada *A* em nível *L* e tenta impor um nível *H* na saída abrindo o transistor inferior, mas a falta do transistor superior faz com que saída *Y* permaneça em aberto (ou seja, em *alta impedância*). Por outro lado, note que o transistor inferior ainda permite conectar a saída a 0 V quando ela estiver em nível lógico *L*.

6.9.1 Conversão de nível de tensão

A ausência do transistor superior nos permite conectar o pino de saída a fontes de tensão com níveis diferentes de V_{DD}. É o que mostra o circuito da Figura 6.7. Quando a saída está em nível lógico *H*, o transistor inferior do estágio de saída se mantém aberto, e o sinal *Y* se conecta a tensão V_{BUS} pelo resistor *R*, conhecido como resistor de *pull-up*. Mais adiante falaremos mais sobre esse resistor.

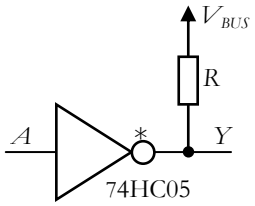


Figura 6.7 Saída coletor aberto com resistor de *pull-up*

6.9.2 Circuito *wired-and*

É possível implementar uma operação AND entre saídas *open collector* (TTL) ou *open drain* (CMOS) simplesmente ligando-as em curto-circuito e com um resistor de *pull-up*. É a chamada lógica *wired-and*. A Figura 6.8 mostra um exemplo de lógica *wired-and* entre três portas 74HC com saída *open drain*.

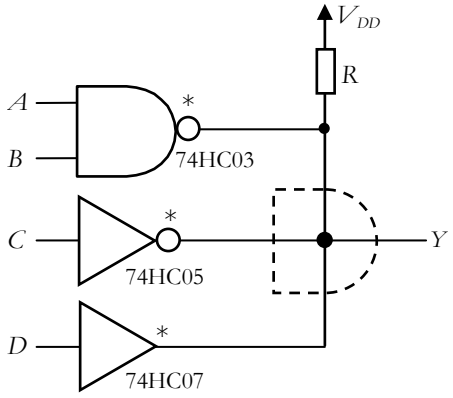


Figura 6.8 Exemplo de lógica w*ired-and*.

O sinal *Y* é um *and* entre as saídas das três portas (a porta AND pontilhada não existe de fato, mas é dessa forma que o *wired-and* costuma ser representado em diagramas lógicos). Em termos lógicos, temos

$$Y = \overline{(A.B)}.\overline{C}.D \ . \tag{6.9}$$

O sinal de saída *Y* ficará em *L* quando uma ou mais das três saídas estiver em nível *L*. Por outro lado, quando todas estiverem em *H*, todas estarão em alta impedância e a via *Y* ficará carregada eletricamente com V_{DD} através do resistor *R*.

6.10 Barramento *wired-and* tolerante a colisões

Dizemos que ocorre uma *colisão* quando dois ou mais dispositivo impõem tensões diferentes a um barramento ao mesmo tempo, fazendo com que as mensagens se embaralharem.

Para evitar que colisões ocorram, é necessário que haja algum esquema de *arbitragem* que defina qual dispositivo pode acessar o meio de transmissão a cada instante. O mais simples consiste em ter um gerenciador central ao qual todos os demais devem solicitar permissão antes de transmitir, e é o mais usado nos casos em que o barramento está contido em um único equipamento e conecta poucos dispositivos. É o que acontece dentro de um computador, por exemplo.

No entanto, um barramento pode conectar vários dispositivos dispersos em uma grande área, formando uma rede de comunicação. Nesses casos, ter um nó centralizando a arbitragem não é uma boa ideia. Seria necessário ter um canal de comunicação individual entre o nó central e cada um dos outros dispositivo, o que aumentaria o custo de instalação da rede. Além disso, toda a rede estaria comprometida se apenas o nó central deixar de funcionar.

A solução é permitir o acesso ao meio físico a qualquer nó que tenha uma mensagem para transmitir. Cada nó da rede deve monitorar todas as mensagens que trafegam pelo meio (mesmo que não sejam destinadas a ele) e iniciar a transmissão somente quando o meio estiver livre. O problema é que colisões podem ocorrer se por acaso mais de um nó começar a transmitir praticamente ao mesmo tempo.

