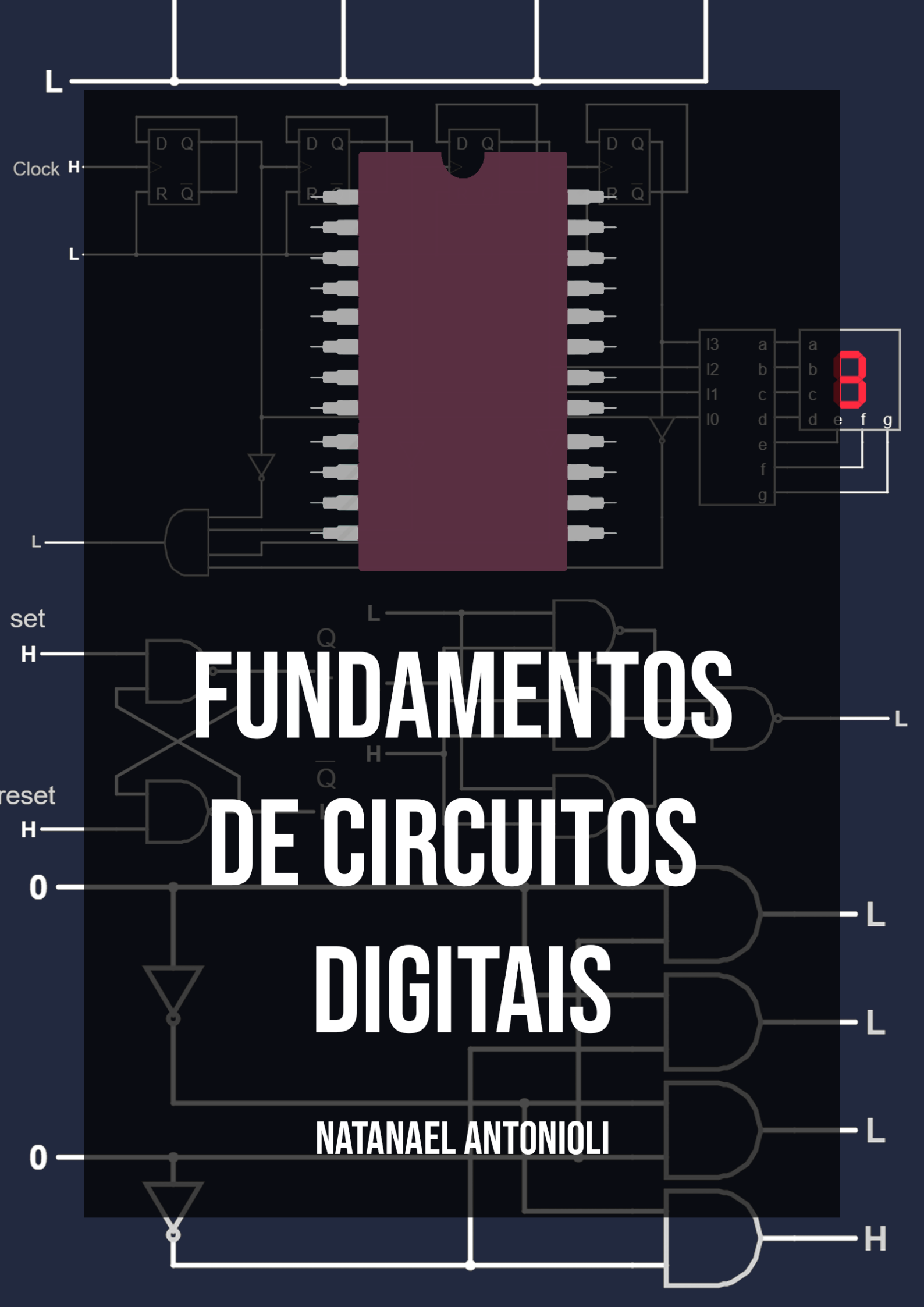


FUNDAMENTOS DE CIRCUITOS DIGITAIS

NATANAEL ANTONIOLI

1 SUMÁRIO

2 PREFÁCIO

Olá, estudante ou interessado em computação. Este é um livro – ou melhor, uma apostila – sobre circuitos digitais. Circuitos digitais estão presentes em praticamente qualquer dispositivo tecnológico: celulares, calculadoras, computadores, carros, portões automáticos, micro-ondas, e assim por diante.

Começamos compreendendo como os transistores – dispositivos responsáveis pela revolução tecnológica – funcionam, que como eles podem ser usados para operações lógicas simples. Descobriremos que transistores podem ser compactados em pequenos chips, e que portas lógicas podem ser vistas como "caixas pretas" constituintes de vários outros circuitos.

A partir daí, verificamos que existem modelos matemáticos que descrevem circuitos lógicos, e que estes podem ser minimizados, obtendo-se uma implementação de menor custo e aquecimento.

Veremos também que, por meio da retroalimentação, podemos criar elementos de memória rudimentares, que compõe os circuitos sequenciais.

Espero que esta apostila lhe seja útil em sua jornada na computação. Bons estudos!

3 NOÇÕES DE CIRCUITOS ELÉTRICOS

3.1 COMPONENTES BÁSICOS DE CIRCUITOS ELÉTRICOS

Circuitos digitais são circuitos eletrônicos que empregam a utilização de sinais elétricos em apenas dois níveis de tensão para definir dois valores binários, 0 e 1. Porém, antes de entendermos como os circuitos digitais funcionam, precisamos, antes, entender como seus componentes mais fundamentais funcionam.

Em um circuito, uma fonte de alimentação é qualquer dispositivo que seja capaz de gerar uma diferença de potencial elétrico entre seus terminais. Exemplos de fontes de tensão são as baterias, pilhas, fontes de corrente contínua, entre outros dispositivos semelhantes.

Figura 1: exemplos de fontes de alimentação.

Tais dispositivos possuem dois terminais: um deles (normalmente o vermelho) é carregado positivamente (polo positivo), possuindo falta de elétrons, e o outro é carregado negativamente (normalmente o preto) possuindo excesso de elétrons (polo positivo). A essa diferença de carga entre os polos denominamos diferença de potencial, a qual é medida em Volts (V).

Alternativamente, uma fonte de alimentação pode ser representada pelo seguinte símbolo, no qual a barra maior indica o polo positivo, e a barra menor indica o polo negativo:

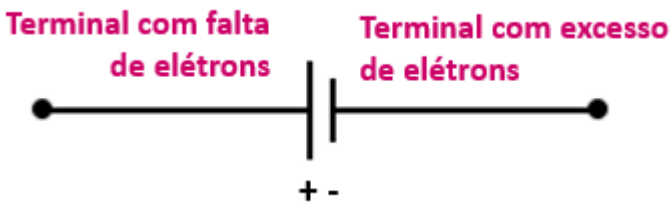


Figura 2: símbolo utilizado para representar uma fonte de alimentação, bem como suas correspondências.

Quando os dois terminais são colocados em contato, a tendência natural é que o terminal que possui falta de elétrons comece a "puxar" elétrons do terminal os possui em excesso. Assim, ocorre um fluxo de elétrons que sai do terminal negativo e vai em direção ao terminal positivo.

Historicamente, a corrente elétrica foi definida como sendo o fluxo de cargas positivas. Porém, elétrons têm cargas negativas e, por essa razão, dizemos que a corrente elétrica possui o sentido oposto do fluxo de elétrons que ocorre em um circuito.

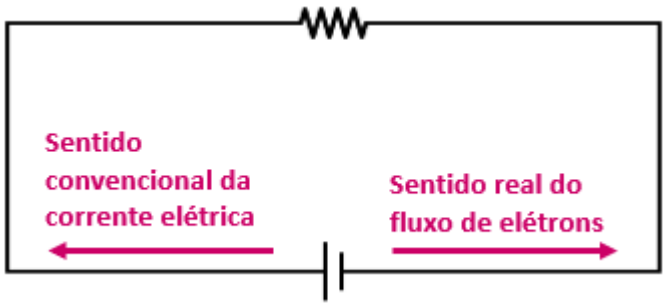


Figura 3: sentido real do fluxo de elétrons versus sentido convencional da corrente elétrica.

Chamamos de resistência elétrica de um corpo a capacidade desse corpo de se opor a passagem de corrente elétrica mesmo quando há uma diferença de potencial aplicada. Essa grandeza é medida em Ohms (Ω).

Qualquer material apresenta resistência elétrica, incluindo os fios, mas esse valor costuma ser muito baixo. Por essa razão, costumamos assumir que os fios utilizados nas simulações de circuitos são ideais, ou seja, não possuem resistência elétrica.

A corrente elétrica é medida em Amperes (A), e sua grandeza corresponde à Columbs por segundo ($^{C}/_{S}$). Columb, por sua vez, é uma medida de carga elétrica – a carga de um elétron equivale a aproximadamente $1{,}6021 \times 10^{-19}$ C. Ou seja, aferir a corrente elétrica em um circuito equivale a aferir a quantidade de elétrons que passam a cada segundo por aquela região.

Para componentes elétricos que obedecem à lei de Ohm, a corrente que passa por um componente pode ser obtida pela fórmula $I = \frac{U}{R}$, em que U é a diferença de potencial no componente e R é a resistência do componente.

Conforme você pode ter percebido, a corrente elétrica tende ao infinito quando a resistência no componente tende a zero (é muito baixa), uma vez que $\lim_{R \to 0} \frac{U}{R} = +\infty$. Uma corrente muito alta definitivamente pode danificar ou inutilizar um componente.

Para evitar que isso ocorra, existem os resistores. Resistores são componentes que têm por finalidade oferecer uma oposição à passagem de corrente elétrica. Na prática, eles podem ser usados para transformar energia elétrica em outra forma de energia (como térmica) ou para reduzir o valor da corrente em um circuito, garantindo que os demais componentes operem em uma faixa segura.

Figura 4: alguns resistores

O resistor é indicado no circuito pelo seguinte símbolo:

Figura 5: símbolo do resistor

Dizemos que dois ou mais resistores estão ligados em série quando eles estão associados em sequência. Nesse caso, podemos considera-los em função da resistência equivalente, que é dada pela soma de todas as resistências. Assim:

$$R_{eq} = \sum_{i=1}^{n} R_i = R_1 + R_2 + \cdots + R_i$$

Essa propriedade significa que podemos "transformar" uma associação de resistores em série em um único resistor. Por exemplo:

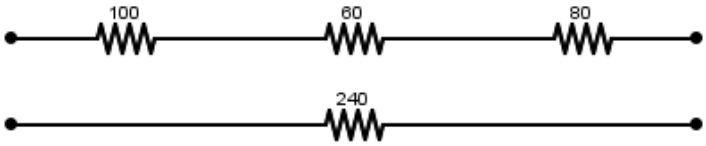


Figura 6: associação de três resistores em série, com resistência equivalente de $\mathbf{100\Omega + 60\Omega + 80\Omega = 240\,\Omega}$

Dizemos que dois ou mais resistores estão em paralelo quando todos os resistores estão submetidos a uma mesma diferença de potencial. Nesse caso, podemos considera-los em função da resistência equivalente, cujo inverso é dado pela soma dos inversos de todas as resistências. Assim:

$$\frac{1}{R_{eq}} = \sum_{i=1}^{n} \frac{1}{R_i} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + 1/R_n$$

Essa propriedade significa que podemos "transformar" uma associação de resistores em paralelo em um único resistor. Por exemplo:

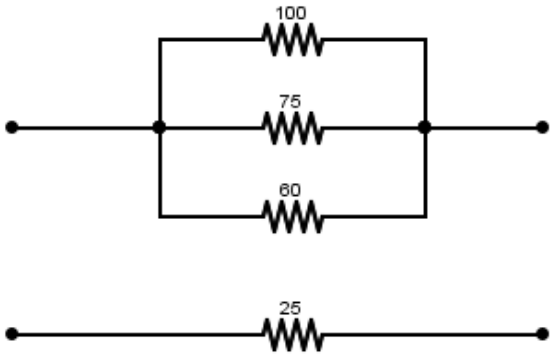


Figura 7: associação de três resistores em paralelo, com resistência equivalente de

$$\left(\frac{1}{100\Omega}\right) + \left(\frac{1}{75\Omega}\right) + \left(\frac{1}{60\Omega}\right)^{-1} = 25\,\Omega$$

Existe, ainda, outro elemento bastante comum em circuitos do tipo. Trata-se da chave, que tem a função de permitir ou bloquear a passagem de corrente em uma região. Existem inúmeros modelos de chaves de acionamento manual, como você pode ver abaixo:

Figura 8: alguns tipos de chaves.

Nos circuitos, as chaves são representadas pelo seguinte símbolo, que podem estar na posição fechada (permitindo corrente) e aberta (bloqueando corrente).

Figura 9: símbolo da chave aberta e fechada no circuito.

Visando obter um indicador visual dá passagem de corrente por uma dada região, podemos conectar uma lâmpada ou um LED naquele trecho do circuito. Esses componentes agem como um tipo particular de resistor, convertendo energia elétrica em energia luminosa.

Figura 10: LED azul e lâmpada.

Nos circuitos, lâmpadas e LEDs podem ser representados pelos seguintes símbolos:

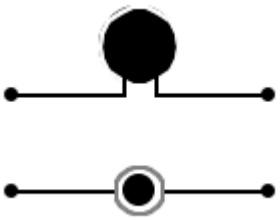


Figura 11: símbolos para lâmpada e LED.

Por fim, as leis de Kirchhoff são muito úteis para a compreensão das grandezas envolvidas em um circuito. Mais especificamente para o nosso caso, a primeira lei de Kirchhoff afirma que a soma das correntes que chegam a um nó é igual a soma das correntes que saem desse nó.

3.2 TRABALHANDO COM CIRCUITOS ELÉTRICOS NA PRÁTICA

Existem três formas de se estudar circuitos elétricos. A primeira delas é na forma completamente teórica, na qual o circuito deve ser desenhado no papel e compreendido através de cálculos.

Porém, eletricidade é uma das áreas do conhecimento que podem ser melhor compreendidas através de estudos mais práticos, seja por meio de simulações, ou por meio de experiências com componentes reais.

A segunda forma de estudo é por meio de softwares de simulação. Um dos mais intuitivos programas do ramo é o Falstad Circuit Simulator, aplicação baseada em Java que pode ser baixada gratuitamente em https://www.falstad.com/circuit/.

Por fim, a terceira forma envolve trabalhar com os componentes reais e com instrumentos de medida. Para tanto, mais alguns conceitos precisam ser apresentados de forma a lhe capacitar para trabalhar com tais equipamentos. Os próximos parágrafos abordarão diversos equipamentos que podem ser usados para realizar e aferir ligações físicas entre os componentes que já conhecemos (fontes de alimentação, resistores, chaves, lâmpadas e LEDs).

A protoboard é o local mais adequado para montar um protótipo do seu circuito, uma vez que ela permite encaixar e desencaixar componentes conforme necessário, possibilitando a reutilização dos mesmos e eliminando a necessidade de soldá-los. Uma protoboard tem a seguinte aparência:

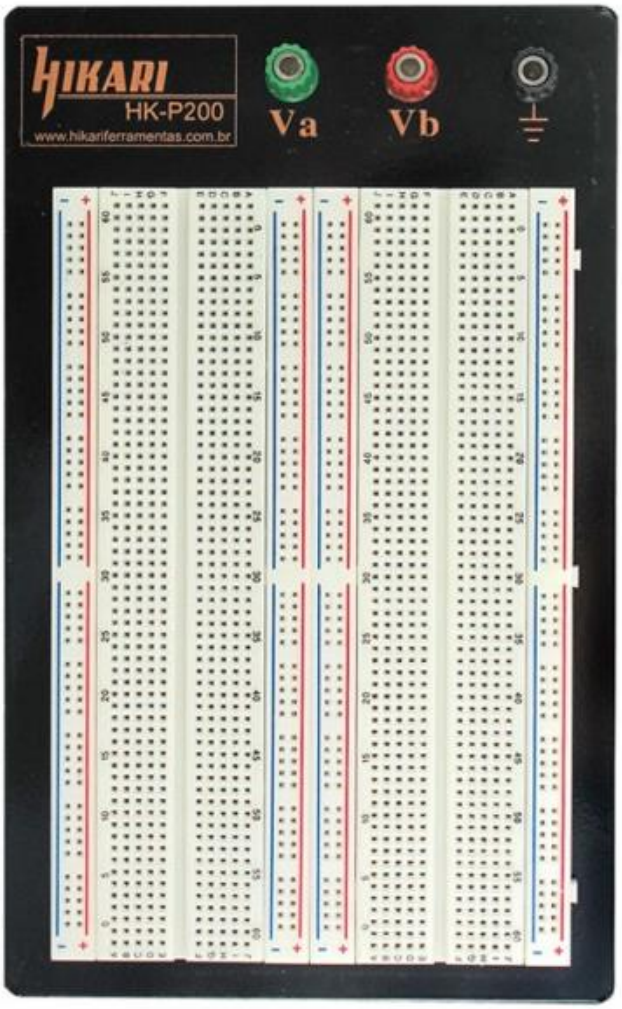


Figura 12: exemplo de protoboard

O funcionamento de uma protoboard é relativamente simples, mas ele só pode ser efetivamente dominado após alguma prática. Uma portoboard possui três áreas: a área de distribuição de alimentação, correspondente as linhas horizontais, a área de montagem de circuitos integrados, correspondente ao meio da protoboard, e a área para montagem de componentes, correspondentes as linhas verticais. A imagem abaixo apresenta a ligação que ocorre nessas áreas: toda a área de alimentação está em curto, ao passo que cada uma das linhas verticais também estão individualmente em curto.

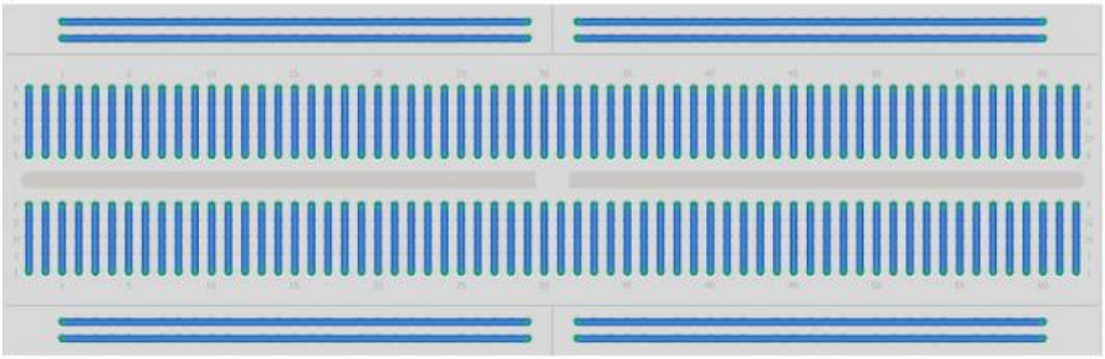


Figura 13: esquema de curtos em uma protoboard.

Uma protoboard pode ser alimentada de diversas formas. A forma que garante uma alimentação mais precisa é por meio de uma fonte de alimentação com corrente contínua ajustável. Porém, tais fontes costumam ser caras, e essa não é uma alternativa viável para iniciantes.

A forma mais simples de executar essa alimentação é através de uma fonte de protoboard, como a presente em http://www.baudaeletronica.com.br/fonte-ajustavel-para-protoboard.html.

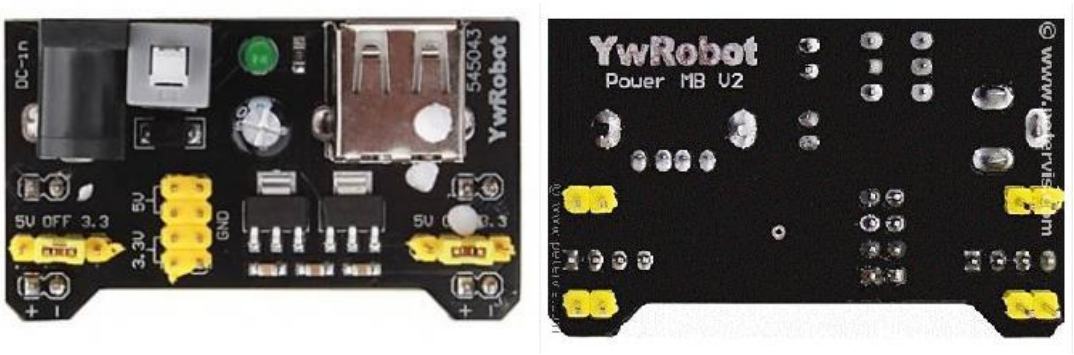


Figura 14: fonte ajustável para Protoboard.

‘ A fonte pode ser configurada para fornecer 5 V ou 3,3 V, sendo alimentada a partir de um cabo USB ou de energia. Ela também conta com quatro pinos inferiores, que permitem que ela seja encaixada na protoboard. Ao fixar esses quatro pinos nas regiões de alimentação, estamos fornecendo 5 V na região vermelha e 0 V na região azul, respectivamente.

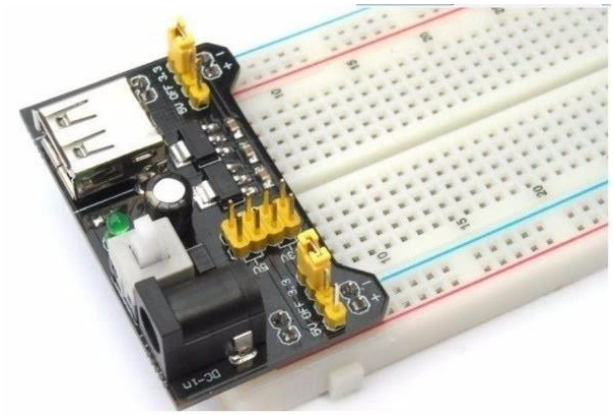


Figura 15: fonte ajustável para protoboard já fixada.

Ao trabalhar com componentes eletrônicos, muitas vezes se faz necessário realizar medidas de grandezas associadas ao circuito, como tensão, corrente e resistência. Para essas situações, é possível utilizar um multímetro. Multímetros vêm com diversos instrumentos implementados para medição das grandezas fundamentais e, em modelos mais avançados, para medição de grandezas como capacitância, temperatura, ganho de transistor, indutância etc.

Figura 16: multímetro digital.

A utilização de um multímetro é bastante simples, e será descrita no nível necessário para este livro nos próximos parágrafos. Todo multímetro possui também dois cabos, que devem ser conectados nas saídas na parte inferior do equipamento.

O cabo preto sempre deve ser conectado na saída marcada com COM. Para medidas de resistência e diferença de potencial, deve-se conectar o cabo vermelho na saída marcada com VΩHz e, para medidas de baixa corrente, deve-se conectar o cabo vermelho na saída marcada com Ma.

Uma vez feitas as conexões, pode-se usar a chave seletora para escolher o tipo de grandeza que se irá medir e a escala. A escala do multímetro corresponde ao maior valor que o equipamento é capaz de medir naquela posição. Visando minimizar incertezas, recomenda-se que sempre seja utilizada a menor escala possível para cada grandeza.

Pode-se medir resistência com o multímetro posicionando os terminais vermelho e preto nos extremos do componente, ou do trecho do circuito que se deseja aferir.

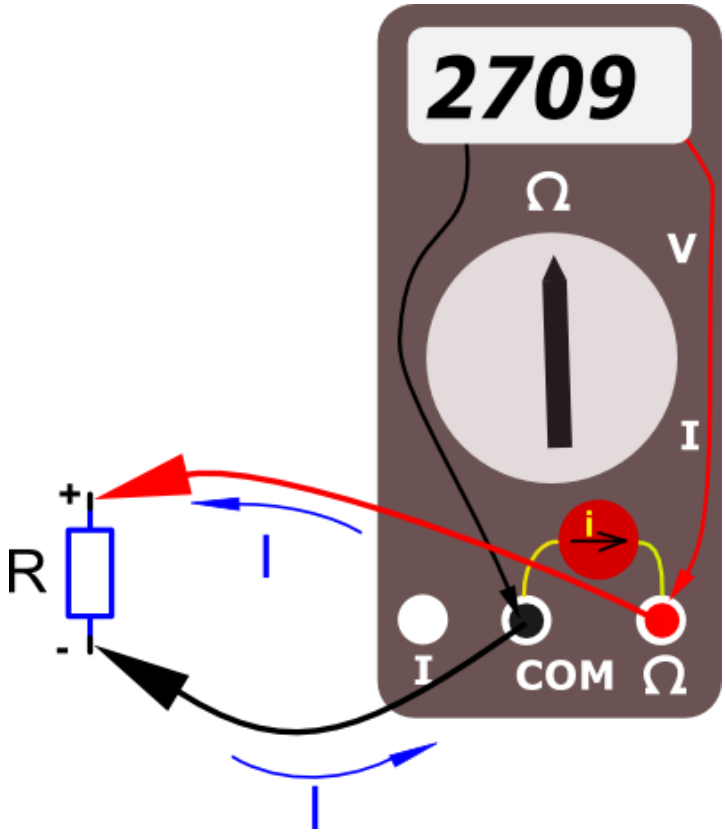


Figura 17: uso de um multímetro para se aferir resistência.

Já para se aferir diferença de potencial (tensão), deve-se ligar os terminais vermelho e preto do multímetro em paralelo com os terminais do componente ou o trecho do circuito. Caso a ligação seja feita em série, o multímetro marcará zero.

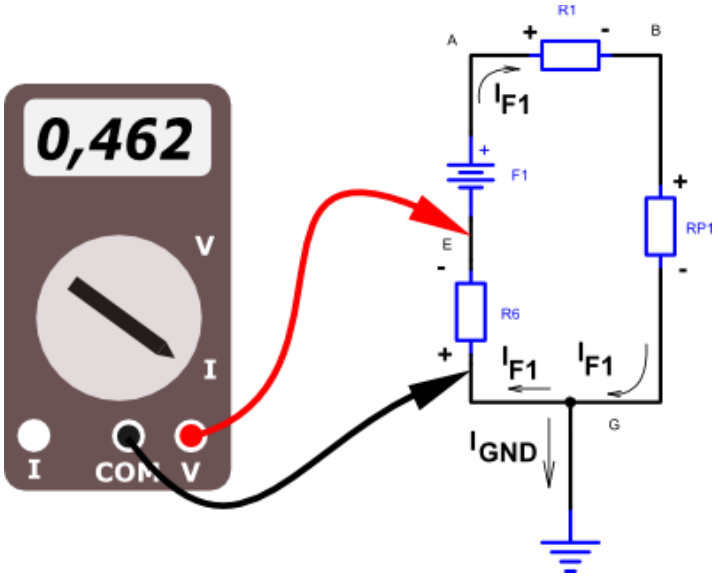


Figura 18: uso de um multímetro para se aferir tensão.