6.10.1 Alocação destrutiva e não destrutiva

Quem escreve uma mensagem deve ao mesmo tempo monitorar a via. Quando percebe que o sinal lido é diferente daquele que tentou impor, terá detectado uma colisão e saberá também que a sua mensagem se perdeu. Nesse caso, deverá aguardar um intervalo aleatório de tempo antes de tentar transmitir novamente.

Há duas formas básicas de tratamento de colisão. A mais simples é conhecida como *alocação destrutiva*, em que as mensagens que colidem se perdem, sem que aja predominância de uma sobre as outras. É a mais usada em redes de computadores, por exemplo.

No entanto, em sistemas onde mensagens críticas devem ser transmitidas com o menor atraso possível, é necessário empregar uma forma de *alocação não destrutiva*, na qual a mensagem de maior prioridade se mantém íntegra após uma colisão e se sobrepõe às demais.

6.10.2 Barramento *wired-and*

Como vimos, colisões devem ser evitadas em um barramento *tri-state* para que saídas em níveis de tensão diferentes não sejam colocadas em curto-circuito.

Uma alternativa é empregar saídas *open drain*. Pode-se construir um barramento eletricamente tolerante a colisões interligando-se saídas *open drain* em *wired-and*, como mostra a Figura 6.9. A saída de cada nó é capaz de impor o nível *L* à via (ao fechar o transistor de saída), mas não o nível *H*. A via somente ficará em nível *H* se as saídas de todos os nós estiverem abertas (em alta impedância), como mostra a figura. Nessa condição a tensão na via será ser imposta pela fonte V_{DD} através do resistor *R* de *pull-up*.

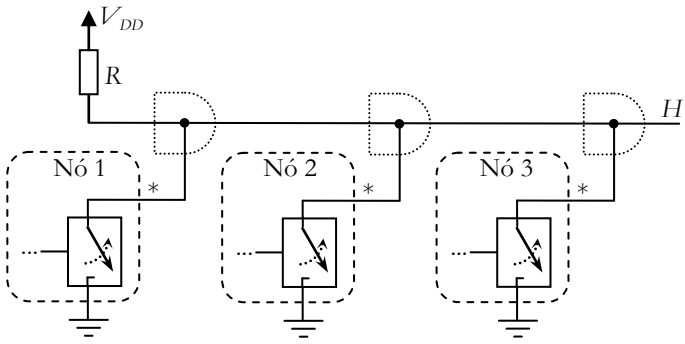


Figura 6.9 Barramento *wired-and* em nível *H*.

6.10.3 Aplicação: rede com alocação não destrutiva

A comunicação serial é largamente empregada devido ao baixo custo de instalação e de manutenção. Na comunicação serial, o envio de mensagens ocorre bit a bit, um bit de cada vez. Para reduzir custos, todos os nós da rede são conectados aos mesmos fios (conexão *multidrop*). Desta forma, um nó é capaz de enviar uma mensagem para todos os outros através do mesmo meio físico.

Uma rede muito empregada na indústria e em veículos é a *Controller Area Network* (*CAN*). Nessa rede toda mensagem começa com um ID, do bit mais significativos ao menos significativo. Por exemplo, supondo que o ID de uma mensagem tenha sete bits e seja igual a 0100101 (37 em decimal), estes bits serão enviados através do meio físico na seguinte ordem: 0-1-0-0-1-0-1. O tempo de duração de cada bit é conhecido por todos os nós da rede.

Quando um nó tentar impor o nível lógico *H* (*recessivo*) no meio físico e ler o nível lógico *L* (*dominante*), deverá perceber que outro nó está enviando uma mensagem de prioridade maior. Neste caso, o nó de menor prioridade deverá abortar a transmissão e manter sua saída em nível *H*. Por exemplo, em um carro, o ID de mensagens enviadas por sensores de travamento das rodas para a unidade de controle do sistema ABS deve ter maior prioridade do que o ID de mensagens para acionamento de pisca-pisca.

Em repouso, quando não há mensagens sendo transmitidas, o meio permanece no nível recessivo *H*. O início de qualquer mensagem é marcado por um bit obrigatoriamente em nível *L*, conhecido como *start-bit*. Na sequência, transmitem-se os bits da mensagem; por fim, o meio deve retornar ao nível *H* e permanecer assim por um tempo mínimo (conhecido como *stop-bit*) como preparativo para o *start-bit* da próxima mensagem.