Por fim, para se aferir corrente, deve-se interromper o trecho do circuito em que a corrente deve ser aferida e conectar o multímetro em série em relação a esse trecho. Caso a ligação seja feita em paralelo, haverá curto circuito, o que pode acarretar na queima do equipamento.

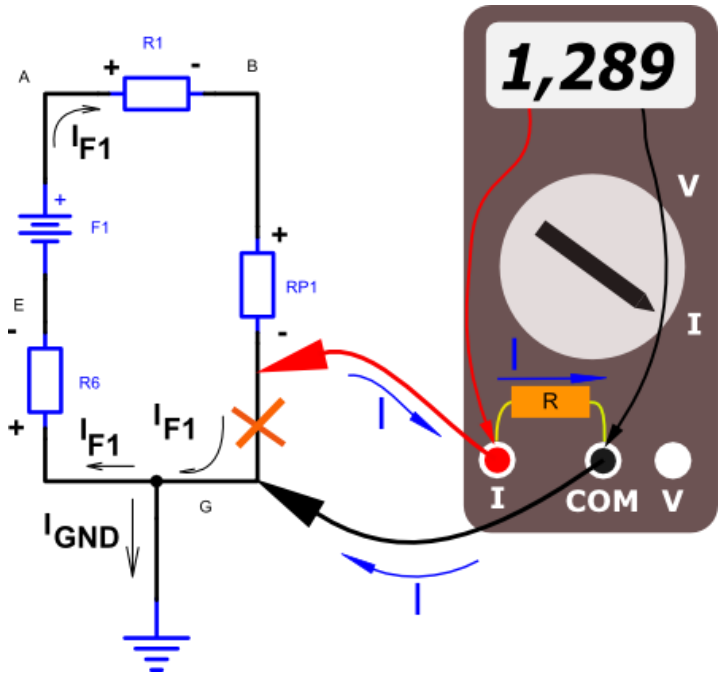


Figura 19: uso de um multímetro para se aferir corrente.

Tendo garantido a alimentação da protoboard e compreendido seu funcionamento, é o momento de colocarmos nossos conhecimentos em prática. Para tanto, considere o seguinte circuito:

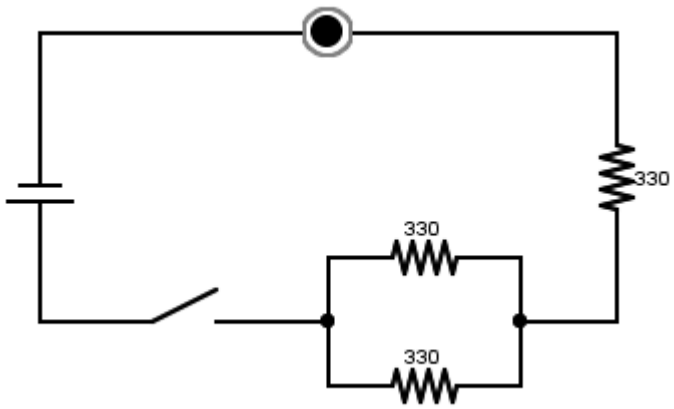


Figura 20: circuito a ser montado na protoboard.

Ele pode ser montado na protoboard da seguinte forma:

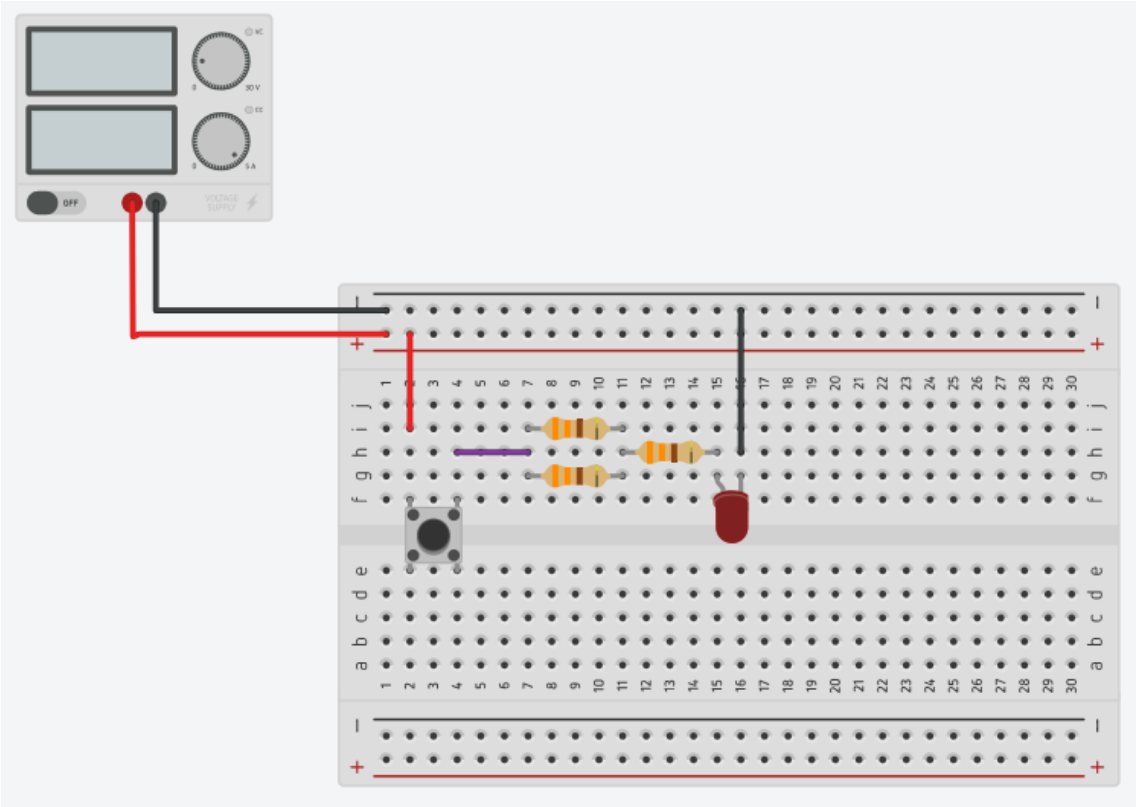


Figura 21: circuito já montado na protoboard.

Caso você opte por realizar a montagem física do circuito, recomendo que também faça medições de tensão e corrente em pontos de sua preferência para, depois, comparar os valores aferidos com as expectativas teóricas. Tente, também, justificar eventuais discrepâncias encontradas.

4 O ADVENTO DO TRANSISTOR

Antes de prosseguirmos, precisamos abordar alguns conceitos fundamentais para a compreensão dos circuitos digitais, uma vez que todos os módulos que trabalharemos a partir do próximo capítulo dependem destes conceitos para existir.

4.1 OS SEMICONDUTORES

Semicondutores são sólidos geralmente cristalinos cuja condutividade elétrica é sensível a condições ambientais, como temperatura ou estado elétrico. Assim, um semicondutor pode ser isolante ou condutor, conforme as condições ambientais – no nosso caso, trabalharemos como o estado elétrico como fator decisivo.

Um semicondutor pode ser intrínseco ou dopado. Os semicondutores dopados contêm cerca de 1 átomo de um elemento químico indesejado (portanto, uma impureza) para cada bilhão de átomos do material do semicondutor. Já os semicondutores dopados possuem, intencionalmente, cerca de 1 átomo de um elemento químico desejado (chamado de dopante) para cada milhão de átomos do material do semicondutor.

Existem dois tipos de semicondutores dopados: do tipo N e do tipo P. Nos semicondutores do tipo N, ocorre a adição de materiais que, ao realizar as ligações químicas com o material do semicondutor, ficam com elétrons livres. Portanto, o semicondutor do tipo N apresenta carga negativa.

Já nos semicondutores do tipo P, ocorre a adição de materiais que, ao realizar ligações químicas com o material do semicondutor, criam lacunas (por possuírem menos elétrons que a quantidade de ligações realizada). Por possuir elétrons a menos, o semicondutor do tipo P apresenta carga positiva.

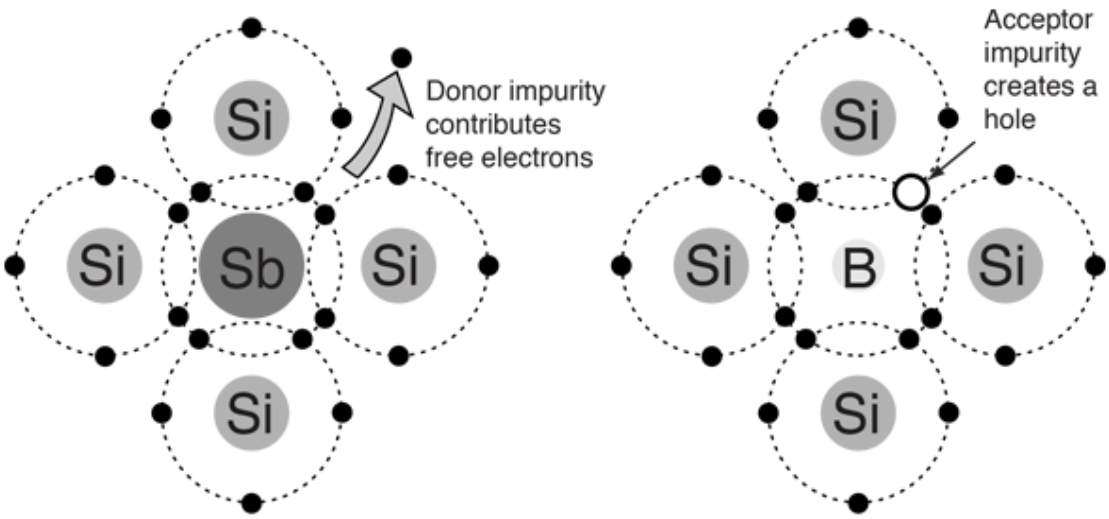


Figura 22: semicondutores de silício do tipo N (a esquerda, com adição de antimônio) e do tipo P (a direita, com adição de boro).

4.2 O TRANSISTOR

Até agora, trabalhamos com chaves que permitiam a passagem de corrente mediante um sinal mecânico, normalmente manual. Porém, existe um componente capaz de controlar a passagem de corrente elétrica por meio de um sinal elétrico: o transistor.

Transistores são componentes feitos de material semicondutor que possuem duas funções: a de amplificação (que não é tão importante em nosso contexto) e a de controle de corrente, fundamental para o entendimento de circuitos digitais. Existem vários tipos de transistores, e dedicaremos os próximos parágrafos para explicar o funcionamento deste componente.

Figura 23: alguns transistores.

Transistores podem ser entendidos como pequenas chaves, mas que, ao invés de serem acionadas por um impulso mecânico, são acionadas por um sinal elétrico. Existem vários tipos de transistor, mas, nesse livro, utilizaremos os transistores BC337. Esses transistores (assim como praticamente todos os outros) possuem três terminais: o terminal coletor, base e emissor. Cada um desses terminais corresponde a um pino do componente, conforme o esquema abaixo:

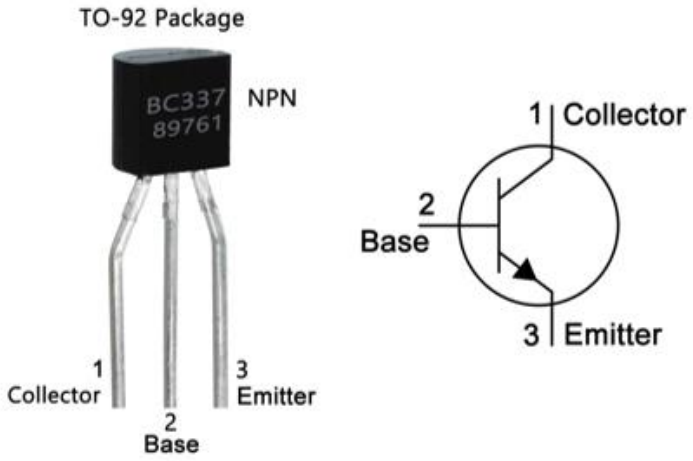


Figura 24: esquema de um transistor do tipo BC337.

Internamente, o transistor NPN é constituído de três regiões: duas de um semicondutor com dopagem N, e uma de um semicondutor com dopagem P, conforme o esquema abaixo:

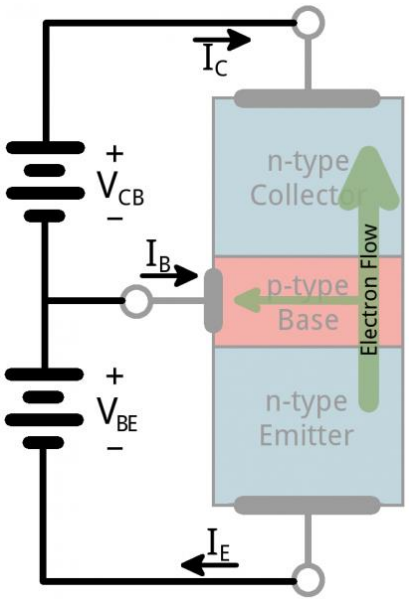


Figura 25: constituição de um transistor NPN.

Normalmente, elétrons podem fluir com facilidade de uma região N para uma região P, mas fluir de uma região P para uma região N requer muita voltagem. Em um semicondutor, elétrons podem facilmente fluir da região P para a região N se a tensão na base for maior que a tensão no emissor.

O transistor é projetado para passar elétrons do emissor para o coletor (então a corrente flui do coletor para o emissor). O emissor "emite" elétrons para a base, que controla o número de elétrons que o emissor "emite". A maior parte dos elétrons é "coletada" pelo coletor, que os envia para a próxima parte do circuito.

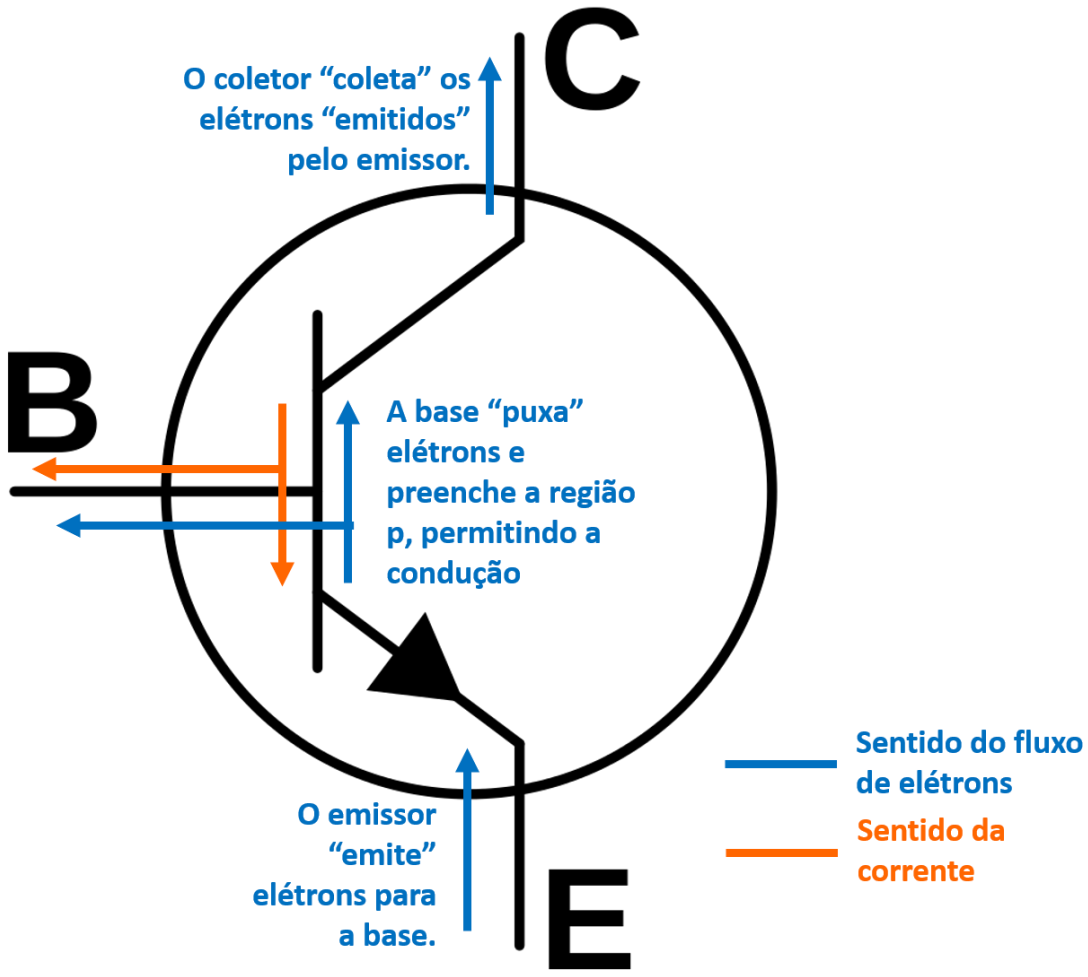


Figura 26: funcionamento de um transistor NPN.

O funcionamento de um transistor é bastante complexo. Porém, para nossas aplicações, consideraremos que todo transistor NPN (o único aqui utilizado) funciona da seguinte forma: se há tensão no pino base, então há passagem de corrente elétrica do coletor para o emissor. Caso contrário, não há passagem de corrente.

Na prática, nossos transistores terão, em sua configuração mais simples, o emissor ligado a um fio doador de elétrons (terra ou carregado negativamente). O coletor e a base ficarão ligados a uma fonte de alimentação, ambas de um valor fixo de 5 V.

A base, por sua vez, também estará ligada a uma chave manual que controla a entrada do transistor, mas que pode ser substituída pela saída de outro transistor se assim necessário. Haverá uma resistência entre a chave e a base, a qual deve ter um valor entre 4000 e 20000 Ω.

Em algum ponto entre o fio terra do emissor e a fonte do coletor também haverá uma resistência, de valor variável conforme o propósito do circuito. Em algum ponto desse percurso, também haverá um fio de saída, que pode ser ligado a um multímetro ou a um LED, ou, ainda, a outro circuito como entrada do transistor.

Assim, um circuito com transistor no qual a saída corresponde ao mesmo valor da entrada (exibida num LED e controlada por uma chave, respectivamente) pode ser feito da seguinte forma:

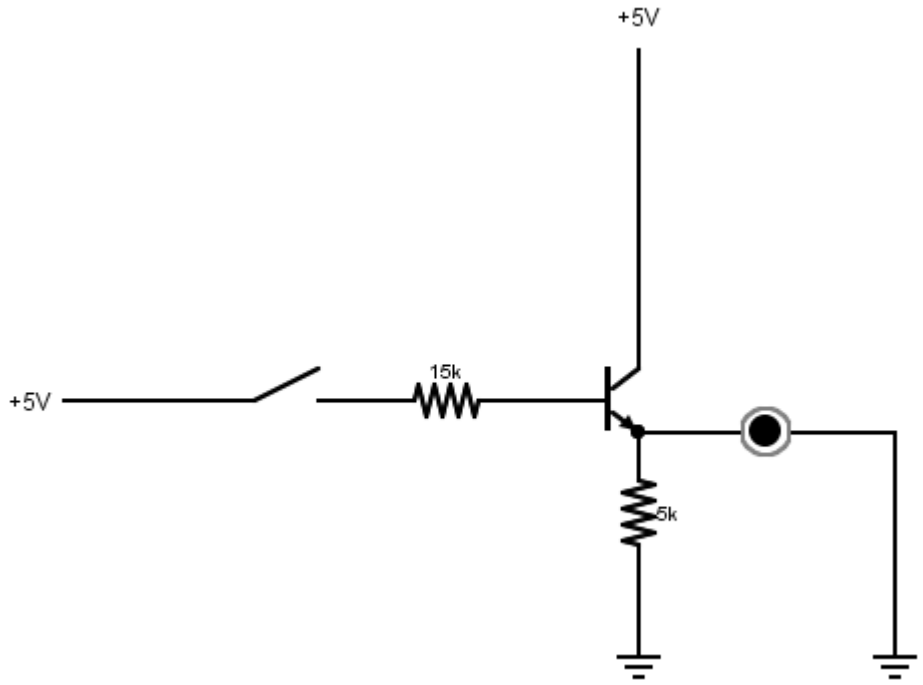


Figura 27: circuito com transistor, no qual a entrada é igual a saída.

O circuito tem duas situações possíveis, conforme a posição da chave:

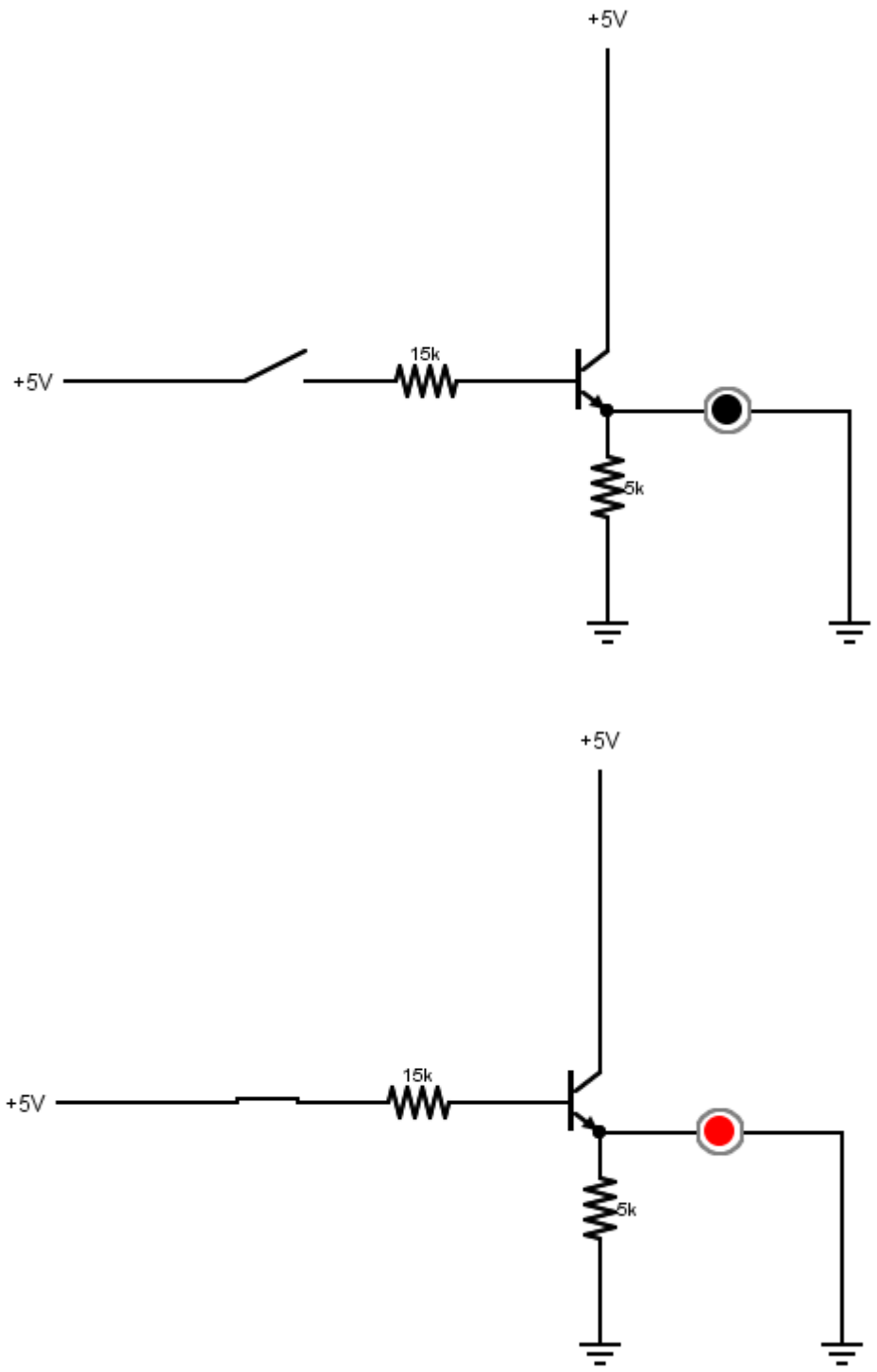


Figura 28: situações do circuito da figura anterior.

De fato, esse circuito carece de uma utilidade prática. Porém, transistores são a base para construção de outros circuitos, que por sua vez são a base de toda a lógica digital.

4.3 NÍVEIS LÓGICOS

Podemos considerar que um transistor opera em função de um ou mais sinais de entrada, e devolve um ou mais sinais de saída. Podemos, também, considerar que o sinal de entrada corresponde a tensão inserida na base, e que o sinal de saída corresponde a tensão que seria lida por um multímetro posicionado no lugar do LED (usamos o LED apenas para ter um indicador visual da presença de tensão).

Também já sabemos por quais razões este curso é chamado de lógica digital: existe um número finito de estados que um sinal pode exibir. Mais especificamente, existem apenas dois sinais: 1 e 0, alta ou baixa voltagem, respectivamente.

Porém, antes é necessário definir o que é alta e baixa voltagem. Idealmente, a alta voltagem corresponde a tensão da fonte de alimentação, que no nosso caso é $5\,V$. Também idealmente, a baixa voltagem corresponde a tensão nula, que no nosso caso é $0\,V$.

Entretanto, na prática, os sinais de entrada e de saída dificilmente se apresentam na forma de valores exatos, como gostaríamos. O que realmente acontece é que transistores costumam devolver sinais ligeiramente próximos dos valores ideais, e, por essa razão, também são projetados para aceitar sinais ligeiramente próximos dos ideais.