Na Figura 6.10 mostra um exemplo de arbitragem entre três nós. Neste exemplo, os nós A, B e C começam a enviar mensagens ao mesmo tempo em $t = 0$, começando pelo *start-bit*. Os identificadores das mensagens são:

- Nó A: ID = 0101001

- Nó B: ID = 0011100
- Nó C: ID = 0011011

No instante *t* = 2, o nó A tenta escrever o nível lógico *H* e lê um valor diferente no meio de comunicação. Com isso, o nó A aborta a transmissão e passa ao modo de escuta. O mesmo ocorre com o nó B no instante *t* = 5, e a partir daí apenas o nó C continua transmitindo. Ao final da mensagem em t = 8 o meio retorna ao nível recessivo *H* e permanece pelo tempo definido como *stop-bit*.

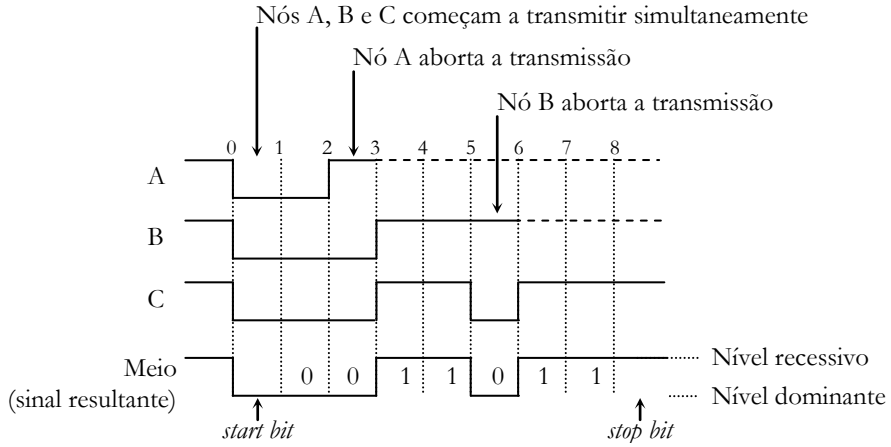


Figura 6.10 Arbitragem entre três mensagens com alocação não destrutiva

Note que apesar de haver mais de uma mensagem trafegando na via até *t* = 5, os bits da mensagem de maior prioridade (do nó C) foram preservados do início ao fim da transmissão.

6.10.4 Implementação de um nó da rede

No laboratório, vamos simular a alocação não destrutiva usando portas com saída *open drain* para representar os nós da rede. O meio físico de comunicação será constituído por um par de fios: um condutor ligado ao terra comum a todos os nós e um condutor que interliga as saídas formando conexões *wired-and*.

A Figura 6.11 mostra o esquema do transmissor que representa um nó da rede. A entrada *DI* fornece o bit a ser transmitido. A entrada *TxE* (*Tx Enable*) habilita o transmissor quando em nível *H* e faz a saída *DO* seguir a entrada *DI*. Como a porta OR é *open drain*, a saída *DO* fica em alta impedância com *TxE* em *L*.

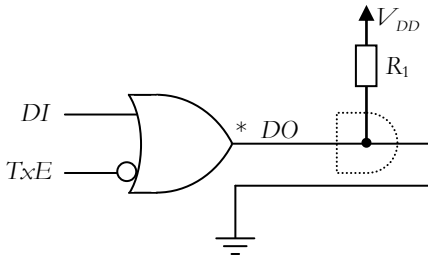


Figura 6.11 Transmissor *open drain*.

Como infelizmente não dispomos de uma porta OR *open drain* no laboratório (uma típica *mosca branca*), teremos que implementar os nós da rede com outros componentes. Por exemplo, o circuito mostrado na Figura 6.12 pode ser montado com: 74HC00 (quad NAND), 74HC08 (quad AND) e 74HC05 (hex inverter open drain).

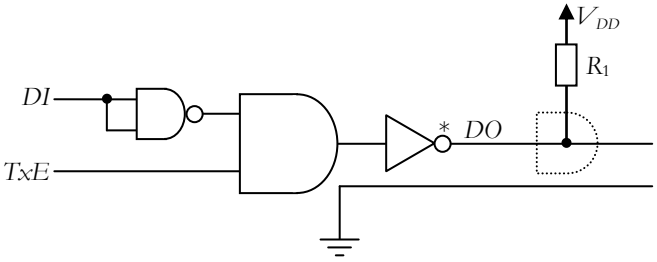


Figura 6.12 Implementação com portas disponíveis no laboratório.

Nota: em redes reais, a coisa é mais complexa – usam-se transformadores para conectar os nós (isolação galvânica) e uma codificação especial para evitar que o meio permaneça constante por muito tempo (*bit stuffing*).

6.11 O resistor de *pull-up*

Nos circuitos com saídas *open drain*, o resistor de *pull-up* é necessário pois é através dele que a saída se conecta à tensão HIGH. Precisamos então de algum critério para dimensionar o valor do resistor de *pull-up*.

Quando uma saída *open drain* está em nível *L*, o transistor de saída está conduzindo, como ilustra a Figura 6.13a. Nesse caso, o resistor serve para limitar a corrente que passa pelo transistor. Nos *datasheets*, é possível encontrar a corrente máxima que uma saída pode drenar quando em nível *L*, e essa corrente costuma ser representada por I_{OL}. Assim, temos um limite inferior para o resistor *R*, dado por

$$R > \frac{V_{DD}}{I_{OL}} .$$

Por exemplo, os componentes CMOS que usamos no laboratório tem I_{OL} da ordem de 20 mA. Com V_{DD} nominal igual a 5 V, temos $R > 250\ \Omega$. No entanto, esse valor de resistência é muito baixo uma vez que faria a fonte V_{DD} fornecer continuamente a corrente I_{OL} apenas para manter a tensão de saída próxima a zero.

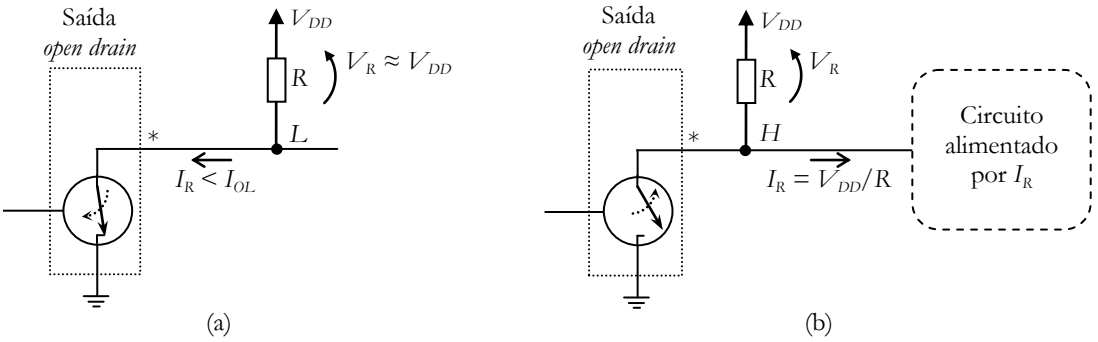


Figura 6.13 Corrente e tensão no resistor de *pull-up* com a saída *open drain*: a) em nível *L*; b) em nível *H*.

Para minimizar esse desperdício de energia, usa-se um resistor de valor bem maior. Por outro lado, o resistor limita a corrente que pode ser fornecida pela saída quando em nível *H*, como mostra a Figura 6.13b. É preciso definir também qual é a máxima queda de tensão V_R admissível sobre o resistor *R*.

Por exemplo, se a saída estiver ligada a entradas de portas lógicas CMOS, temos que ter $V_R < V_{DD}/3$ para que o nível lógico *H* se mantenha (como vimos na experiência anterior). Porém, entradas de portas CMOS drenam correntes muito baixas – da ordem de nano-Ampères, e com isso o valor máximo do resistor seria da ordem de megaOhms. No laboratório, usaremos como pull-up resistores na faixa de 1 kΩ a 100 kΩ.

6.12 Pré-Relatório e Relatório

A Parte B da apostila contém dois tipos de itens que você deverá responder:

- **Exercícios**: constituem o *pré-relatório;* podem ser feitos antes da aula, mas recomendamos que sejam feitos com antecedência para que se possa aproveitar melhor o tempo no laboratório.
- **Anotações**: devem ser feitas individualmente *durante* a aula e constituem o *relatório*.

ATENÇÃO: leia as atividades da PARTE B e não apenas os enunciados dos exercícios do pré-relatório

Muitos detalhes necessários para fazer os exercícios estão descritos nas atividades em que se inserem. Além disso, você já terá uma noção do que deverá fazer e perderá menos tempo com a leitura durante a aula.