Assim, precisamos definir quatro intervalos, para voltagens altas e baixas na entrada e na saída de um transistor do tipo NPN. Tensões de entrada aceitáveis variam entre 0 e $0{,}8\,V$ para sinais em baixa voltagem, e entre 2 e $5\,V$ para sinais em alta voltagem. Já as tensões de saída variam 0 e $0{,}5\,V$ para sinais em baixa voltagem, e entre $2{,}7$ e $5\,V$ para sinais em alta voltagem.

Quaisquer entradas entre $0{,}8$ e $2\,V$ produzem um comportamento inesperado e, portanto, a saída produzida é desconhecida. Justamente por essa razão, tais situações devem ser evitadas quando o circuito é planejado.

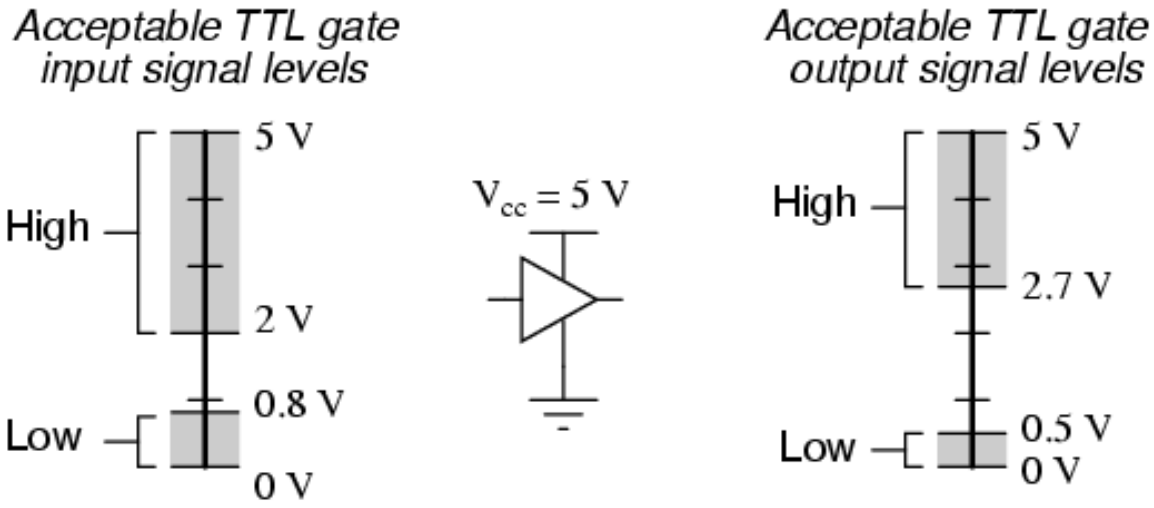


Figura 29: sinais de entrada e saída aceitáveis para alta e baixa voltagem em um transistor NPN.

5 AS PORTAS LÓGICAS

É chegado o momento de compreender a estrutura das portas lógicas. Portas lógicas são circuitos pequenos envolvendo transistores, que recebem uma ou duas entradas, e devolvem uma saída conforme um dado comportamento. A seguir, conheceremos o funcionamento das seis portas lógicas existentes.

Do próximo em diante, podemos consideras as portas lógicas como caixas fechadas, e não mais nos preocuparmos com o que ocorre a nível de transistores.

5.1 A PORTA NOT

Finalmente, é chegado o momento de construirmos os blocos fundamentais de circuitos digitais: a porta NOT. A porta NOT corresponde a um circuito que inverte o valor da entrada. Assim, caso o circuito receba um sinal 1, ele devolverá um sinal 0 e, caso o circuito receba um sinal 0, ele devolverá um sinal 1.

Uma possibilidade de circuito que faz essa operação, bem como seus dois estados possíveis, está representada abaixo:

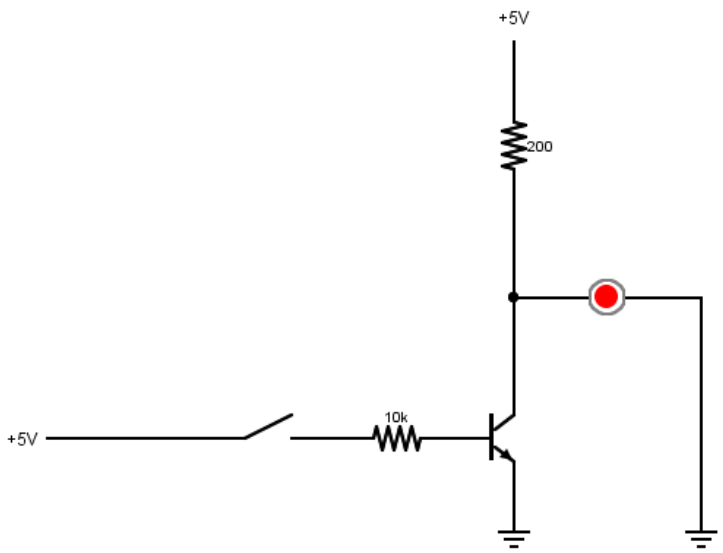


Figura 30: Circuito inversor.

Esse circuito é tão comum que ele pode ser representado por um símbolo próprio, o símbolo da porta lógica NOT:

Figura 31: símbolo da porta lógica NOT.

Assim, os possíveis estados da porta lógica NOT passam a ser:

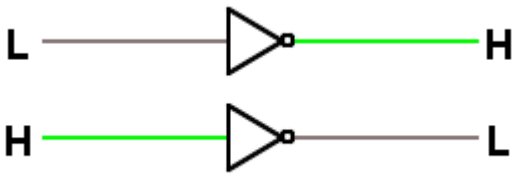


Figura 32: dois possíveis estados da porta lógica NOT.

Essa operação, de inverter o sinal de entrada, é denominada negação lógica, e pode ser representada pelo sinal de barramento ($\bar{}$). Podemos, também, escrever essa operação através de uma função matemática, dada por:

$$f(a) = \bar{a}$$

Outra possibilidade para descrever essa operação é através de uma tabela verdade, que indica o comportamento do circuito conforme suas entradas:

Tabela 1: tabela-verdade para o circuito inversor.

x	$\bar{x}$
0	1
1	0

5.2 A PORTA AND

Outro circuito bastante importante em lógica digital é a porta AND. A porta AND corresponde a um circuito que apresente 1 como saída se e somente se todas as entradas do circuito forem 1, e apresenta 0 caso contrário.

Uma possibilidade de circuito que faz essa operação, bem como seus quatro estados possíveis, está representada abaixo. Esse circuito é chamado circuito de conjunção.

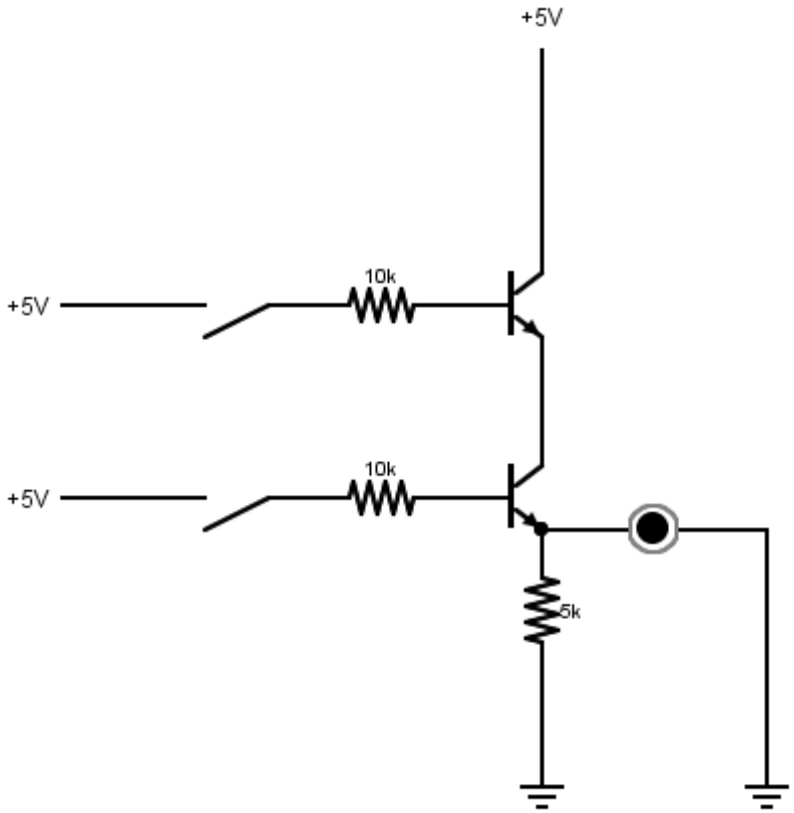


Figura 33: circuito de conjunção.

Esse circuito é tão comum que ele pode ser representado por um símbolo próprio, o símbolo da porta lógica AND:

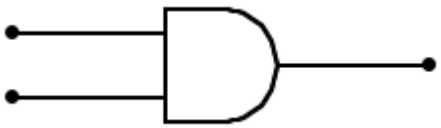


Figura 34: símbolo da porta lógica AND.

Assim, os possíveis estados da porta lógica AND passam a ser:

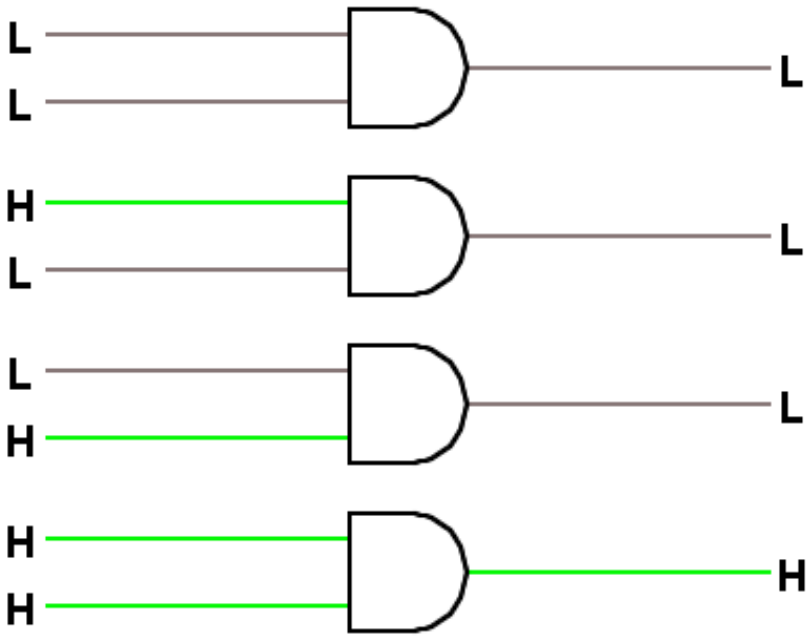


Figura 35: quatro possíveis estados da porta lógica AND.

Essa operação é denominada conjunção lógica, e pode ser representada pelo sinal de multiplicação (.). Podemos, também, escrever essa operação através de uma função matemática, dada por:

$$f(a, b) = a.b$$

Outra possibilidade para descrever essa operação é através de uma tabela verdade, que indica o comportamento do circuito conforme suas entradas:

Tabela 2: tabela-verdade para o circuito de conjunção.

b	a	$a\,.\,b$
0	0	0
0	1	0
1	0	0
1	1	1

5.3 A PORTA OR

Outro circuito bastante importante em lógica digital é a porta OR. A porta OR corresponde a um circuito que apresente 1 como saída se uma das entradas forem 1, e apresenta 0 caso contrário.

Uma possibilidade de circuito que faz essa operação, bem como seus quatro estados possíveis, está representada abaixo. Esse circuito é chamado circuito de disjunção.

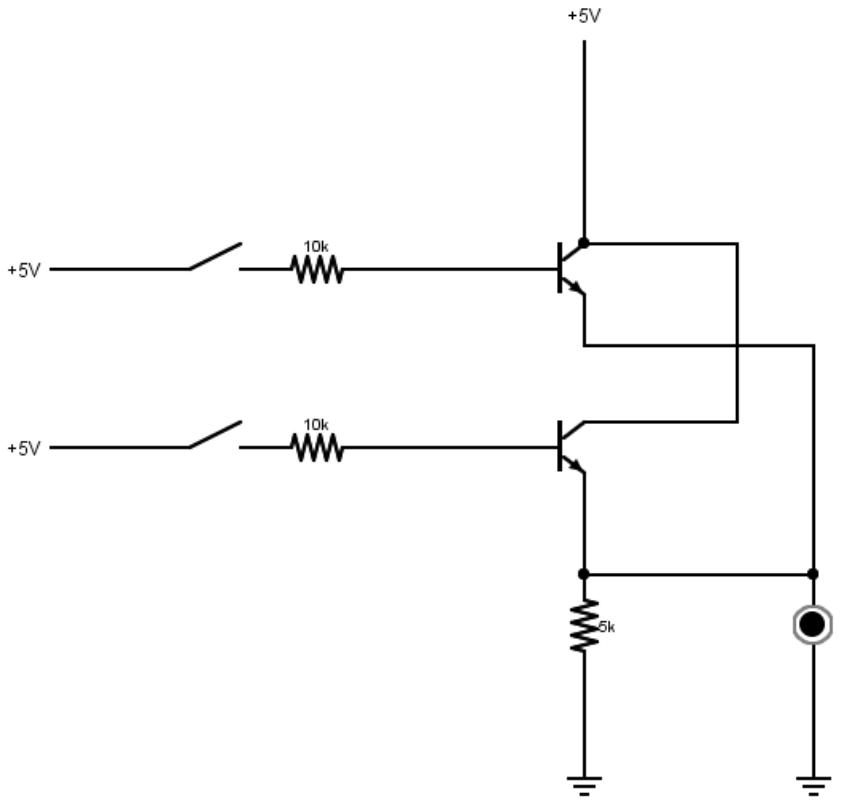


Figura 36: circuito de disjunção.

Esse circuito é tão comum que ele pode ser representado por um símbolo próprio, o símbolo da porta lógica OR:

Figura 37: símbolo da porta lógica OR.

Assim, os possíveis estados da porta lógica OR passam a ser:

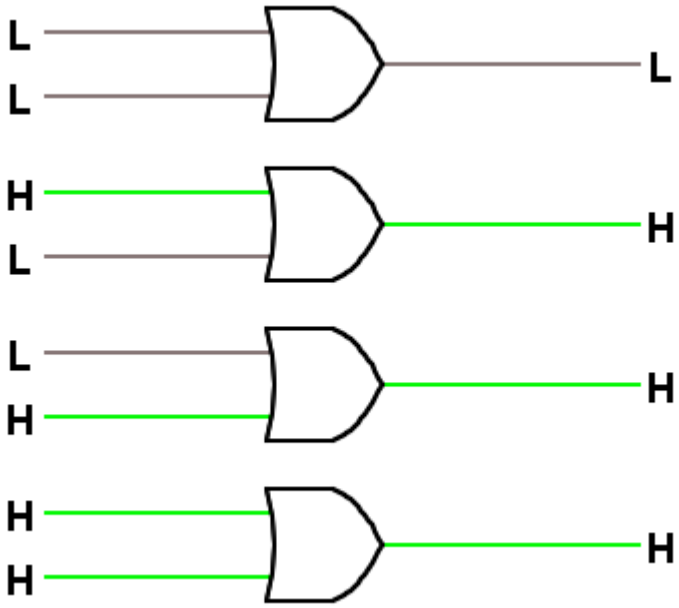


Figura 38: quatro possíveis estados da porta lógica OR.

Essa operação é denominada disjunção lógica, e pode ser representada pelo sinal de soma $(+)$. Podemos, também, escrever essa operação através de uma função matemática, dada por:

$$f(a,b) = a + b$$

Outra possibilidade para descrever essa operação é através de uma tabela verdade, que indica o comportamento do circuito conforme suas entradas:

Tabela 3: tabela-verdade para o circuito de disjunção.

b	a	$a + b$
0	0	0
0	1	1

1	0	1
1	1	1

Neste momento, já projetamos as 3 portas lógicas básicas. Porém, ainda existem outras 3 portas, que são o resultado de combinações entre as portas básicas. Por essa razão, não exibiremos todos os casos no circuito com transistores, mas apenas o caso base, em que $a = b = 0$.

5.4 A PORTA NAND

Outro circuito bastante importante em lógica digital é a porta NAND. A porta NAND corresponde a associação em série de uma porta AND seguida de uma porta NOT.

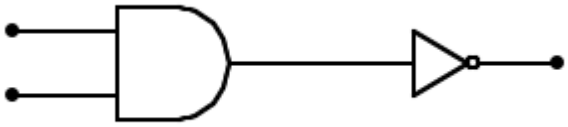


Figura 39: formação da porta NAND.

Por ser uma operação muito comum, essa porta lógica tem um símbolo próprio, representado abaixo:

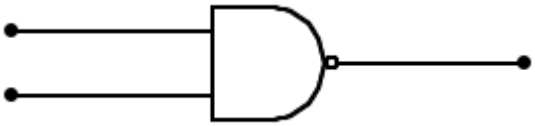


Figura 40: símbolo da porta NAND.

Seu comportamento é deduzível da sua formação, e é, basicamente, o oposto do comportamento da porta AND. Assim, a porta NAND devolverá 0 apenas caso as duas entradas sejam 1. Caso contrário, ela devolverá 1.

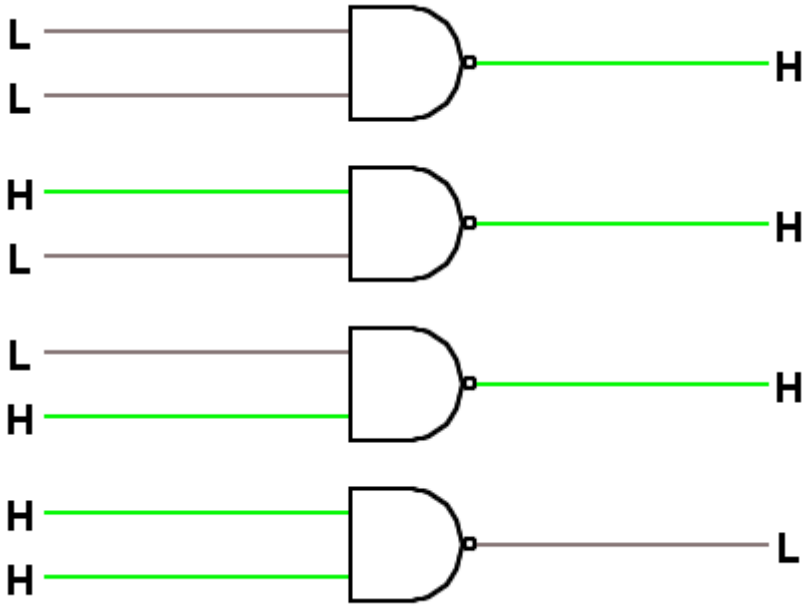


Figura 41: quatro possíveis estados da porta lógica NAND.

A porta lógica NAND também pode ser representada pelo sinal da linha vertical (|). Podemos, também, escrever essa operação através de uma função matemática, dada por:

$$f(a,b) = a \mid b$$

Outra possibilidade para descrever essa operação é através de uma tabela verdade, que indica o comportamento do circuito conforme suas entradas:

Tabela 4: tabela-verdade para a porta lógica NAND.

b	a	$a \mid b$
0	0	1
0	1	1
1	0	1
1	1	0

Por fim, a porta lógica NAND também pode ser construída utilizando transistores, conforme o seguinte circuito:

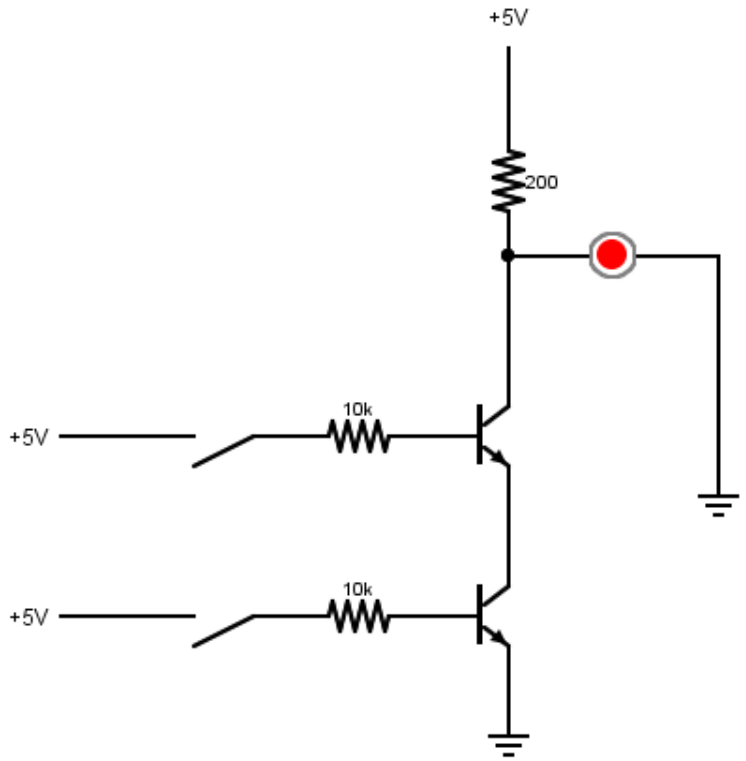


Figura 42: circuito com transistores de uma porta NAND.

5.5 A PORTA NOR

Outro circuito bastante importante em lógica digital é a porta NOR. A porta NOR corresponde a associação em série de uma porta OR seguida de uma porta NOT.

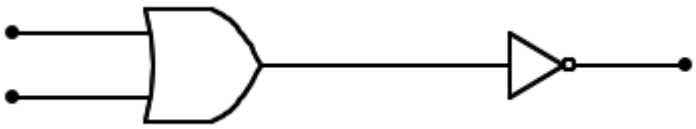


Figura 43: formação da porta NOR.

Por ser uma operação muito comum, essa porta lógica tem um símbolo próprio, representado abaixo:

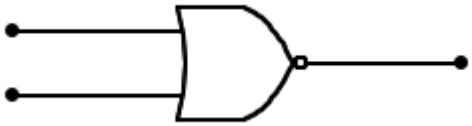


Figura 44: símbolo da porta NOR.

Seu comportamento é deduzível da sua formação, e é, basicamente, o oposto do comportamento da porta OR. Assim, a porta NOR devolverá 1 apenas caso as duas entradas sejam 0. Caso contrário, ela devolverá 1.

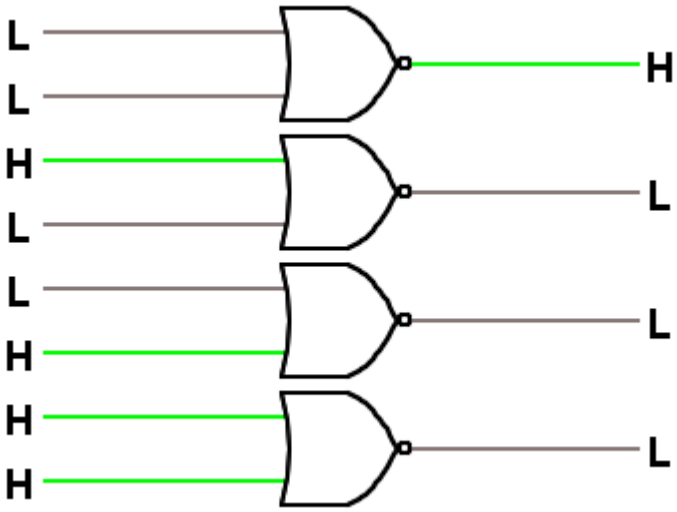


Figura 45: quatro possíveis estados da porta lógica NOR.

A porta lógica NAND também pode ser representada pelo sinal da seta para baixo (↓). Podemos, também, escrever essa operação através de uma função matemática, dada por:

$$f(a, b) = a \downarrow b$$

Outra possibilidade para descrever essa operação é através de uma tabela verdade, que indica o comportamento do circuito conforme suas entradas:

Tabela 5: tabela-verdade para a porta NOR.

b	a	$a \mid b$
0	0	1
0	1	0

1	0	0
1	1	0

Por fim, a porta lógica NOR também pode ser construída utilizando transistores, conforme o seguinte circuito:

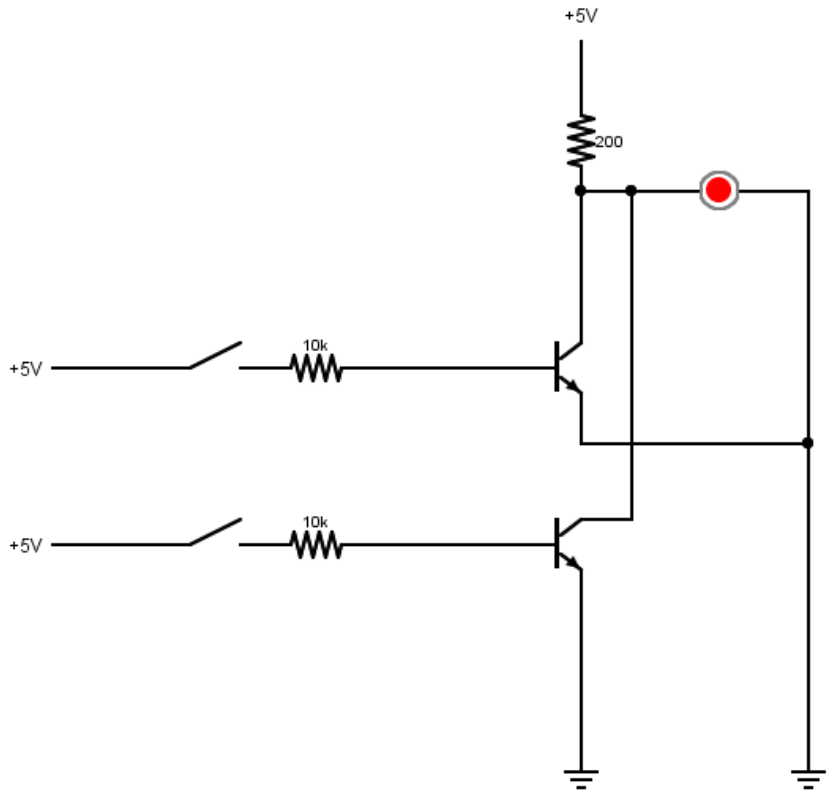


Figura 46: circuito com transistores de uma porta NOR.

5.6 A PORTA XOR

Outro circuito bastante importante em lógica digital é a porta XOR. A porta XOR corresponde a uma associação de portas NAND, AND e OR conforme a expressão $f(a,b) = (A \mid B).(A + B)$. Em termos de um arranjo de portas lógicas, ela pode ser descrita por:

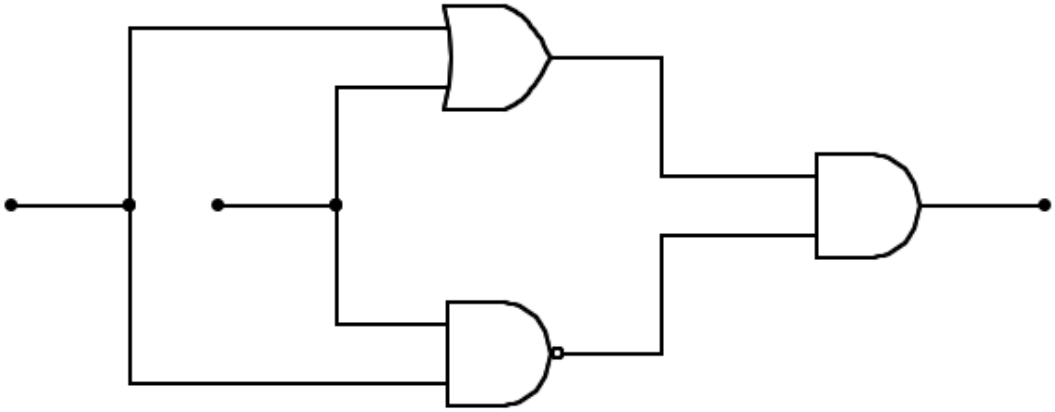


Figura 47: formação da porta XOR.

Por ser uma operação muito comum, essa porta lógica tem um símbolo próprio, representado abaixo:

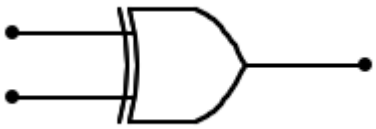


Figura 48: símbolo da porta XOR.

Seu comportamento não é tão deduzível, mas o nome da operação nos ajuda a entender: “ou exclusivo”. Portanto, a porta lógica devolverá 1 se e somente se apenas 1 entrada for 1. Caso contrário, ela devolverá zero.

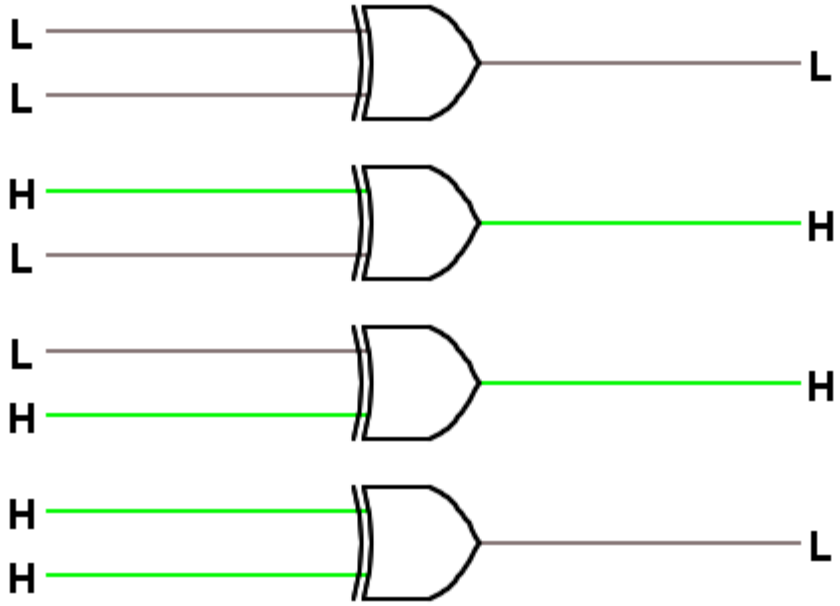


Figura 49: quatro possíveis estados da porta lógica XOR.

A porta lógica NAND também pode ser representada pelo sinal de ou exclusivo ($\oplus$). Podemos, também, escrever essa operação através de uma função matemática, dada por:

$$f(a, b) = a \oplus b$$

Outra possibilidade para descrever essa operação é através de uma tabela verdade, que indica o comportamento do circuito conforme suas entradas:

Tabela 6: tabela-verdade para a porta XOR.

b	a	$a \mid b$
0	0	0
0	1	1
1	0	1
1	1	0

Por fim, a porta lógica XOR também pode ser construída utilizando transistores, conforme o seguinte circuito:

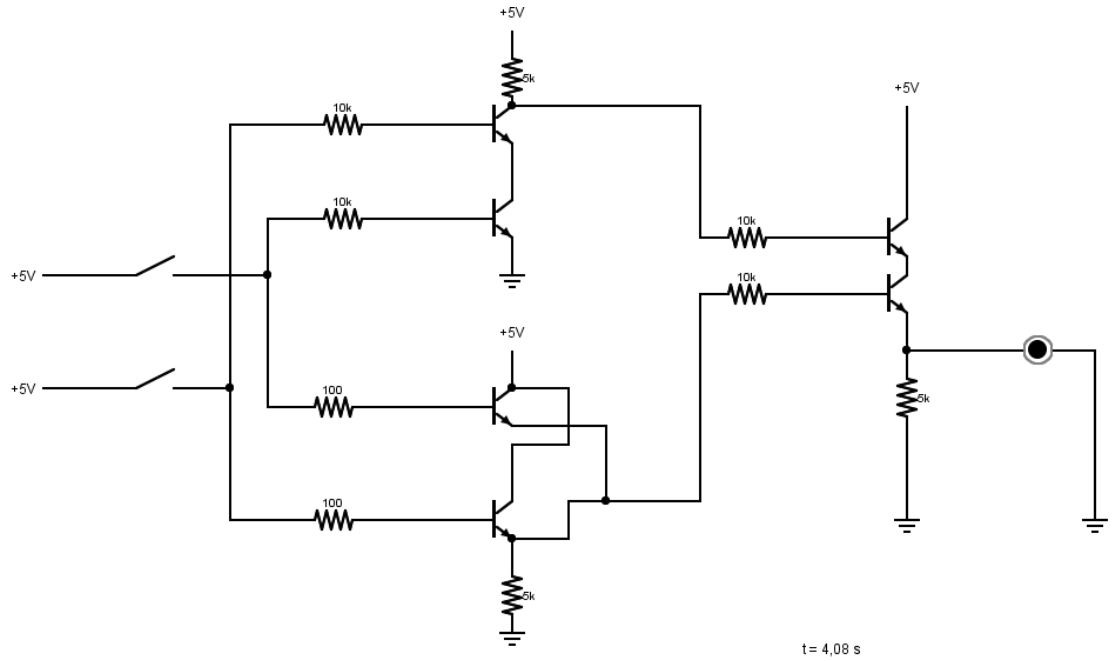


Figura 50: circuito com transistores de uma porta XOR.

5.7 PORTAS COM MAIS DE DUAS ENTRADAS

Todas as portas – com exceção da porta NOT, que só aceita uma única entrada – podem aceitar quantas entradas se fizer necessário. Para tanto, basta desenhar o componente com mais entradas – quantas forem necessárias. O funcionamento da porta é exatamente o mesmo. Se precisar, releia a última seção e troque as palavras “duas” por qualquer número maior que dois.

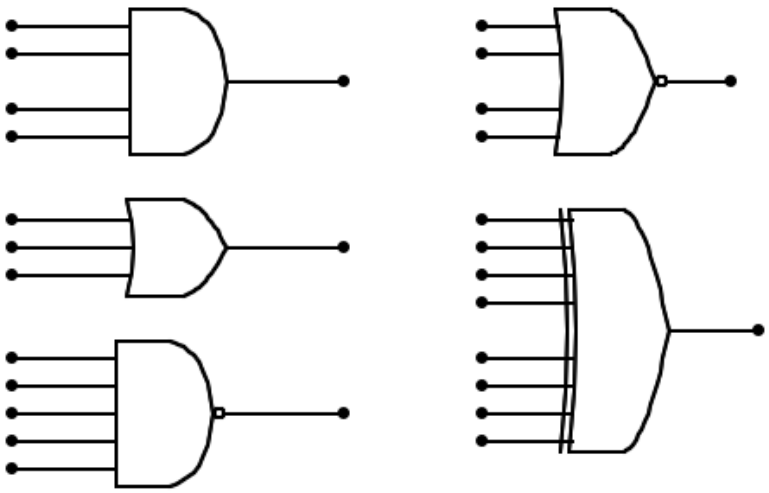


Figura 51: algumas portas lógicas com mais de duas entradas.

5.8 CIRCUITOS INTEGRADOS

No início deste capítulo, mencionamos que as portas lógicas são as unidades básicas de todo circuito digital, e que não mais era necessário nos atentarmos ao funcionamento a nível de transistores. Mas como isso é feito?

Portas lógicas são fabricadas comercialmente na forma de circuitos integrados. Circuitos integrados são pequenos chips com várias entradas e saídas, e algumas portas lógicas embutidas.

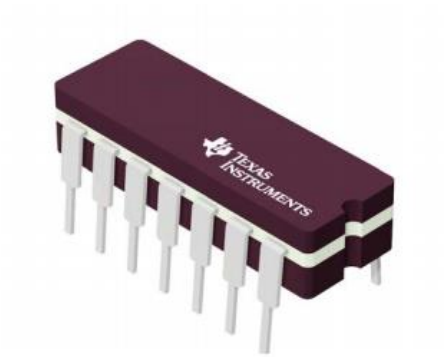


Figura 52: aparência geral de um circuito integrado.

Existem seis modelos comerciais correspondentes a cada uma das seis portas lógicas, além de uma porção de outros modelos que atendem à casos específicos. Na imagem abaixo, há um esquema desses seis modelos.

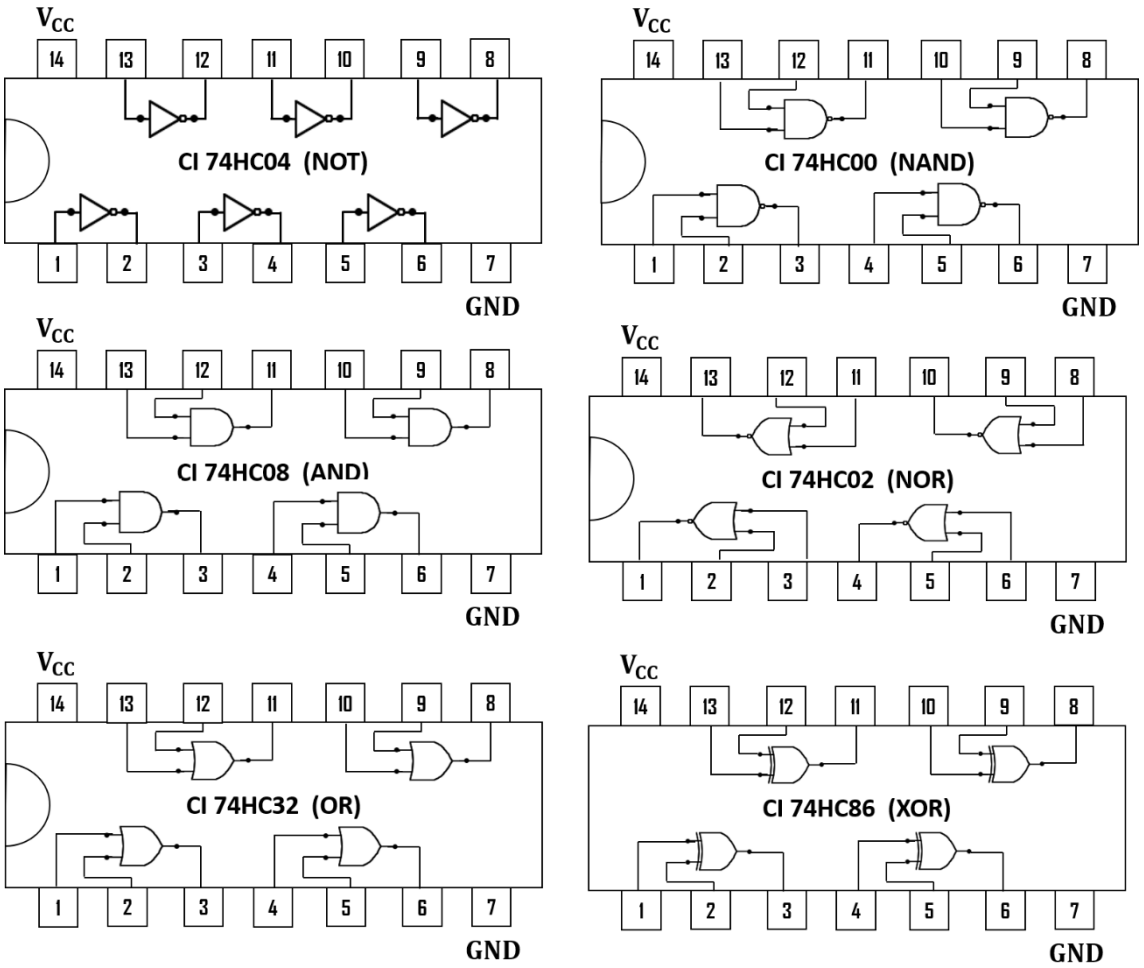


Figura 53: esquema dos circuitos integrados com as 6 portas lógicas.

Em todos eles, as entradas V_{CC} e GND são usadas para fornecer a tensão nominal do circuito (que no nosso caso é 5 V) e para fornecer o neutro, respectivamente.

Todo circuito integrado possui um documento de especificações técnicas fornecido pelo fabricante, denominado *datasheet*. *Datasheets* destes componentes podem ser encontradas em http://www.baudaeletronica.com.br/Documentos/SN74HC04N.pdf, https://www.baudaeletronica.com.br/Documentos/sn74hc00.pdf, https://www.baudaeletronica.com.br/Documentos/74HC_HCT02_CNV_2.pdf, http://www.baudaeletronica.com.br/Documentos/SN74HC08N.pdf,

https://www.baudaeletronica.com.br/Documentos/74HC32.pdf e https://www.baudaeletronica.com.br/Documentos/74HC_HCT86_CNV_2.pdf.

Além de informações que já conhecemos, a *datasheet* também contém os valores máximos de tensão que podem ser fornecidos para o circuito, de forma a não danificar o componente. Esses valores podem ser encontrados na seção denominada "*operating conditions*", ou algo parecido.

recommended operating conditions (see Note 3)

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		2	5	6	V
V_{IH}	High-level input voltage	V_{CC} = 2 V	1.5			V
		V_{CC} = 4.5 V	3.15			
		V_{CC} = 6 V	4.2			
V_{IL}	Low-level input voltage	V_{CC} = 2 V			0.5	V
		V_{CC} = 4.5 V			1.35	
		V_{CC} = 6 V			1.8	
V_I	Input voltage		0		V_{CC}	V
V_O	Output voltage		0		V_{CC}	V
$\Delta t/\Delta v$	Input transition rise/fall time	V_{CC} = 2 V			1000	ns
		V_{CC} = 4.5 V			500	
		V_{CC} = 6 V			400	
T_A	Operating free-air temperature	I-suffix device	−40		85	°C
		Q-suffix device	−40		125	

NOTE 3: All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, *Implications of Slow or Floating CMOS Inputs*, literature number SCBA004.

Figura 54: condições de operação retiradas da *datasheet* de um circuito integrado.

No exemplo da figura acima, deve-se levar em consideração que o valor da voltagem de alimentação pode ser, no máximo, $6\,\mathrm{V}$. Portanto, exceder esse limite acarretará em danos ao componente. Os valores máximos para input são o mesmo da alimentação.

A datasheet também contém informações importantes no tocante aos níveis lógicos adotados no componente. Em projetos pequenos, tal tipo de planejamento pode não ser necessário, mas, em projetos maiores, pode ser um fator importante a ser considerar.

Por fim, vale lembrar que não é necessária a utilização de resistores em série com relação aos inputs, ao V_{CC} e ao GND em um circuito integrado. Porém, ao visualizar o output do circuito integrado por meio de um LED, tal resistor é necessário de forma a fazer com que o LED não apresente brilho com o sinal em baixa voltagem. Caso contrário,

o LED brilhará mais forte quando um sinal de alta voltagem for fornecido, mas ainda continuará brilhando em baixa voltagem.

Na prática, o circuito integrado já contém diodos e resistores para evitar danos dentro dos valores de tensão especificados na datasheet como

Para a porta NOT, o circuito pode ser montado na *protoboard* da seguinte forma:

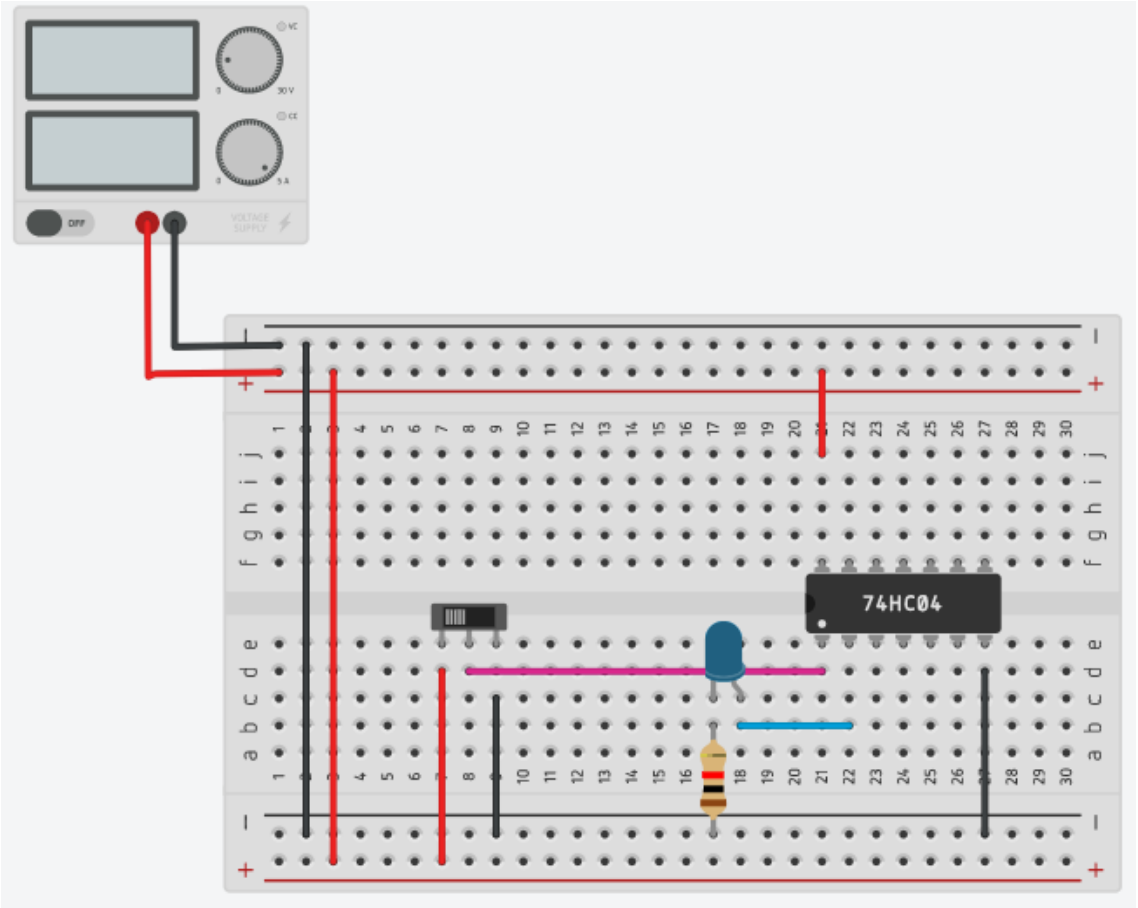


Figura 55: circuito lógico com porta NOT.

Para todas as demais portas (uma vez que os chips seguem um mesmo padrão), o circuito pode ser montado na *protoboard* da seguinte forma:

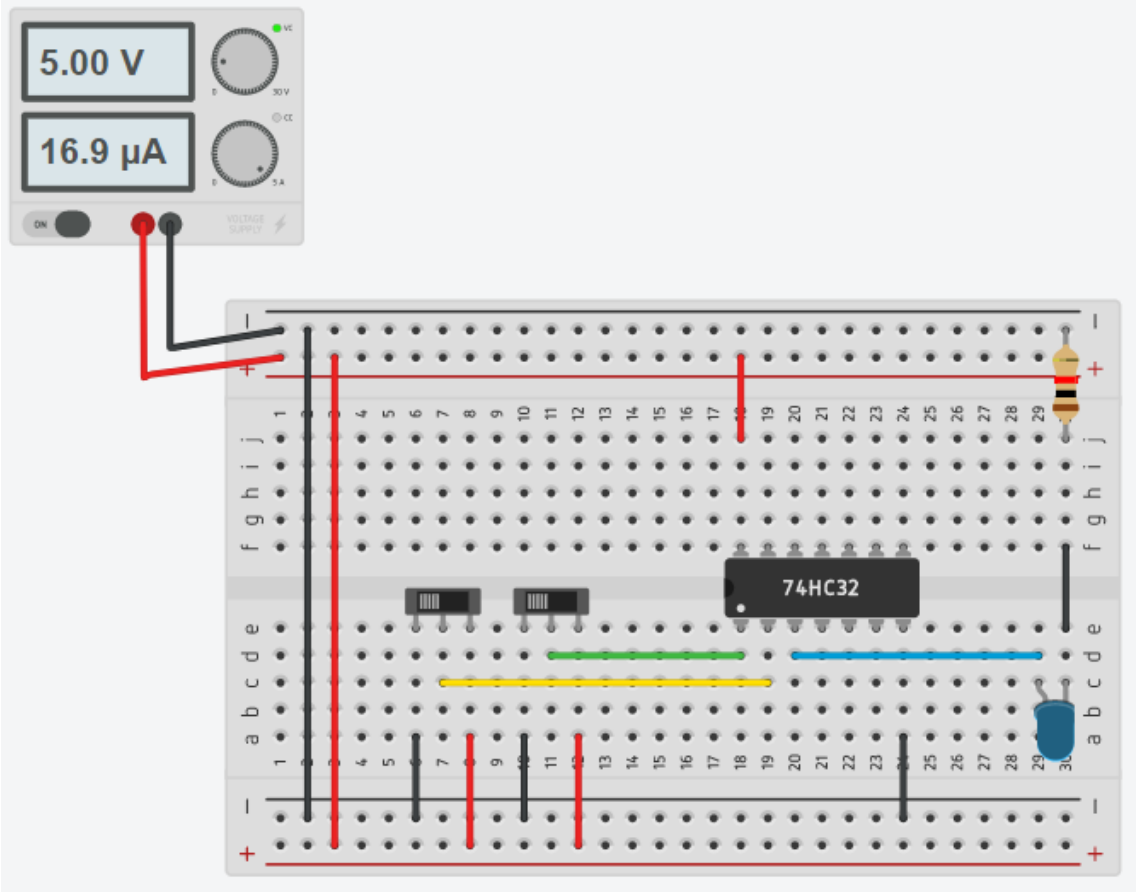


Figura 56: circuito lógico com a porta OR, mas que se repete para qualquer outro circuito de 4 portas.

6 DESCREVENDO CIRCUITOS LÓGICOS

Já sabemos quais são os blocos fundamentais de todos os circuitos e sabemos bem como eles funcionam. É chegado, então, o momento de associarmos estes blocos em circuitos maiores e de aprendermos como um circuito e seu comportamento podem ser representados.

6.1 CIRCUITOS SIMPLES COM UMA PORTA LÓGICA

No capítulo anterior, verificamos que existem 6 portas lógicas com comportamentos específicos. É chegado o momento de verificarmos como essas portas lógicas podem ser aplicadas em situações práticas. Suponha que você precise projetar um circuito que atenda à seguinte especificação:

"Um alarme para um consultório médico deve ser ativado sempre que uma porta for aberta, ou que um sensor de presença detecte alguém na recepção. Construa um circuito capaz de ativar o alarme com um sinal 1, tendo a disposição um sensor de presença que devolve 1 sempre que detectar algo a frente, e um sensor na porta que devolve 1 sempre que a porta for aberta"

A solução deste problema envolve somente identificar qual tipo de relação lógica está envolvida. O alarme deve ser ativado sempre que uma das duas situações ocorrer, e não há motivos para não o acionar quando ambas as situações ocorrerem simultaneamente. Portanto, os dois sensores podem ser ligados por uma porta lógica OR, que devolve a saída. Assim, a implementação seria:

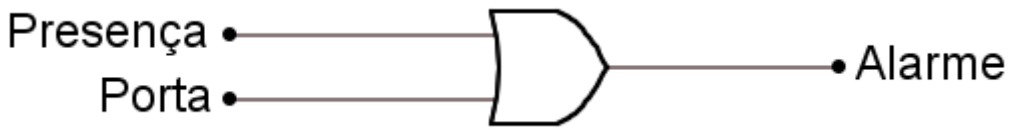


Figura 57: implementação de circuito que resolve o problema proposto.

6.2 ASSOCIANDO PORTAS LÓGICAS

No capítulo anterior, verificamos que é possível associar portas lógicas de forma que a saída de uma porta corresponda a entrada de outra. É exatamente dessa forma que circuitos digitais são constituídos.

Por exemplo, suponha que você precise criar um circuito que atenda à seguinte especificação:

"Um aquecedor deve ligar sempre que a temperatura de uma sala passar de $9° C$*, quando a hora no momento estiver entre 6 e 7 da manhã. Construa um circuito com uma saída, que deve ser 1 para ligar o aquecedor e 0 para desliga-lo, tendo a disposição um sensor de temperatura que devolve 1 caso a temperatura seja maior ou igual a 9º C e 0 caso contrário, e um relógio que devolve 1 sempre que a hora estiver entre 06:00 e 07:00."*

Agora, já temos um problema um pouco mais complicado: o funcionamento do sensor de temperatura funciona de forma inversa, devolvendo 1 na situação em que não precisamos ligar o aquecedor. A solução para esse problema envolve, portanto, o uso de uma porta NOT invertendo a saída do sensor de temperatura, e uma porta AND conjugando a saída da porta NOT e do relógio. Portanto, temos:

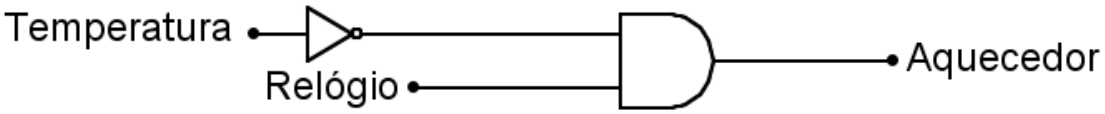


Figura 58: implementação de circuito que resolve o problema proposto.

Vamos complicar ligeiramente nossas especificações. Suponha, agora, que você precisa projetar um circuito que atenda à seguinte descrição:

"Um sistema de farol com acionamento automático prevê que o farol de um veículo seja acionado sempre que um sensor de luminosidade detectar baixa iluminação no ambiente (representada por uma saída 0) ou sempre que a velocidade do veículo ultrapassar os 80 km/h (representada por uma saída 1 em um sensor de velocidade). Caso o motorista desejar, ele poderá ligar o farol a qualquer momento. Caso também for

de seu agrado, ele poderá desativar esse sistema a qualquer momento, ficando apenas com o controle manual".

Aqui, a situação é um pouco mais complicada para ser resolvida por uma abordagem meramente intuitiva. Em primeiro lugar, o problema envolve quatro entradas: do sensor de luminosidade, do velocímetro, de uma chave que liga ou desliga o farol e de uma "chave mestra" que desliga o sistema.

Podemos projetar o primeiro nível da solução ao considerarmos que temos duas situações que podem motivar o acionamento automático do farol: a velocidade acima do limite determinado e a luminosidade abaixo do limite determinado. Com isso, já sabemos que o sensor de luminosidade deve ser ligado a uma porta NOT, e que essa porta NOT, bem como o velocímetro, devem ser ligadas em uma porta OR.

Porém, o sistema automático pode ser desligado a qualquer momento, fazendo com que apenas o sistema manual funcione. Por essa razão, o controle automático também deve estar associado a uma porta AND, que controla a permanência desse sistema.

Sabemos, também, que o farol pode ser acionado a qualquer momento pelo motorista. Portanto, parece razoável associarmos o controle do farol manual com o controle automático em outra porta OR.

Assim, temos o seguinte desenho:

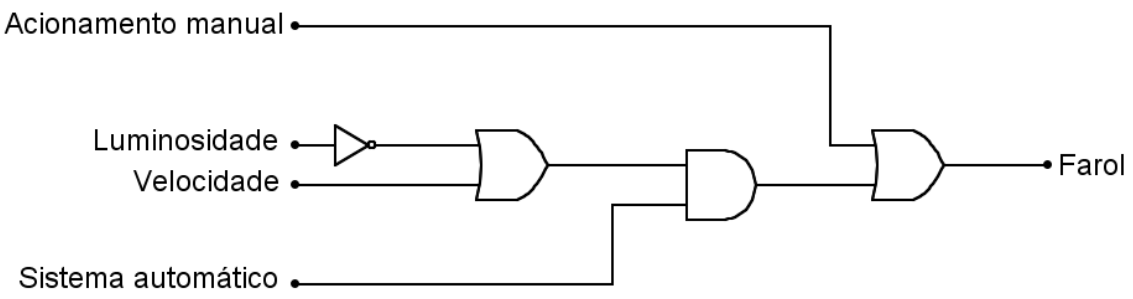


Figura 59: circuito que atende às especificações do problema.

6.3 EXPRESSÃO BOOLEANA

Circuitos podem também ser descritos por uma fórmula matemática, denominada expressão booleana, assim como portas lógicas. Para tanto, basta utilizar os operadores que já conhecemos.

Assim, circuito do alarme pode ser descrito da seguinte forma, assumindo que o sensor de presença é representado pela letra a e o sensor da porta é representado pela letra b:

$$alarme(a, b) = a + b$$

O circuito do aquecedor pode ser descrito da seguinte forma, assumindo que o relógio é representado pela letra r, e o sensor de temperatura é representado pela letra t:

$$aquecedor(r, t) = \bar{t} + r$$

Porém, o circuito do farol não é tão simples de ser descrito "à primeira vista". Para descrevê-lo, a melhor estratégia é dividir o circuito em partes menores. Para tanto, assumiremos que o acionamento manual é representado pela letra m, que o controle do sistema automático é representado pela letra s, que o sensor de luminosidade é representado pela letra l e que o velocímetro é representado pela letra v.

Primeiramente, podemos fazer a seguinte divisão, indicando a fórmula $f = m + S_1$.

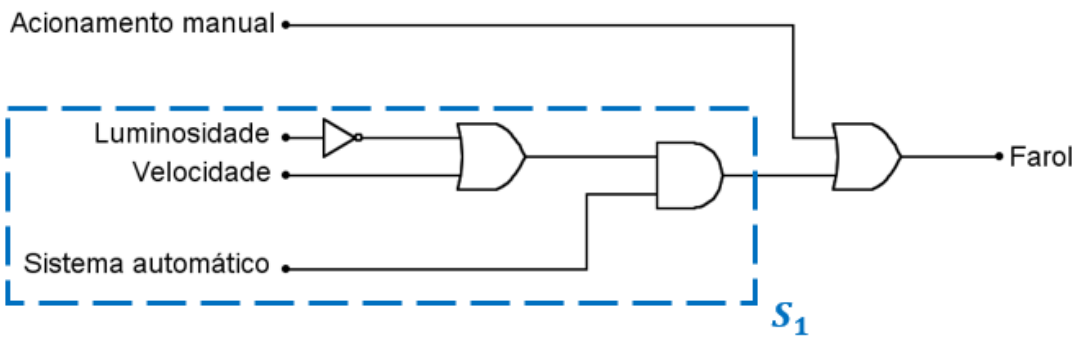


Figura 60: primeira divisão para obtenção da expressão booleana.

E0m S_1, podemos fazer mais uma divisão, indicando a fórmula $S_1 = s \,.\, T_1$.

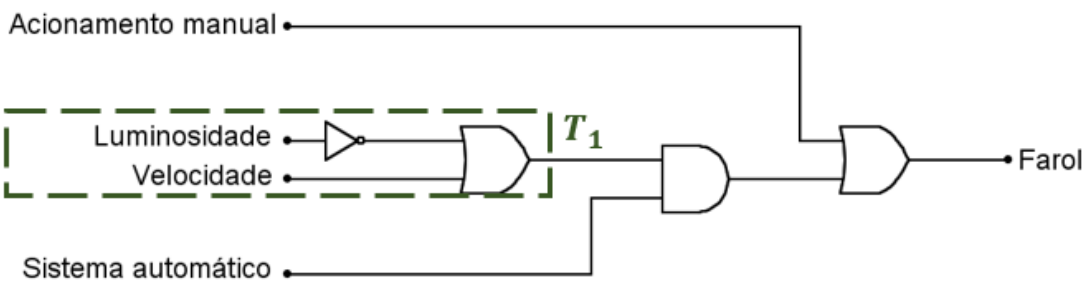


Figura 61: segunda divisão para obtenção da expressão booleana.

Daí, facilmente decorre que $T_1 = \bar{l} + s$. Assim, podemos retornar à expressão completa, substituindo cada divisão por seu real conteúdo:

$$farol(m, s, l, v) = m + \left(s \,.\, (\bar{l} + v)\right)$$

Com alguma prática, você logo conseguirá obter a expressão booleana de um circuito "à primeira vista". Porém, até lá, adote a estratégia de dividir o circuito em partes menores, até quando necessário, e depois substituir cada parte por seu real conteúdo.

6.4 TABELA-VERDADE

Outra possibilidade para se descrever o funcionamento de um circuito (assim como o de uma porta lógica) é através do uso de tabelas-verdade. Uma tabela-verdade é uma tabela que apresenta o comportamento de um circuito em cada uma das situações possíveis.

Para entendermos o que é uma tabela-verdade e como ela pode ser construída, acompanharemos o passo a passo para construção de tabela-verdade para o seguinte circuito, de expressão booleana $S(a, b, c) = a + b.c$

Figura 62: circuito de exemplo.

O primeiro passo para construção de qualquer tabela-verdade é a criação de várias colunas, uma responsável por cada entrada do circuito. Assim, temos:

Tabela 7: primeira versão da tabela-verdade.

a	b	c

Em seguida, devemos preencher as n colunas da nossa tabela com todas as entradas possíveis, uma em cada linha. Felizmente, podemos usar princípios matemáticos para se listar todas essas entradas.

Em primeiro lugar, o princípio fundamental da contagem nos indica que, para n entradas de um circuito, existem 2^n combinações de sinais possíveis. Isso acontece porque temos 2 sinais (1 e 0) para cada uma das entradas, então, temos um total de sinais dado por:

$$2 \times 2 \times 2 \times \ldots n \text{ vezes } \ldots \times 2$$

Para preencher a tabela, devemos preencher a coluna da mais a direita com os valores 1 e 0 alternando de 1 em 1, por 2^n vezes. Assim, temos:

Tabela 8: segunda versão da tabela-verdade.

a	b	c
		0
		1
		0
		1
		0
		1
		0
		1

Em seguida, preenchemos a segunda coluna, da direita para a esquerda, com valores de 1 e 0 em intervalos de 2 em 2, totalizando 2^n vezes. Assim, temos:

Tabela 9: terceira versão da tabela-verdade.

a	b	c
	0	0
	0	1
	1	0
	1	1
	0	0
	0	1
	1	0
	1	1

Depois, vamos repetindo o processo, visando preencher as colunas da forma que já conhecemos segundo o mesmo padrão. Assim, temos:

Tabela 10: quarta versão da tabela-verdade.

a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Em seguida, basta adicionarmos uma última coluna, correspondente a saída da função, e preenchermos com os valores obtidos em cada caso. Porém, essa conclusão nem sempre é imediata. Por essa razão, pode-se adicionar quantas mais colunas você achar necessário para que a tabela possa ser preenchida com facilidade.

Uma possibilidade, ainda, é recuperar a estratégia de modularização e colocar o próprio nome dos módulos como colunas da tabela. Essa prática, além de poupar espaço horizontal – o qual pode ficar bastante grande para casos mais complexos – também torna a tabela final muito mais compreensível. Veja:

Tabela 11: quinta versão da tabela-verdade.

a	b	c	$b.c$	$a + b.c$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Com base na tabela, podemos identificar facilmente as situações em que o farol está apagado e as situações em que o farol está ligado, bastando, apenas, verificar os valores 1 e 0.

Em suma, neste capítulo verificamos que é possível representar um circuito lógico de várias maneiras diferentes: na forma de um diagrama, na forma de uma expressão lógica e na forma de uma tabela verdade. Ao passo que as duas primeiras formas retratam a estrutura do circuito, a última retrata o comportamento deste circuito. Portanto, é possível que existam mais de um circuito com a mesma tabela-verdade.

6.5 IMPLEMENTANDO O CIRCUITO DO FAROL NA PRÁTICA

Planejar um circuito no papel e montá-lo em uma protoboard são duas atividades distintas e que exigem um certo planejamento. Nesta seção, verificaremos como o circuito do farol que discutimos na seção anterior pode ser implementado em uma protoboard utilizando as portas lógicas básicas.

O primeiro passo para a implementação é determinarmos o número de componentes necessário para o projeto. Neste caso, precisaremos de 1 porta NOT, 2 portas OR e 1 porta AND. Portanto, precisamos de 3 circuitos integrados, dos tipos CI 74HC04, CI 74HC08 e CI 74HC32. Temos 3 entradas, então precisaremos de 3

interruptores. Por fim, temos uma saída, então precisaremos de 1 LED e 1 resistor (um de 1000 Ω dará conta do recado).

Em seguida, devemos nomear os fios envolvidos na montagem, de forma que uma entrada e uma saída de um mesmo fio tenham o mesmo nome. Isso será útil para facilitar o processo de montagem. Se possível, atribua uma cor diferente para cada fio. Lembre-se que as saídas e entradas podem ser ligadas diretamente em um interruptor e LED, respectivamente, desprezando a necessidade de fios.

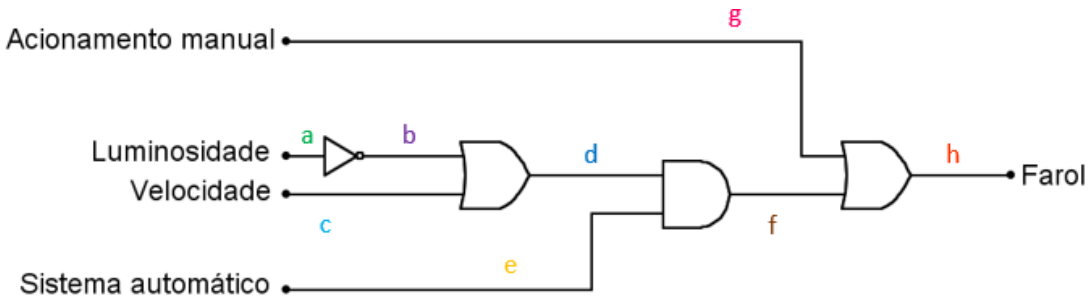


Figura 63: esquema do circuito com os fios a serem utilizados marcados.

Em seguida, o próximo passo envolve ligar as portas lógicas, os interruptores e o LED indicador final.

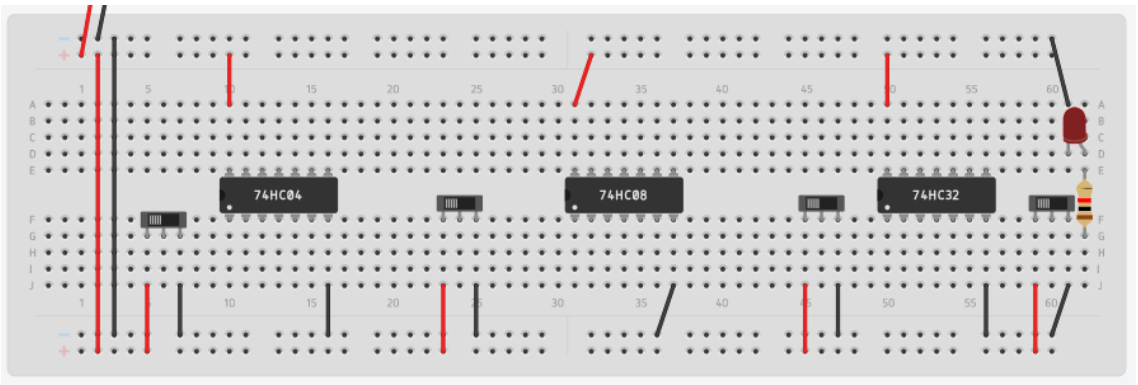


Figura 64: primeira versão do circuito na protoboard.

Depois, fazemos a ligação dos fios da forma prescrita no projeto:

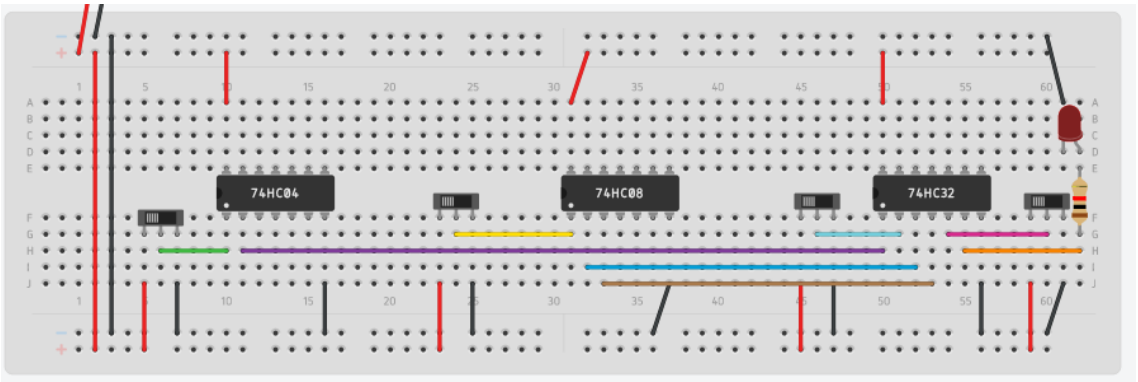


Figura 65: versão final do projeto na protoboard.

7 USANDO O SIMULADOR FALSTAD

Testar os conceitos estudados é uma peça fundamental em qualquer forma de aprendizado. Nesta seção, introduziremos o Falstad, uma ferramenta para simulação de circuitos elétricos e digitais.

O Falstad não pode ser considerado uma ferramenta profissional, uma vez que carece de diversas funções que seriam necessárias para o projeto comercial de circuitos. Porém, para estudo e compreensão, é uma excelente ferramenta.

7.1 DOWNLOAD

Você pode baixa-lo em https://github.com/NatanaelAntonioli/FalstadNovaVersao/, em uma versão compilada especialmente para Windows. Opcionalmente, você pode utilizar a versão web em https://www.falstad.com/circuit/. No momento da escrita deste livro, a versão enviada para a página do GitHub era a única disponível para o download mais atualizado desse programa.

Uma vez iniciado, você verá essa interface.

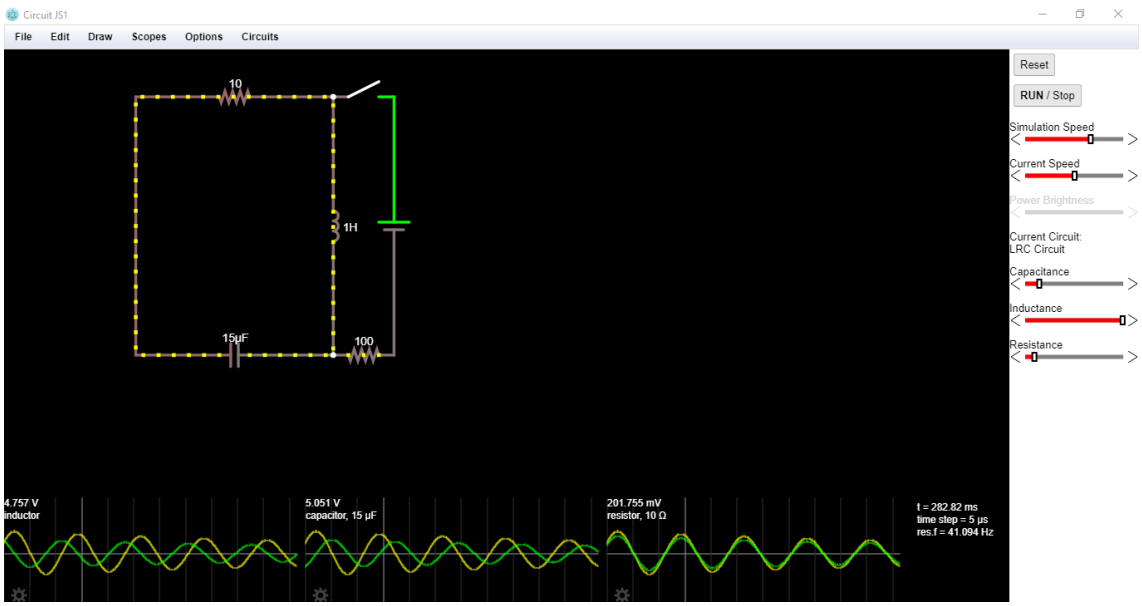


Figura 66: interface inicial do Falstad.

7.2 FUNÇÕES BÁSICAS

Na guia Options, você encontra diversas opções de configurações, como o uso da notação europeia para resistores, ou a escolha da convenção para os símbolos de portas lógicas. Para o nosso uso, consideraremos as seguintes configurações:

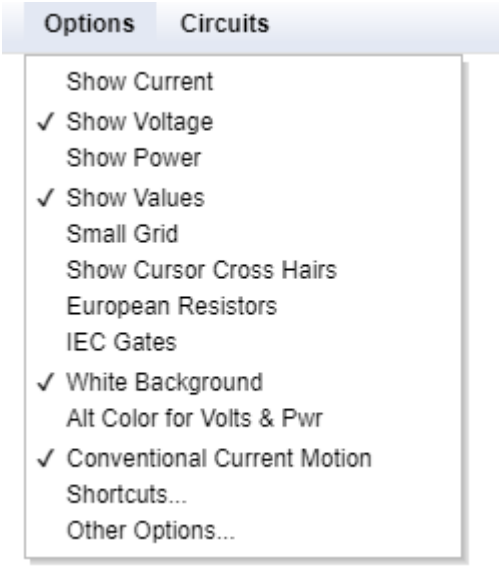


Figura 67: configuração utilizada no Falstad.

Podemos, então, começar a inserir os elementos envolvidos no circuito. Para inserir um sinal de entrada, basta utilizar a tecla “i” como atalho e, para um sinal de saída, basta utilizar a tecla “o”. Esses sinais ficarão verdes e exibirão a letra H quando estiverem com valor 1, e ficarão cinzas com a letra L quando estiverem com valor 0.

Figura 68: inserindo sinais de entrada e saída.

Podemos, então, clicar com o botão direito na tela para inserir outros elementos. Por enquanto, se atente apenas aqueles presentes na seção *Logic Gates, Input and Output*, pois é o que usaremos no momento. Você pode, também, utilizar os atalhos do teclado para inserir os componentes.

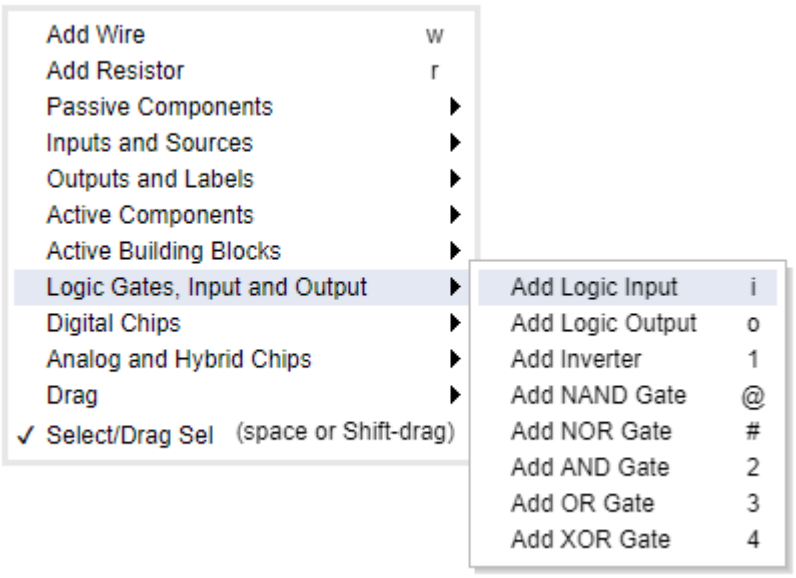


Figura 69: componentes que podem ser adicionados ao circuito.

Assim, podemos fazer uma porta lógica XOR da seguinte forma:

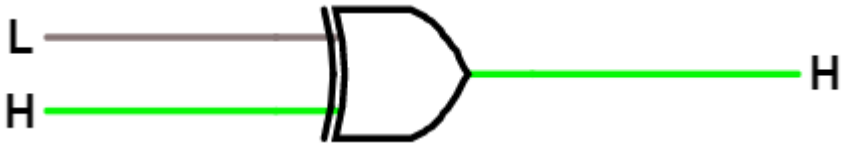


Figura 70: porta XOR.

E podemos testa-la da forma que preferirmos. O circuito do farol, por sua vez, pode ser feito exatamente da mesma forma. Veja:

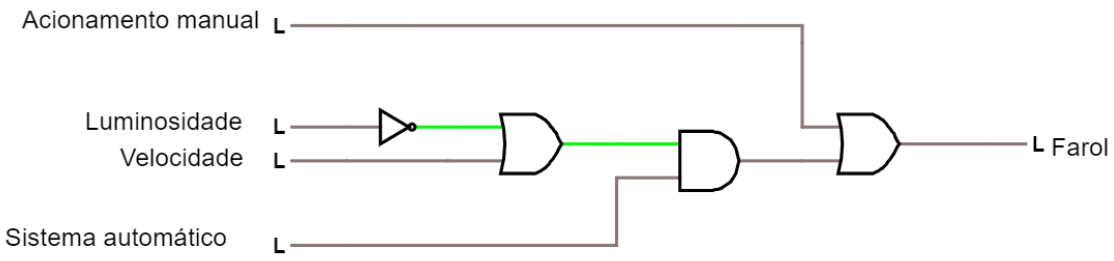


Figura 71: circuito do farol.

7.3 SUBCIRCUITOS

No futuro, verificaremos que diversos circuitos podem ser entendidos como "caixas pretas", dentro das quais algum processo ocorre. Isso possibilita, portanto, que estes circuitos possam ser encaixados em circuitos maiores sem prejudicar o entendimento do processo.

No Falstad, esses circuitos são chamados de subcircuitos. Para fazer um subcircuito, basta trocarmos os inputs e outputs por um *Labeled Node*, que pode ser encontrado na seção *Outputs and Labels*. Nomeamos cada nó com seu identificador.

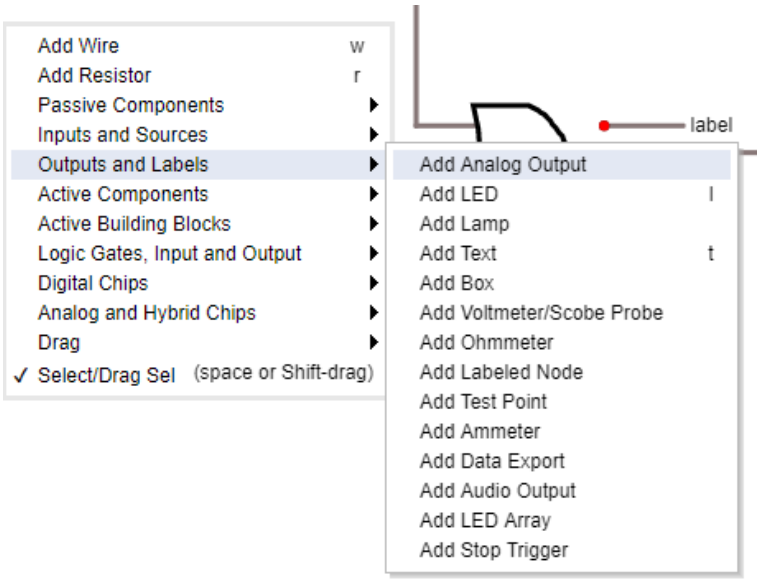


Figura 72: inserção de nós com identificador.

Assim, teremos:

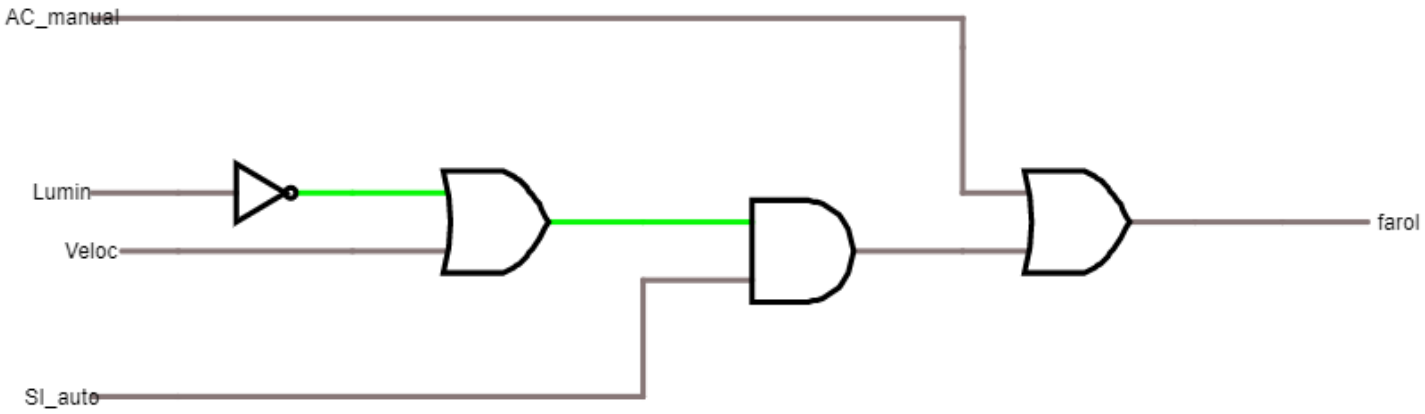


Figura 73: circuito do farol.

Então, vamos na seção File e selecionamos Create Subcircuit....

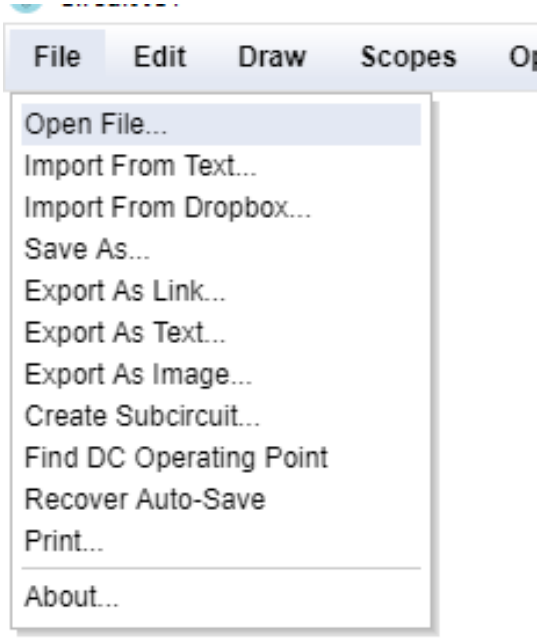


Figura 74: criando um subcircuito.

Finalmente, ajustamos a posição dos pinos de entrada e de saída do subcircuito da forma que desejarmos.

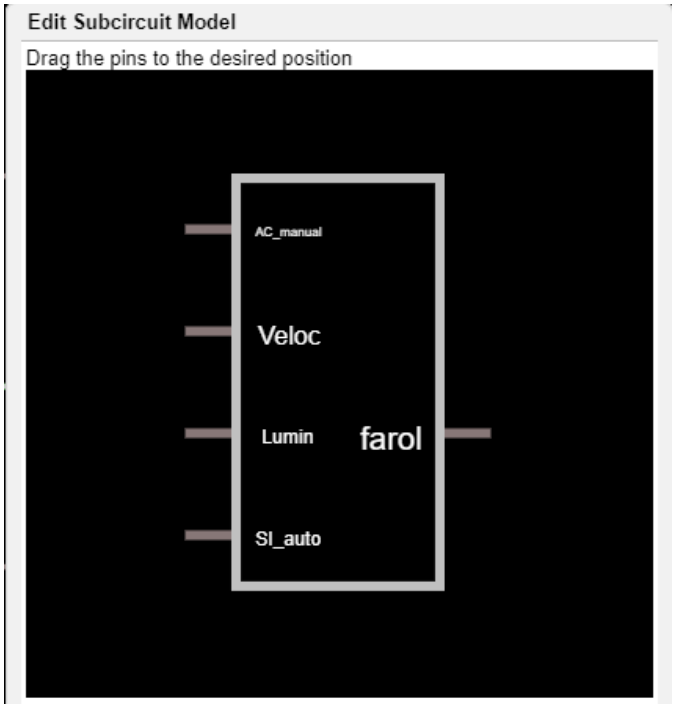


Figura 75: ajustando os pinos de entrada e saída.

Para adicionar o circuito criado, podemos clicar com o botão direito e adicionar o subcircuito pelo caminho Active building blocks > Add Subcircuit Instance.

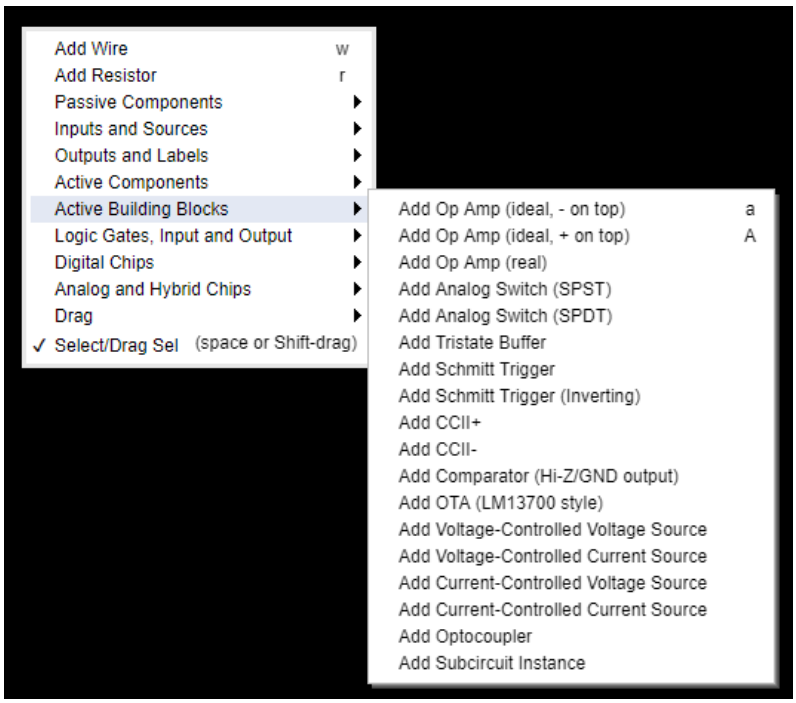


Figura 76: adicionando o subcircuito.

Finalmente, basta ligarmos os terminais do subcircuito (que agora funciona como uma “caixa preta”) em saídas e entradas para vermos o resultado.

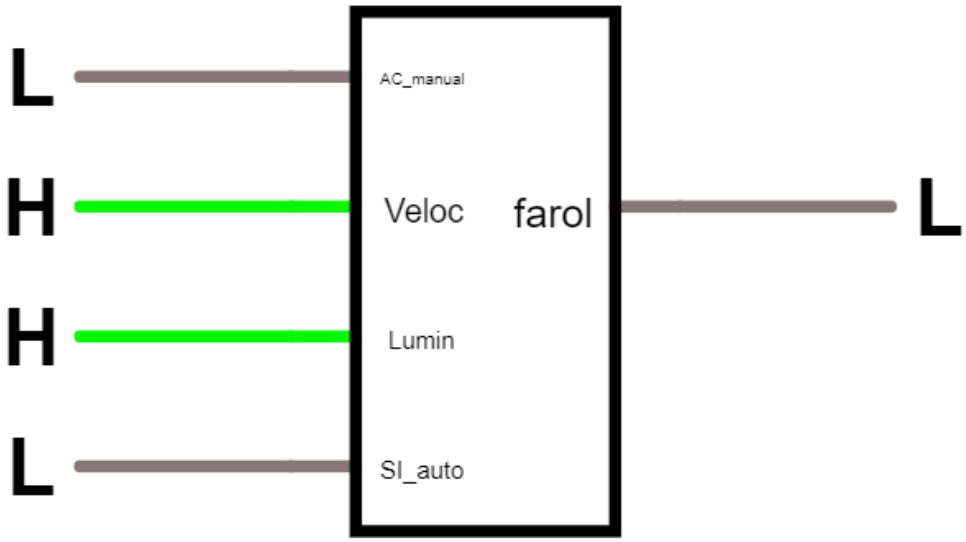


Figura 77: circuito do farol sendo implementado como uma "caixa preta".

8 ÁLGEBRA DE BOOLE

Sabemos que uma mesma tabela-verdade pode corresponder a vários circuitos que têm o mesmo comportamento. Quanto mais reduzido for um circuito (ou seja, menor número de portas envolvidas), menores serão os custos envolvidos em sua produção, menor será o atraso de propagação e menor será a tendência do circuito a superaquecer. Portanto, é importante possuirmos técnicas que permitam encontrar o menor circuito que corresponde a uma mesma tabela-verdade, denominado circuito mínimo.

8.1 AXIOMAS, TEOREMAS E PROPRIEDADES

Uma dessas técnicas envolve a compreensão da álgebra de Boole, um ramo da matemática no qual as variáveis envolvidas são valores-verdade, representados por 1 (verdadeiro) e 0 (falso). Na prática, as portas lógicas aqui estudadas representam operações em álgebra de Boole, e, através do domínio destas operações e de suas propriedades, somos capazes de encontrar expressões equivalentes.

Essas propriedades são dadas na forma de leis. A seguir, enunciaremos todas elas, em que p,q e r são variáveis binárias.

A álgebra de Boole está sustentada em alguns axiomas. São eles:

$$0.0 = 0$$
$$1.1 = 1$$
$$0.1 = 1.0 = 0$$
$$1 + 1 = 1$$
$$0 + 0 = 0$$
$$1 + 0 = 0 + 1 = 1$$
$$x = 0 \leftrightarrow \bar{x} = 1$$
$$x = 1 \leftrightarrow \bar{x} = 0$$

Existem, também, os teoremas que dizem respeito às operações envolvendo uma variável. São eles:

$$x.0 = 0$$

$$x.1 = x$$
$$x.x = x$$
$$x.\bar{x} = 0$$
$$\bar{\bar{x}} = x$$
$$x + 1 = 1$$
$$x + 0 = x$$
$$x + x = x$$
$$x + \bar{x} = 1$$

Existem, ainda, as propriedades comutativas:

$$x.y = y.x$$
$$x + y = y + x$$

As propriedades associativas:

$$x.(y.z) = (x.y).z$$
$$x + (y + z) = (x + y) + z$$

As propriedades distributivas:

$$x.(y + z) = x.y + x.z$$
$$x + y.z = (x + y).(x + z)$$

As leis de absorção:

$$x + x.y = x$$
$$x.(x + y) = x$$

As leis de combinação:

$$x.y + x.\bar{y} = x$$
$$(x + y).(x + \bar{y}) = x$$

Que podem ser provadas da seguinte forma:

$$x.y + x.\bar{y} = x.(y + \bar{y}) = x.1 = x$$

$$x.x + x.\bar{y} + x.y + y.\bar{y} = x + x.(y + \bar{y}) + 0 = x + x.1 = x$$

E as leis de De Morgan:

$$\overline{x.y} = \bar{x} + \bar{y}$$

$$\overline{x + y} = \bar{x}.\bar{y}$$

$$x + \bar{x}.y = x + y$$

$$x.(\bar{x} + y) = x.y$$

Na prática, podemos obter uma expressão equivalente a partir de uma expressão original por meio da manipulação algébrica, isto é, trabalhando a expressão original de forma a se obter uma expressão equivalente menor. Podemos facilmente olhar para uma expressão equivalente e verificar que ela é menor, mas não podemos saber dessa forma se ela é a menor expressão possível, o que torna esse método bastante ineficaz.

Ainda assim, é interessante saber que essas propriedades existem e como elas podem ser usadas.

8.2 PROVANDO A EQUIVALÊNCIA DE EXPRESSÕES LÓGICAS

Suponha, por exemplo, que queremos provar a seguinte equivalência:

$$(x_1 + x_3).(\overline{x_1} + \overline{x_3}) = x_1.\overline{x_3} + \overline{x_1}x_3$$

Uma possibilidade para essa prova é justamente através da tabela-verdade. Basta construirmos uma tabela-verdade para cada uma das expressões, como indicado a seguir.

x_1	x_3	$\overline{x_1}$	$\overline{x_3}$	$A = (x_1 + x_3)$	$B = (\overline{x_1} + \overline{x_3})$	$A.B$
0	0	1	1	0	1	0
1	0	0	1	1	1	1
0	1	1	0	1	1	1
1	1	0	0	1	0	0

Figura 78: tabela-verdade para o lado esquerdo (LHS).

x_1	x_3	$\overline{x_1}$	$\overline{x_3}$	$A = (x_1.\overline{x_3})$	$B = \overline{x_1}x_3$	$A + B$
0	0	1	1	0	0	0
1	0	0	1	1	0	1
0	1	1	0	0	1	1
1	1	0	0	0	0	0

Figura 79: tabela-verdade para o lado direito (RHS)

Outra possibilidade é através da manipulação algébrica. Aqui, devemos manipular os dois lados ao mesmo tempo, ou somente um dos lados por vez, ou somente um único lado, visando provar a igualdade estabelecida. No nosso exemplo, poderíamos seguir alguns passos:

Primeiro, aplicaremos a distributiva no LHS:

$$(x_1 + x_3)(\overline{x_1}) + (x_1 + x_3)(\overline{x_3}) =$$

Aplicando mais uma vez a mesma propriedade:

$$= (x_1\overline{x_1}) + (x_3\overline{x_1}) + (x_1\overline{x_3}) + (x_3\overline{x_3}) =$$

Usando os teoremas:

$$= 0 + (x_3\overline{x_1}) + (x_1\overline{x_3}) + 0 = (x_3\overline{x_1}) + (x_1\overline{x_3}) +$$

E a relação está provada.

Não existe uma forma única de se fazer isso, e não existe um algoritmo que sempre permita provar ou negar uma igualdade. Apenas a prática irá torna-lo mais apto para realizar esse tipo de processo.

9 FORMAS CANÔNICAS

Veremos agora que, partindo de uma tabela-verdade, existem duas formas canônicas que podem ser obtidas, e que permitem produzir um circuito com o mesmo funcionamento do funcionamento especificado na tabela-verdade.

9.1 MINTERMOS

Mintermos e maxtermos são utilizados para reescrever uma função lógica em uma forma padronizada no sentido de obter-se uma simplificação da mesma. Esta simplificação é traduzida na redução do número de portas do circuito lógico que implementa tal função. De uma tabela-verdade, podemos escrever o maxtermo e o mintermo de uma linha. Para isso considere, esse modelo de tabela-verdade com 3 variáveis e saídas correspondentes).

	a	b	c	S
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Figura 80: exemplo de tabela-verdade

Podemos escrever o mintermo de uma linha negando as variáveis que tem valor 0, mantendo as variáveis que tem valor 1, e realizando uma conjunção lógica entre elas. Assim, temos:

$$m_0 = \bar{a}.\bar{b}.\bar{c}$$
$$m_1 = \bar{a}.\bar{b}.c$$
$$m_2 = \bar{a}.b.\bar{c}$$

$$m_3 = \bar{a}.b.c$$
$$m_4 = a.\bar{b}.\bar{c}$$
$$m_5 = a.\bar{b}.c$$
$$m_6 = a.b.\bar{c}$$
$$m_7 = a.b.c$$

Assim, podemos escrever um circuito na forma canônica de mintermos representando-o na forma de soma dos produtos das linhas cuja saída da tabela-verdade é igual a 1. No nosso caso, temos:

$$\sum m = m_0 + m_4 + m_5 = \bar{a}.\bar{b}.\bar{c} + a.\bar{b}.\bar{c} + a.\bar{b}.c$$

Isso nos permite, portanto, produzir um circuito que tenha o mesmo comportamento da tabela verdade (apesar de não sabermos se esse circuito é mínimo ou não).

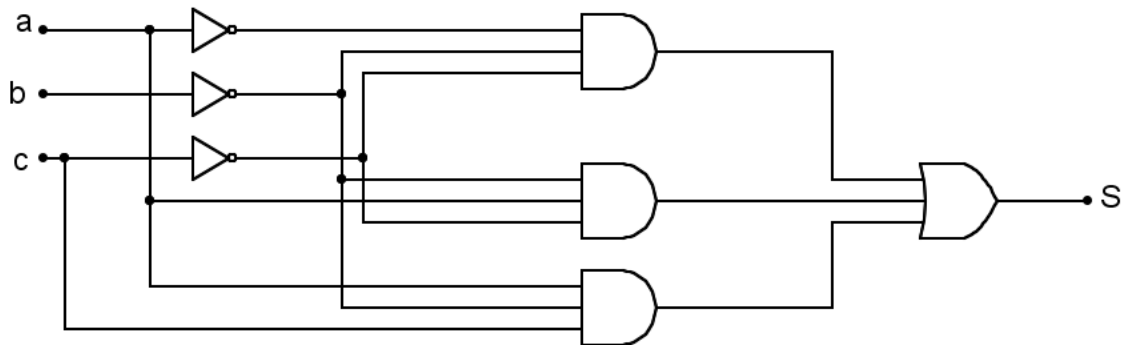


Figura 81: circuito obtido pela soma de mintermos.

9.2 MAXTERMOS

Podemos escrever o maxtermo de uma linha negando as variáveis que tem valor 1, mantendo as variáveis que tem valor 0, e realizando uma disjunção lógica entre elas. Assim, temos:

$$M_0 = x_1 + x_2 + x_3$$
$$M_1 = x_1 + x_2 + \overline{x_3}$$
$$M_2 = x_1 + \overline{x_2} + x_3$$
$$M_3 = x_1 + \overline{x_2} + \overline{x_3}$$

$$M_4 = \overline{x_1} + x_2 + x_3$$
$$M_5 = \overline{x_1} + x_2 + \overline{x_3}$$
$$M_6 = \overline{x_1} + \overline{x_2} + x_3$$
$$M_7 = \overline{x_1} + \overline{x_2} + \overline{x_3}$$

Assim, podemos escrever um circuito na forma canônica de maxtermos representando-o na forma de soma dos produtos das linhas cuja saída da tabela-verdade é igual a 1. No nosso caso, temos:

$$\sum M = M_1 . M_2 . M_3 . M_6 . M_7 =$$
$$= (x_1 + x_2 + \overline{x_3}) . (x_1 + \overline{x_2} + x_3) . (x_1 + \overline{x_2} + \overline{x_3}) . (\overline{x_1} + \overline{x_2} + x_3) . (\overline{x_1} + \overline{x_2} + \overline{x_3})$$

Isso nos permite, portanto, produzir um circuito que tenha o mesmo comportamento da tabela verdade (apesar de não sabermos se esse circuito é mínimo ou não).

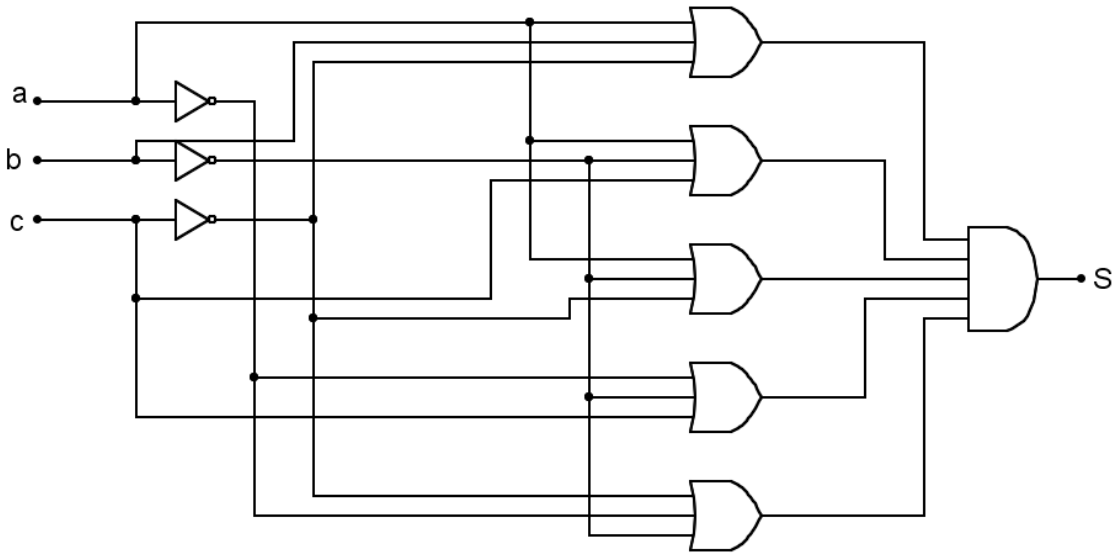


Figura 82: circuito obtido pelo produto de maxtermos.

Em suma, neste capítulo conhecemos duas formas de se obter um circuito a partir de uma tabela-verdade. Porém, ainda não somos capazes de determinar o circuito mínimo que pode ser obtido através da tabela-verdade.

10 MAPAS DE KARNAUGH

Mapa de Karnaugh são uma forma de se obter a expressão mínima a partir de uma tabela-verdade. Em relação à manipulação algébrica, mapas de Karnaugh são considerados superiores pela simplicidade e pela garantia de uma expressão mínima para situações de até 4 variáveis de entrada. Porém, para expressões com mais de 4 variáveis, esse método torna-se demasiadamente complexo.

A partir de um mapa de Karnaugh, podem ser obtidas duas expressões mínimas: a expressão na forma de produto das somas, e a expressão na forma de soma de produtos. Verificaremos como cada uma das expressões podem ser obtidas em diversas situações.

10.1 MAPAS DE DUAS VARIÁVEIS

O primeiro passo para a construção de um mapa de duas variáveis é numeração das linhas da tabela-verdade. Assim, temos:

Tabela 12: tabela-verdade de 2 variáveis.

n	a	b
0	0	0
1	0	1
2	1	0
3	1	1

Em seguida, devemos construir uma tabela 2x2, na qual as linhas correspondem aos valores possíveis de a, e as colunas correspondem aos valores possíveis de b. Os números pequenos correspondem às linhas da tabela-verdade.

Tabela 13: mapa de Karnaugh com 2 variáveis.

a \ b	**0**	**1**
0	0	1
1	2	3

Esses passos são comuns para qualquer tabela-verdade. Porém, o próximo passo envolve saber as saídas que a tabela-verdade específica possui. Suponha que tenhamos:

Tabela 14: exemplo de tabela-verdade com saídas.

n	a	b	S
0	0	0	1
1	0	1	0
2	1	0	1
3	1	1	1

Então, o mapa de Karnaugh preenchido com as saídas da tabela-verdade em cada posição numerada seria:

Tabela 15: mapa de Karnaugh preenchido com valores da tabela-verdade acima.

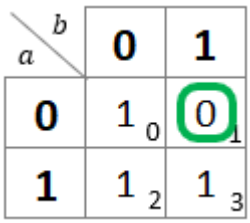
a \ b	**0**	**1**
0	1 0	0 1
1	1 2	1 3

Em seguida, devemos circular, na tabela, grupos de números iguais que possuam uma quantidade de números equivalente a uma potência de 2 (1, 2, 4, 8 ou 16). Podemos optar por circular os uns ou os zeros. Assim, para as duas possibilidades, teremos:

Tabela 16: mapa de Karnaugh com os uns circulados.

a \ b	**0**	**1**
0	1 0	0 1
1	1 2	1 3

Tabela 17: mapa de Karnaugh com os zeros circulados.



Quando optamos por circular apenas os uns, estamos trabalhando com uma implementação do tipo soma de produtos (SOP) e, quando optamos por circular apenas os zeros, estamos trabalhando com uma implementação do tipo produto das somas (POS).

Em ambas as implementações, um grupo é representado pelas variáveis que ele engloba, de forma que variáveis que se apresentam na forma negada e original podem ser suprimidas. Na situação SOP, deve-se multiplicar as variáveis presentes dentro do grupo e somar os grupos, ao passo que, na situação POS, deve-se também multiplicar as variáveis presentes dentro do grupo, somar os grupos, e negar, por meio das leis de De Morgan, a expressão resultante.

Assim, na implementação SOP, o grupo vermelho agrupa as variáveis $\bar{b}, \bar{a}, a$ e, por conter as duas formas da variável a, ele é dado apenas pela variável $\bar{b}$. Pela mesma razão, o grupo azul é dado pela variável a. Deve-se, então, realizar a soma das variáveis dos grupos e depois negar o resultado, e a expressão mínima é dada por $a + \bar{b}$.

Já na implementação POS, o único grupo existente é o verde, e ele agrupa as variáveis $\bar{a}$ e b. Portanto, a soma dos produtos é dado por $\bar{a}.b$. Porém, o último passo da implementação POS nos diz que o resultado final é $a + \bar{b}$.

Os mapas de Karnaugh com 3 ou quatro variáveis têm o mesmo funcionamento nas implementações SOP e POS, porém se diferem na forma como o mapa é disposto.

Para tanto, consideraremos a seguinte tabela-verdade de 3 variáveis, já numerada e com exemplos de saídas.

Tabela 18: tabela-verdade de 3 variáveis já preenchidas com exemplos.

n	a	b	c	S
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

O mapa de Karnaugh resultante corresponde a uma tabela 2x4, que contém todas as possibilidades que as variáveis podem obter. Porém, para permitir que o mapa seja circulado, é necessário dispor as variáveis da seguinte forma:

Tabela 19: disposição das variáveis em um mapa de 3 entradas.

c \ ab	**00**	**01**	**11**	**10**
0	0	2	6	4
1	1	3	7	5

E, então, podemos preencher os valores:

Tabela 20: disposição das variáveis em um mapa de 3 entradas com linhas numeradas.

c \ ab	**00**	**01**	**11**	**10**
0	0 $_0$	0 $_2$	1 $_6$	1 $_4$
1	1 $_1$	1 $_3$	0 $_7$	1 $_5$

Em seguida, já podemos selecionar as duas implementações (SOP e POS). Aqui, há um detalhe adicional: pode-se fazer círculos que ultrapassam os limites da tabela e continuam no outro lado. Veja:

Tabela 21: solução SOP do mapa com 3 variáveis.

c \ ab	00	01	11	10
0	0 0	0 2	1 6	1 4
1	1 1	1 3	0 7	1 5

Nesse caso, os grupos verde, roxo e azul são, respectivamente, dados por $\bar{a}c$, $a\bar{c}$ e $a\bar{b}$. Assim, a implementação final é dada por $\bar{a}c + a\bar{c} + a\bar{b}$.

Tabela 22: solução POS do mapa com 3 variáveis.

c \ ab	00	01	11	10
0	0 0	0 2	1 6	1 4
1	1 1	1 3	0 7	1 5

Nesse caso, os grupos amarelo e verde são, respectivamente, dados por $\bar{a}.\bar{c}$ e $a.b.c$. Assim, a implementação de soma de produtos é dada por final é dada por $(\bar{a}.\bar{c}) + (a.b.c)$ e o resultado final é dado por $(a + c).(\bar{a} + \bar{b} + \bar{c})$.

Por fim, consideraremos a seguinte tabela-verdade de 4 variáveis, já numerada e com exemplos de saídas.

Tabela 23: tabela-verdade de 4 variáveis já preenchidas com exemplos.

n	a	b	c	d	S
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0

8	1	0	0	0	1
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	1
14	1	1	1	0	1
15	1	1	1	1	0

O mapa de Karnaugh resultante corresponde a uma tabela 4x4, que contém todas as possibilidades que as variáveis podem obter. Porém, para permitir que o mapa seja circulado, é necessário dispor as variáveis da seguinte forma:

Tabela 24: disposição das variáveis em um mapa de 4 entradas.

ab \ *cd*	**00**	**01**	**11**	**10**
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Preenchendo os valores das linhas:

Tabela 25: preenchimento das variáveis em um mapa de 4 entradas.

ab \ *cd*	**00**	**01**	**11**	**10**
00	0_0	0_1	0_3	0_2
01	0_4	0_5	0_7	1_6
11	1_{12}	1_{13}	1_{15}	1_{14}
10	1_8	1_9	1_{11}	1_{10}

O resultado da implementação SOP seria:

Tabela 26: disposição das variáveis em um mapa de 3 entradas.

ab \ cd	00	01	11	10
00	0_0	0_1	0_3	0_2
01	0_4	0_5	0_7	1_6
11	1_{12}	1_{13}	1_{15}	1_{14}
10	1_8	1_9	1_{11}	1_{10}

Assim, o grupo roxo é dado apenas pela variável a, e o grupo verde é dado pelas variáveis $b.c.\bar{d}$ e, portanto, a implementação SOP é dada por $a + (b.c.\bar{d})$.

O resultado da implementação POS seria:

Tabela 27: disposição das variáveis em um mapa de 3 entradas.

ab \ cd	00	01	11	10
00	0_0	0_1	0_3	0_2
01	0_4	0_5	0_7	1_6
11	1_{12}	1_{13}	1_{15}	1_{14}
10	1_8	1_9	1_{11}	1_{10}

Assim, o grupo roxo é dado pelas variáveis $\bar{a}.\bar{b}$, o grupo azul é dado pelas variáveis $\bar{a}\bar{c}$ e o grupo verde é dado $\bar{a}d$. Assim, a soma dos produtos é dada por $\bar{a}.\bar{b} + \bar{a}\bar{c} + \bar{a}d$ e, portanto, a implementação POS é dada por $(a + b).(a + b).(a + \bar{d})$.

10.2 DIRETRIZES GERAIS PARA CIRCULAR GRUPOS

Já sabemos o que deve ser feito, mas, talvez, você ainda tenha algumas dúvidas a respeito de como circular conjuntos. Portanto, seguem algumas diretrizes:

1. As regiões circuladas sempre precisam conter um número de uns ou zeros potência de dois, como 1, 2, 4, 8 e 16.
2. As únicas formas permitidas (considerando-se quando os grupos extrapolam as margens da tabela) são os retângulos e quadrados.
3. Os grupos devem ter o maior tamanho possível.
4. A implementação mínima é aquela que usa o menor número de grupos.

10.3 DON'T CARES

Até agora, lidamos com tabelas-verdade nas quais as saídas eram definidas para todas as situações de entrada. Porém, imagine, agora, a seguinte implementação:

"Dois sensores informam a temperatura de uma sala. Um deles, a retorna 1 sempre que a temperatura for maior que 16°C, e o outro, b retorna 1 sempre que a temperatura for maior que 18°C. Construa um circuito que devolva 1 sempre que a temperatura estiver entre 16°C e 18°C."

O problema pode ser especificado conforme a seguinte tabela-verdade:

Tabela 28: tabela-verdade para o problema do monitoramento de temperatura.

n	a	b	S
0	0	0	0
1	0	1	?
2	1	0	1
3	1	1	0

Porém observe que a condição na linha 1 nunca se concretizará, uma vez que é impossível que a temperatura em um ambiente seja maior que 18ºC e, ao mesmo tempo, menor que 16ºC. Isso significa que o comportamento do nosso circuito nessa condição simplesmente não importa (*don't care*). Por isso, a linha 1 pode ser simplesmente marcada com uma letra d.

Tabela 29: tabela-verdade para o problema do monitoramento de temperatura com .

n	s_{16}	s_{18}	S
0	0	0	0
1	0	1	d
2	1	0	1
3	1	1	0

Isso significa que, ao trabalhar com o mapa de Karnaugh, podemos fazer com que *d* assuma qualquer valor, conforme a nossa conveniência. Assim, temos:

Tabela 30: mapa de Karnaugh com 2 variáveis preenchido com os números da tabela-verdade.

a \ b	**0**	**1**
0	0 $_0$	d $_1$
1	1 $_2$	0 $_3$

Ao resolver o mapa, podemos simplesmente assumir que os valores d possuem o valor que for mais conveniente para formar os maiores grupos possíveis. No caso acima, a presença de um *don't care* permite assumir que o valor lido seja 0 nessa situação, o que nos permite formar:

Tabela 31: mapa de Karnaugh resolvido com .

a \ b	**0**	**1**
0	0 $_0$	d $_1$
1	1 $_2$	0 $_3$

Assim, a expressão obtida é $\overline{(\bar{A} + B)} = A.\bar{B}$.

Considere, agora, o seguinte mapa de Karnaugh:

Tabela 32: disposição das variáveis em um mapa de 4 entradas com *cares*.

ab \ cd	00	01	11	10
00	1_0	1_1	1_3	0_2
01	1_4	d_5	d_7	0_6
11	0_{12}	0_{13}	0_{15}	0_{14}
10	0_8	0_9	0_{11}	0_{10}

Nesse caso, os *don't cares* podem ser consideramos como 1, de forma a fazer:

Tabela 33: disposição das variáveis em um mapa de 3 entradas.

ab \ cd	00	01	11	10
00	1_0	1_1	1_3	0_2
01	1_4	d_5	d_7	0_6
11	0_{12}	0_{13}	0_{15}	0_{14}
10	0_8	0_9	0_{11}	0_{10}

Assim, a expressão resultante na forma SOP passa a ser dada por $\bar{a}\bar{c} + \bar{a}d$.

10.4 SOFTWARES PARA SOLUÇÃO DE MAPAS DE KARNAUGH

Resolver um mapa de Karnaugh definitivamente não é das experiências mais prazerosas, e resolver mapas de Karnaugh na mão com mais de 4 variáveis são um verdadeiro desafio. Porém, existem vários programas que resolvem tais mapas utilizando algoritmos computacionais, e que você deve utilizar quando necessário.

Um deles é o Karnaugh Map Minimizer, que pode ser baixado em https://sourceforge.net/projects/k-map/. Ele permite a resolução de mapas de até 8 variáveis, e suporta *don't cares*.

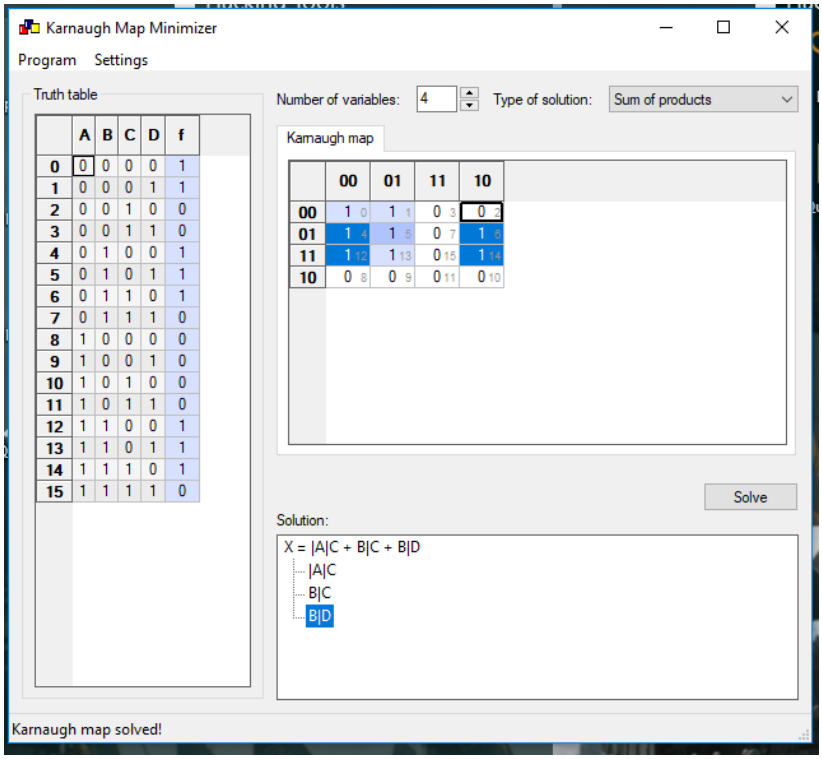


Figura 83: interface do Karnaugh Map Minimizer

11 A REPRESENTAÇÃO BINÁRIA

Já sabemos como trabalhar com sinais binários, que podem assumir valores 1 ou 0. Porém, ainda não sabemos como representar números com esses sinais. A melhor maneira de começarmos esse tópico é recordando como outro sistema, com o qual estamos muito familiarizados, funciona: o sistema decimal.

11.1 RECORDANDO: O SISTEMA DECIMAL

O primeiro passo para escrever números em binário é justamente recordar o funcionamento do sistema decimal. O sistema decimal é composto por dez algarismos, 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9, cada um deles representando uma dada quantidade de itens. A base de nosso sistema é, portanto, o número 10. Assim, temos:

$$1708 = 1 \times 10^3 + 7 \times 10^2 + 0 \times 10^1 + 8 \times 10^0 = 1000 + 700 + 0 + 8 = 1708$$

O mesmo princípio vale para o sistema binário: o sistema binário é composto por dois algarismos, 0 e 1 e sua base é, portanto, o número 2, ou 10, em binário. Isso significa que temos:

$$11101 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 16 + 8 + 4 + 0 + 1 = 29$$

11.2 ESCREVENDO EM BINÁRIO

Podemos escrever um número em binário através de um algoritmo prático: deve-se dividir o número por 2 sucessivamente, e anotar o resto de cada divisão. O número será formado pela aglutinação de todos os restos, com o último resto obtido mais à esquerda, e o primeiro mais à direita. Assim, temos o dispositivo prático:

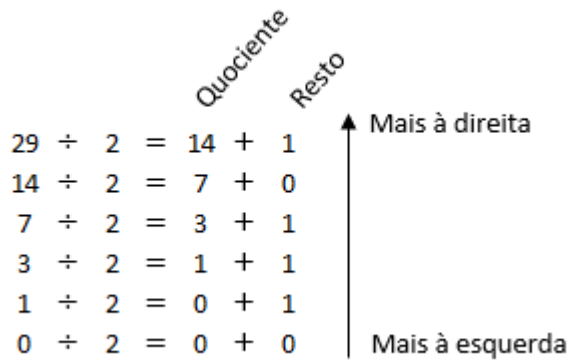


Figura 84: dispositivo prático para conversão de decimal em binário.

Para fazer o caminho inverso, também temos um dispositivo prático: deve-se escrever o número binário na vertical, e multiplicar o enésimo algarismo do número binário (de cima para baixo) por 2^n e somar os resultados.

$$
\begin{array}{rrrr}
1 & \times 2\text{^} & 0 = & 1 \\
0 & \times 2\text{^} & 1 = & 0 \\
1 & \times 2\text{^} & 2 = & 4 \\
1 & \times 2\text{^} & 3 = & 8 \\
1 & \times 2\text{^} & 4 = & 16 \\
0 & \times 2\text{^} & 5 = & 0 \\
 & & & \hline
 & & & 29
\end{array}
$$

Figura 85: dispositivo prático para conversão de binário em decimal.

Em um número binário, chamamos o bit mais a esquerda de bit mais significativo, ou MSB (*most significant bit*), e chamamos o bit mais a direita de bit menos significativo, ou LSB (*less significant bit)*. Chamamos um conjunto de 8 bits de um byte.

Dada uma quantidade de b bits, o número n de números binários que podemos representar utilizando esses bits é dado pela fórmula $n = 2^b$ e corresponde aos números entre 0 e $2^b - 1$. Assim, com 4 bits, podemos representar 16 números, entre 0 e 15. São eles:

Tabela 34: números decimais e binários entre 0 e 15.

0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

Note que, para representar o número 16, precisaríamos de mais um bit. Portanto, esse número seria denotado por 10000.

11.3 N ADIÇÃO DE NÚMEROS BINÁRIOS

Em decimal, estamos acostumados a realizar operações de adição da seguinte forma: para somar $27 + 85$, primeiro fazemos $7 + 5 = 12$, então colocamos o número 2 no resultado e passamos o número 1 para a unidade seguinte, fazendo, então, $2 + 8 + 1 = 11$, e, mais uma vez, passando o número 1 para a unidade seguinte, fazendo, finalmente $1 + 0 + 0 = 1$ e formando, assim, o número 112.

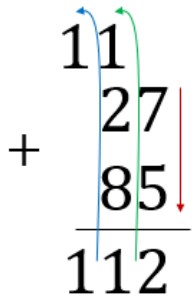


Figura 86: operação de soma em decimais.

Por se tratarem de sistemas multiplicativos, é exatamente dessa forma que a adição com números binários funciona. Considere, assim, a soma entre $11101 + 111010$,

ambos os números em binários. A forma de fazer isso envolve, primeiro, “armarmos” a conta da mesma forma que faríamos com números decimais.

$$\begin{array}{r} 11101 \\ +\ 111010 \\ \hline \end{array}$$

Figura 87: "armando" a conta com binários.

Em seguida, somamos as três primeiras colunas: $1 + 0 = 1, 0 + 1 = 1$ e $1 + 0 = 1$. Em todas essas, não passamos nenhum número para cima, uma vez que o resultado “cabe” em um único dígito.

$$\begin{array}{r} 11101 \\ +\ 111010 \\ \hline 111 \end{array}$$

Figura 88: primeiras colunas somadas.

Em seguida, somamos a quarta coluna. Nessa coluna, temos $1 + 1 = 2$, só que não podemos utilizar o número 2 ao trabalhar com binários. Portanto, temos $1 + 1 = 10$. Esse número não “cabe” em um único dígito e, portanto, devemos passa-lo para cima:

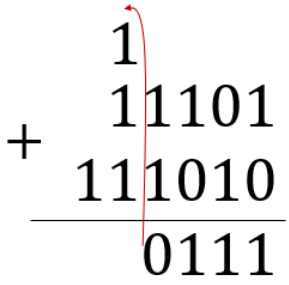


Figura 89: quarta coluna somada.

Ao somar a próxima, fazemos $1 + 1 + 1 = 3$. Pela mesma razão, não podemos utilizar o número 3 ao trabalhar com binários. Portanto, temos $1 + 1 + 1 = 11$. Esse número também não "cabe" em um dígito e, portanto, devemos passa-lo para cima:

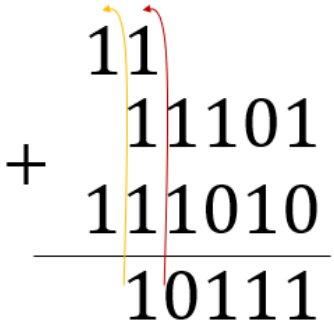


Figura 90: quinta coluna somada.

E, assim, fazemos o mesmo processo até o final.

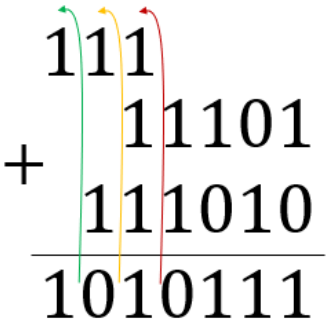


Figura 91: soma terminada.

11.4 A FORMA DE COMPLEMENTO DE 2

No sistema decimal, utilizamos o sinal de menos na frente de um número para indicar que este é negativo. Porém, em sistemas digitais, não existem artifícios para inserir este sinal na frente de um número binário negativo.

Ainda assim, evidentemente precisamos trabalhar com binários negativos. Também é interessante que a operação de adição possa funcionar como uma operação de subtração quando um número negativo é adicionado em um dos fatores a serem somados.

A forma escolhida para tal representação é a de complemento de 2. Ao escrever um número de n bits nessa forma, iremos utilizar o bit mais significativo como um indicador de sinal. Isso significa que, a partir de agora, números positivos possuirão um zero à esquerda em toda situação. Assim, o número 4 precisa, agora, de 4 bits (ao invés de 3) para ser representado, da forma 0100.

Um número negativo pode ser obtido na forma de complemento de dois por um algoritmo prático: deve-se inverter todos os bits (trocando 1 por 0 e 0 por 1) e, em seguida, somar 1 ao resultado invertido. Assim, temos:

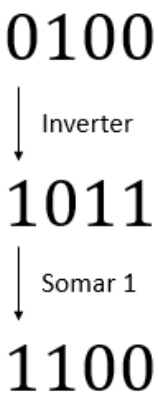


Figura 92: procedimento para obter número em complemento de 2.

O processo para converter um número em complemento de 2 para binário sem sinal (de forma a descobrir qual número corresponde) é exatamente o mesmo: primeiro invertê-lo, depois somar 1. Veja:

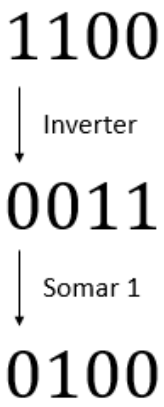


Figura 93: procedimento para obter número em binário sem sinal.

A grande vantagem de utilizarmos essa forma de representação é que, agora, podemos subtrair números binários da mesma forma que os somamos, utilizando exatamente o mesmo processo. Veja, por exemplo, a operação $6 - 4 = 2$:

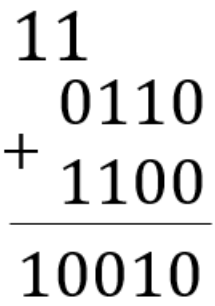


Figura 94: subtração 6 - 4 = 2.

Aqui, temos um detalhe importante: esperávamos um resultado com 4 bits (três para o número, um para o sinal), mas obtivemos um resultado composto por 5 bits. Esse último bit é denominado *ignore* e, como o nome diz, deve ser ignorado sempre que o resultado retorne um número para além do bit de sinal.

11.5 OVERFLOW ARITMÉTICO

Algumas vezes, somamos dois números cujo resultado excede a quantidade de bits que podemos usar. Suponha, por exemplo, que queiramos somar 4 e 6 de forma a obter um resultado que caiba em 4 bits com sinal. Teríamos a seguinte operação:

$$\begin{array}{r} 0100 \\ +\ 0110 \\ \hline 1010 \end{array}$$

Figura 95: resultado da soma entre 4 e 6.

O resultado esperado seria 10. E, de fato, se esperarmos um número sem sinal, esse resultado se confirma. Porém, esse não é o nosso caso, e o número devolvido foi -6, um resultado totalmente incoerente. Quando isso acontece, chamamos de overflow aritmético. Na prática, o overflow aritmético ocorre se e somente se o resultado possuir o sinal contrário do sinal esperado na operação.

11.6 *CARRY IN* E *CARRY OUT*

Em circuitos combinacionais de adição, trabalharemos com os conceitos de *carry in* e *carry out*. Dizemos que, em uma operação de soma de 1 bit, o *carry out* é o número que é "passado para cima" na próxima operação, e o *carry in* é o número que é recebido de uma operação anterior.

12 CIRCUITOS COMBINACIONAIS NOTÁVEIS

Já sabemos como circuitos podem ser construídos e descritos em linguagens de descrição de hardware (HDL). Sabemos, também, que circuitos podem ser vistos como módulos individuais, como "caixas fechadas" que recebem sinais de entrada e sinais de saída. É, então, o momento de conhecer alguns dos circuitos mais utilizados.

12.1 DISPLAYS E DECODERS

Já sabemos que, em lógica digital, trabalhamos com o sistema binário. Conhecemos, também, alguns circuitos que fazem operações matemáticas. Porém, uma pessoa que faça uso de um dispositivo que projetarmos pode desejar visualizar os resultados devolvidos pelo circuito na forma decimal, e não binária.

Uma maneira de exibir números para um usuário na forma decimal é por meio do uso de displays de 7 segmentos. Um dispositivo do tipo permite exibir os números de 0 a 9 e, adicionalmente algumas letras do alfabeto com apenas 7 ou 8 LEDs. 7 LEDs são usados para cada uma das partes do número a ser exibido, ao passo que um LED adicional pode ser usado para representar uma vírgula.

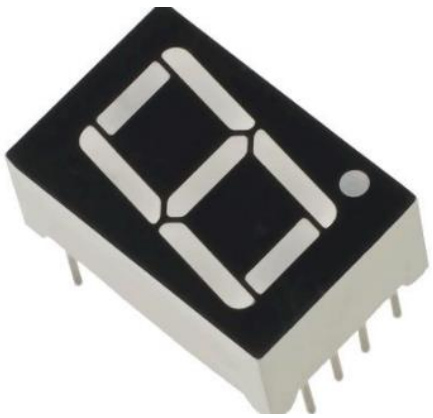


Figura 96: display de 7 segmentos.

Assim, esse dispositivo pode ser visto como um circuito que recebe tensão em 8 ou 7 pinos e, conforme os pinos energizados, ilumina os segmentos correspondentes a cada um dos pinos.

No simulador Falstad, esse componente possui 7 terminais (ou 8 na variedade com vírgula, o qual não utilizaremos). Esses terminais são indicados com letras de *a* até *e*, e correspondem aos LEDs do display conforme a figura abaixo.

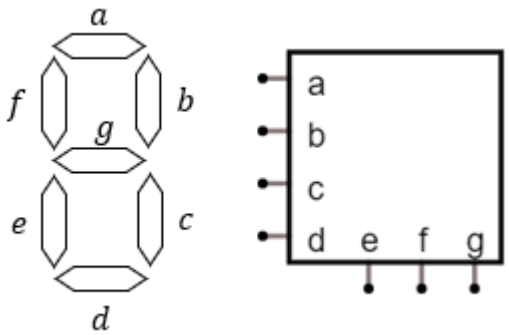


Figura 97: display de 7 segmentos e terminais correspondentes.

Um decoder é um circuito que recebe um número binário (de até 4 bits) e devolve, em seus 7 terminais, quais terminais devem ser energizados para se representar o número desejado no display. A maioria dos decoders permite representar números de 0 a 9, mas alguns ainda exibem os números 10,11,12,13,14 e 15 com as letras A,B,C,D,E e F.

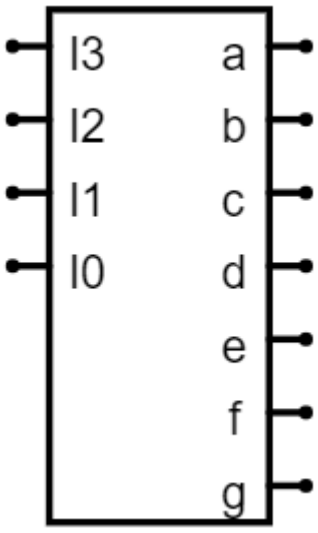


Figura 98: decoder de 7 segmentos.

No simulador, podemos montar um circuito que recebe um número de 4 bits e o converte para decimal (exibindo no display) da seguinte forma:

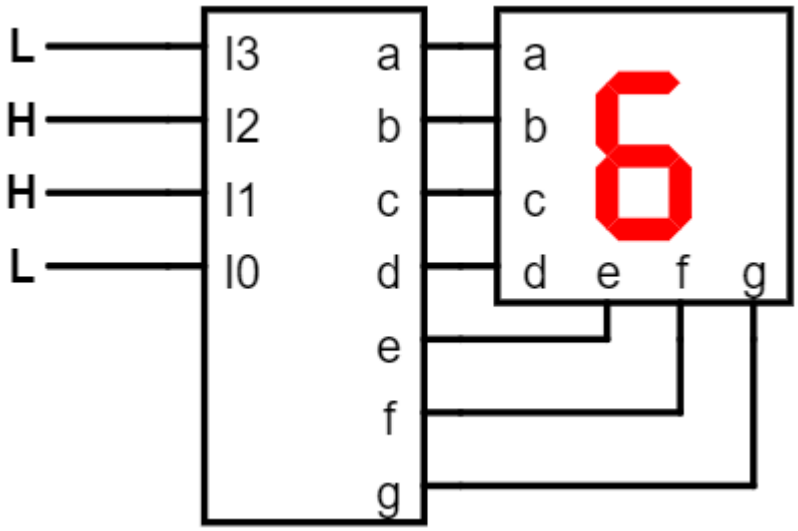


Figura 99: decoder e display de 7 segementos associados.

12.2 MULTIPLEXADORES

Um multiplexador simples é um circuito que recebe dois canais de entrada e um seletor, e devolve uma única saída (daí o nome *mux2to1*). Sua função é de encaminhar o sinal de um dos canais de entrada para a saída, conforme o valor do seletor. Isso é feito por meio do circuito abaixo:

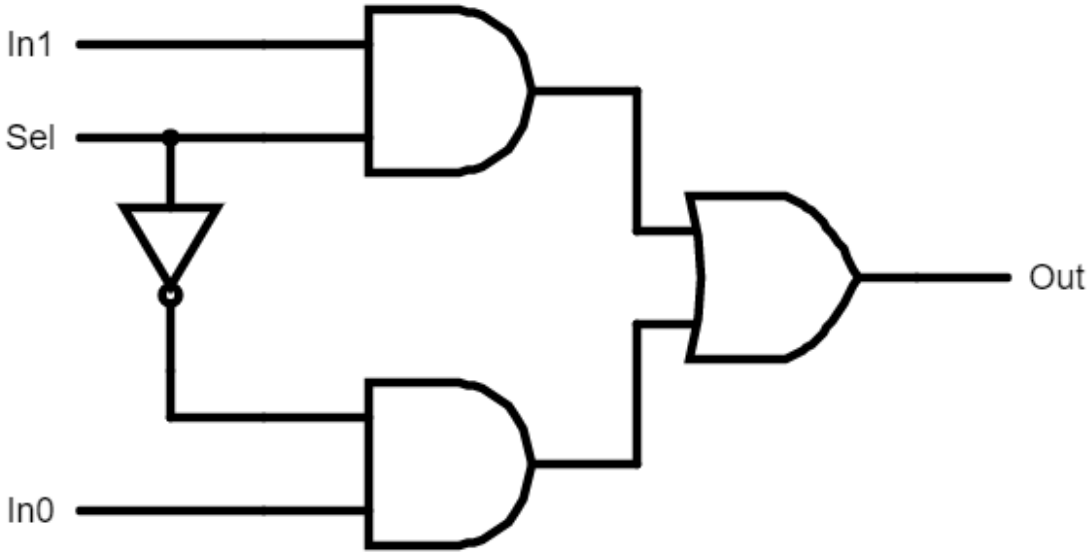


Figura 100: circuito do multiplexador de 2 para 1.

O multiplexador pode ser descrito pela seguinte tabela-verdade:

Sel	Out
0	x_1
1	x_2

Figura 101: tabela-verdade do multiplexador.

Assim, podemos lidar com o multiplexador como uma “caixa fechada” que se comporta segundo a tabela-verdade acima.

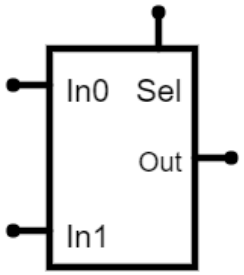


Figura 102: multiplexador como uma "caixa fechada".

Para n entradas de um seletor, o multiplexador suporta 2^n canais. Ou seja, podemos construir um *Mux4to1* usando 2 entradas de seletor e 4 entradas de canais ou, ainda, utilizando 3 *Mx2to1*. Para tanto, dois multiplexadores devem ser ligados de forma que eles recebam as 4 entradas e as saídas desses multiplexadores devem ser ligadas em um terceiro multiplexador. Assim, temos:

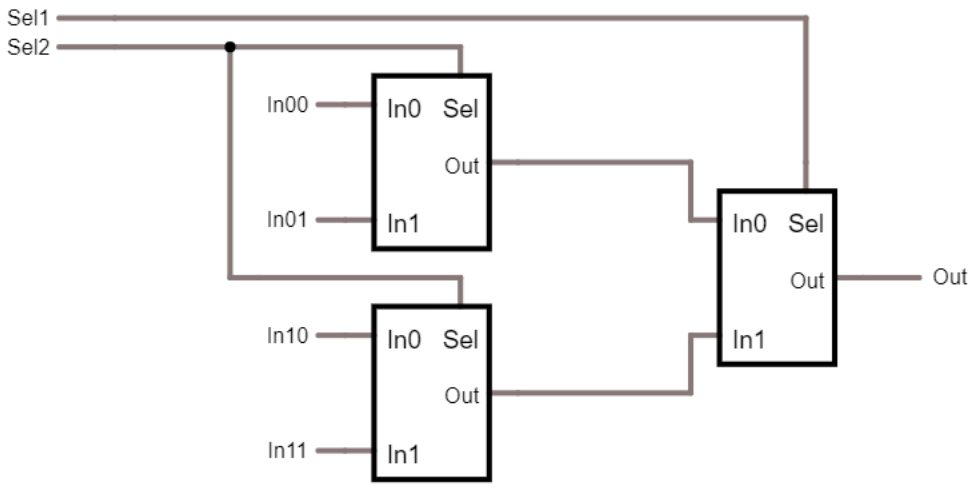


Figura 103: formação de um *mux4to1* usando 3 *mux2to1*.

Esse multiplexador pode ser descrito, também, pela seguinte tabela-verdade:

Tabela 35: tabela-verdade para o *mux4to1*.

$Sel1$	$Sel2$	S
0	0	In_0
0	1	In_{01}
1	0	In_{10}
1	1	In_{11}

12.3 SOMADORES -SUBTRATORES

Somadores-subtratores são circuitos que operam com n bits e são capazes de realizar as operações de soma ou subtração, conforme o número inserido. Tais operações são feitas na forma binária.

O circuito capaz de somar dois bits *A* e *B* (na forma normal ou de complemento de 2) e devolver um resultado e mais um carry-out é denominado *half-adder*, e pode ser descrito da seguinte forma:

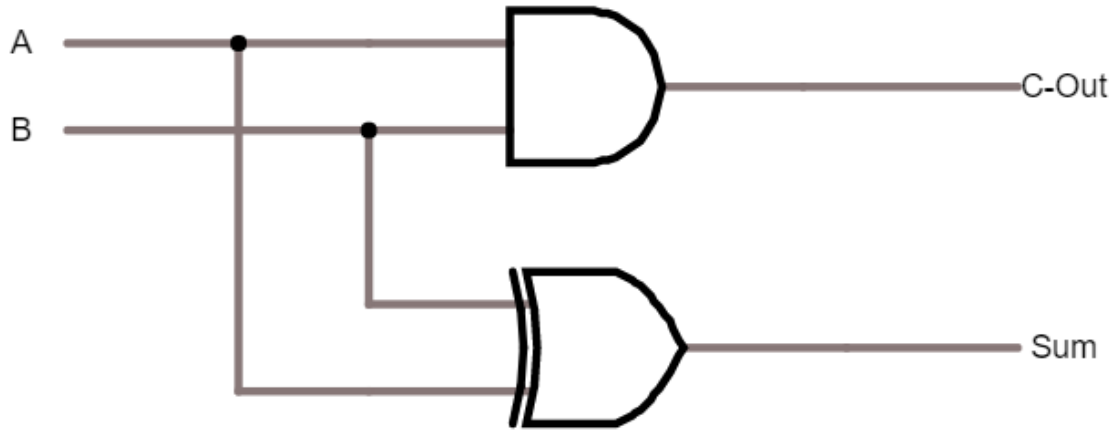


Figura 104: circuito *half-adder*.

Este circuito possui a seguinte tabela-verdade:

Tabela 36: tabela-verdade para o *half-adder*.

A	B	$C-out$	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Assim, podemos lidar com o *half-adder* como uma caixa fechada. Veja:

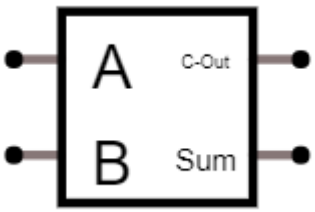


Figura 105: circuito *half-adder* fechado.

A partir de dois meio-somadores, podemos fazer um somador completo, que é capaz de receber três bits (dois de valores sendo somados e um carry-in) e devolver dois bits (o resultado da soma e o carry-out). Veja:

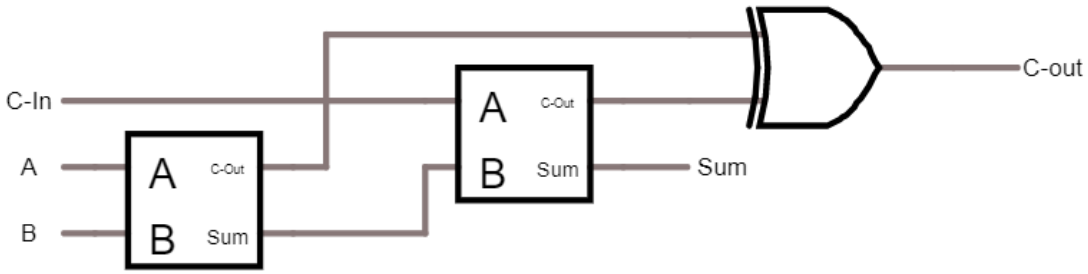


Figura 106: circuito *full-adder*.

Este circuito possui a seguinte tabela-verdade:

Tabela 37: tabela-verdade para o *full-adder*.

a	b	cin	$cout$	s
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

Da mesma forma, também podemos lidar com o somador completo como uma caixa fechada. Veja:

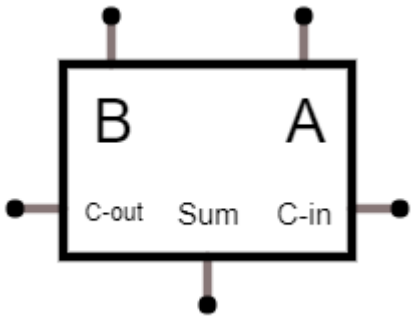


Figura 107: somador completo fechado.

Ao ligar o carry-in de um somador no carry-out de outro (e repetir esse processo n vezes), ligando, também, as entradas A e B de cada somador nos n bits correspondentes de dois números, temos um somador completo, no qual os dois números são somados e um carry out é devolvido. Veja:

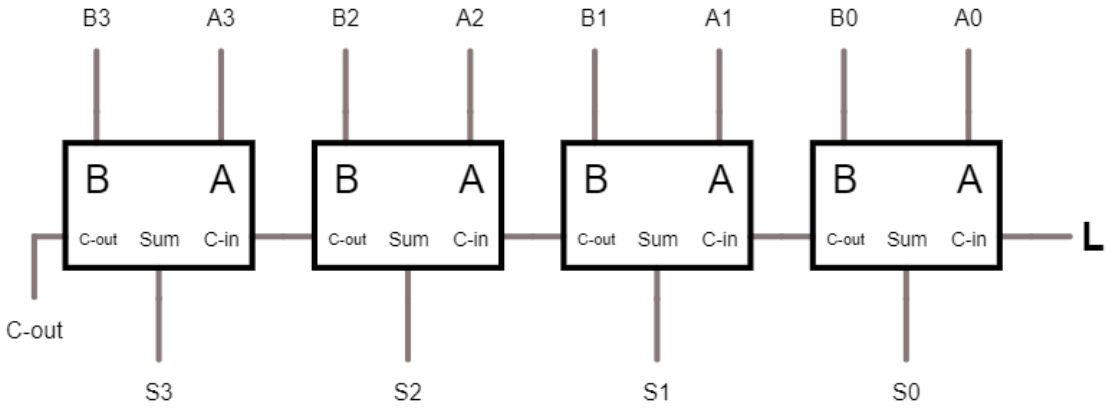


Figura 108: circuito do somador de 4 bits.

Podemos facilmente transformar o circuito em um somador e subtrator adicionando um sinal que deve ser acionado para permitir subtração e ligando-o no único carry in vazio, além de em portas XOR com cada um dos bits do número B. Isso é feito pelo seguinte código Verilog, o qual envolve ignorar o valor do *carry-in*.

O resultado é o seguinte circuito:

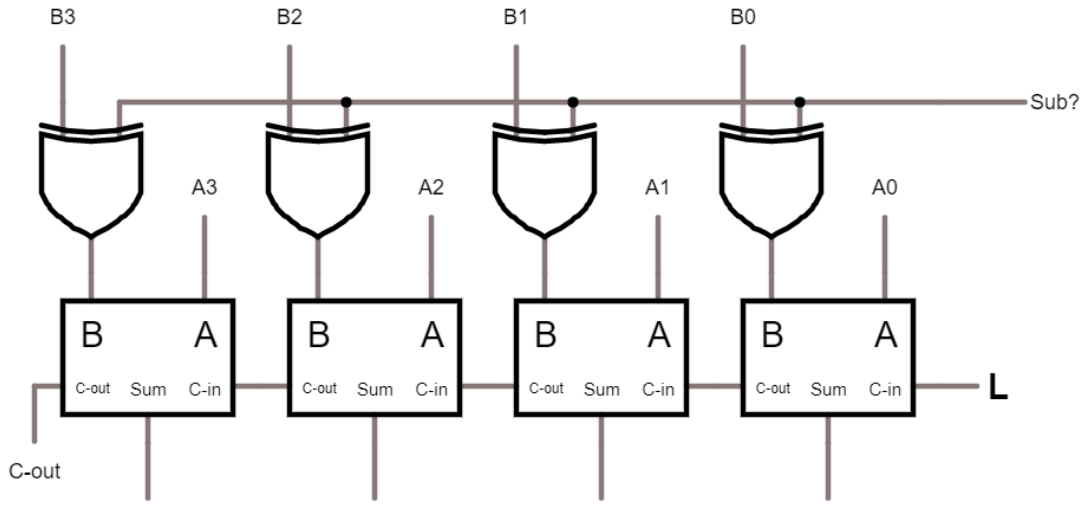


Figura 109: circuito do somador subtrator. O resultado é dado sempre em complemento de 2.

13 LATCHES

Todos os circuitos que trabalhamos até agora são circuitos combinacionais: a saída depende de uma combinação de entradas no presente. Porém, diversas implementações requerem circuitos cuja saída depende não só das entradas no tempo presente, mas também das entradas no passado. Esses circuitos são chamados circuitos sequenciais, e têm como característica fundamental a presença de elementos de memória.

Considere o seguinte problema:

"Um agricultor possui um galinheiro, dentro do qual existem galinhas que, as vezes, escapam. Há um sensor de presença no caminho que dá acesso ao galinheiro, e esse sensor devolve 1 no momento em que alguma galinha passa por ele, e devolve 0 logo em seguida. Projete um sistema que informe o agricultor, ao acordar, se alguma galinha escapou durante a noite"

Nesse caso, precisamos de um circuito capaz de armazenar um sinal 1, algo que os circuitos combinacionais que estudamos até aqui não conseguem fazer.

13.1 LATCH SR

Um circuito denominado latch set-reset (ou latch SR) é um circuito capaz de armazenar um bit de informação enquanto ele se mantiver ligado. Um latch SR pode ser sensível a um sinal alto (1) ou a um sinal baixo (0). Todo latch possui duas entradas, um set e um reset, e duas saídas, uma saída principal (Q) e uma saída negada (Q').

Um latch SR sensível a sinal alto é denominado active high. Em seu estado inicial, ambas as entradas começam com sinal 0. Um sinal 1 no reset faz com que a entrada Q seja levada a 0, e a entrada Q' seja levada a 1. Um sinal 1 no set faz com que a entrada Q

seja levada a 1 e a entrada Q’ seja levada a 0. Por mais que o sinal no set seja levado para 0, as entradas continuarão definidas, até que um sinal de reset seja enviado.

Esse latch pode ser descrito pelo seguinte circuito:

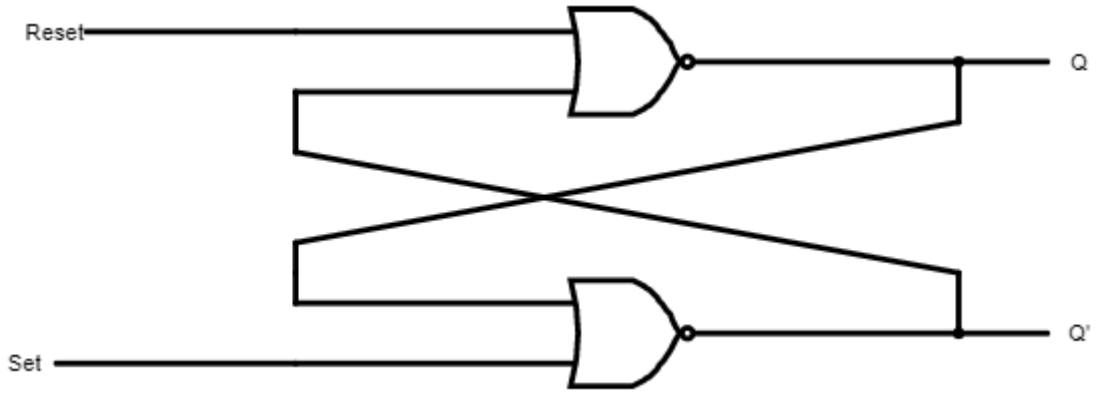


Figura 110: latch SR active high.

Um latch SR sensível a sinal baixo é denominado active low. Em seu estado inicial, ambas as entradas começam com sinal 1. Um sinal 0 no reset faz com que a entrada Q seja levada a 1, e a entrada Q’ seja levada a 0. Um sinal 0 no set faz com que a entrada Q seja levada a 0 e a entrada Q’ seja levada a 1. Por mais que o sinal no set seja levado para 1, as entradas continuarão definidas, até que um sinal de reset seja enviado.

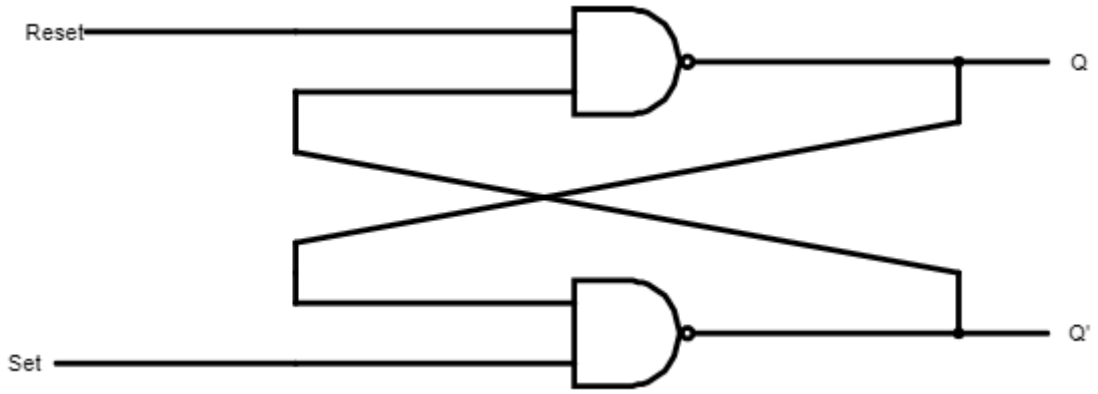


Figura 111: latch SR active low.

13.2 SOLUCIONANDO O PROBLEMA INICIAL

Para solucionarmos o problema do agricultor, tudo que precisamos fazer é utilizar um latch SR, ligando o sensor de presença à função set: se alguma galinha escapar, o circuito exibirá 1 até que seja feito um reset manual.

14 FLIP FLOPS

Neste capítulo, abordaremos mais alguns elementos sequenciais muito úteis na construção de circuitos.

Um flip-flop é um circuito que também é capaz de armazenar 1 bit de memória, mas cujo funcionamento é síncrono em relação a um sinal de clock. Mas o que isso significa?

14.1 SINAL DE CLOCK

Um clock é um sinal que oscila entre dois estados (1 e 0 no nosso caso) em um intervalo de tempo constante. Se fizermos um gráfico relacionando o tempo com o valor lido no clock, teríamos:

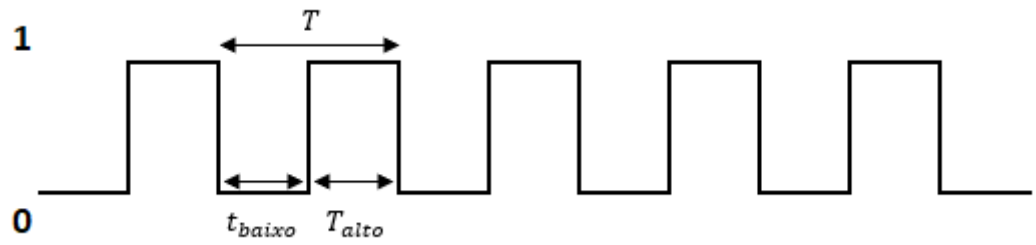


Figura 112: função de onda para um sinal de clock.

Nesse gráfico, o sinal de clock alterna entre 1 e 0, de forma a permanecer um tempo t_{baixo} no nível lógico 0, e em seguida permanecer um t_{alto} no nível lógico 1. Dizemos que T equivale ao período do clock. A relação $f = \frac{1}{T}$ nos dá a frequência do clock em hertz.

Já a relação entre $\frac{t_{alto}}{T} \times 100$ nos dá o clico de trabalho do clock. No nosso caso, em que $t_{baixo} = t_{alto}$, o ciclo de trabalho sempre será 50%.

O momento em que o sinal passa de 0 para 1 é chamado borda positiva, ou borda de subida. Já o momento em que o sinal passa de 1 para 0 é chamado de borda negativa, ou borda de descida.

14.2 FLIP FLOP SR

Suponha, então, que queiramos um latch SR que só aceite valores de set e reset nos momentos em que o clock possui valor 1. Para tanto, basta ligarmos os sinais de set e reset em duas portas AND, juntamente com o sinal de clock, obtendo o seguinte circuito:

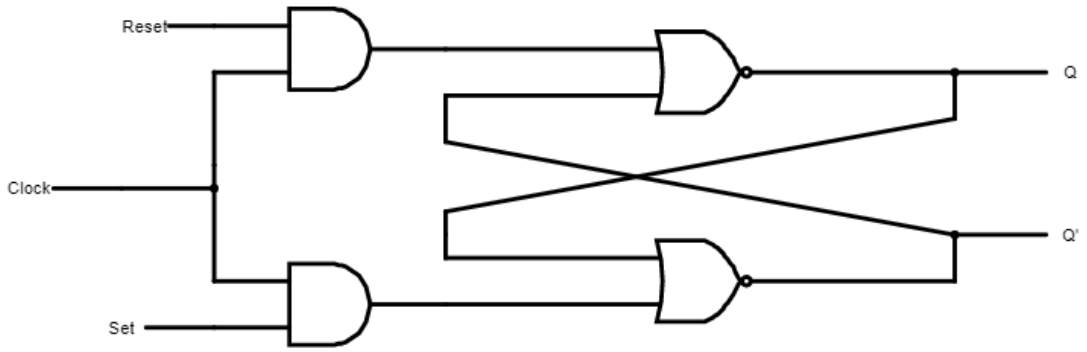


Figura 113: circuito de um flip flop SR

Esse circuito pode ser tratado como uma “caixa preta”, passando a ser chamado, então, de flip flop SR. Veja:

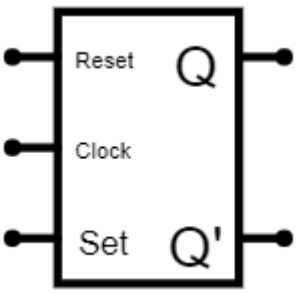


Figura 114: Flip flop SR.

O flip flop SR também pode ser descrito por uma tabela-verdade. Veja:

Tabela 38: tabela-verdade para o flip flop SR.

$Clock$	S	R	Q	Q'	Comentário
1	0	0	1	1	Sem mudança
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	?	?	Indeterminado
0	0	0	0/1	0/1	Sem mudança
0	0	1	0/1	0/1	Sem mudança
0	1	0	0/1	0/1	Sem mudança
0	1	1	0/1	0/1	Sem mudança

14.3 FLIP FLOP D

A partir de agora, manteremos nosso foco em entender os próximos componentes como "caixas pretas", objetivando entender o funcionamento de cada flip flop, mas sem nos atentarmos ao que acontece em seu interior. Essa escolha vem da existência de diversas formas para projeto desses componentes, bem como de suas complexidades.

Um flip flop D é um circuito que possui duas entradas: uma para o sinal de clock, simbolizado a partir de agora por >, e outra para uma entrada D. Ele também possui duas saídas, uma Q, e outra negada, Q'. Veja seu esquemático:

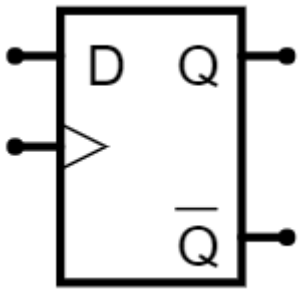


Figura 115: flip flop D.

Sempre que clock subir (portanto, estiver em sua borda de subida), o valor de Q será atualizado para o valor de D naquele momento, e o valor de Q' receberá seu equivalente negado.

Todo flip flop também pode ser ativado pela descida do clock. Para tanto, tudo que precisamos fazer é negar a entrada do clock. Isso pode ser representado por um círculo na entrada do clock, como mostrado abaixo.

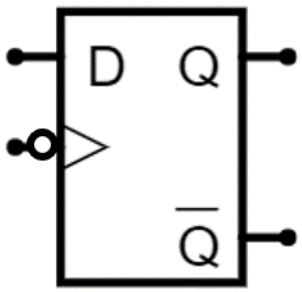


Figura 116: flip flop D com borda negativa.

Adicionalmente, o flip flop D também pode possuir pinos de set e reset que têm a função de levar a entrada Q para 1 e 0, respectivamente. Essas duas entradas funcionam de forma assíncrona, ou seja, independente do clock.

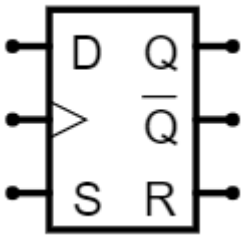


Figura 117: flip flop D com set e reset.

O flip flop D pode ser descrito pela seguinte tabela-verdade:

Tabela 39: tabela-verdade para o flip flop D.

$Clock$	D	Q	Q'	Comentários
0	0	0/1	0/1	Sem mudança
0	1	0/1	0/1	Sem mudança
1 (s/d)	0	0	1	O valor de D é passado para Q
1 (s/d)	1	1	0	O valor de D é passado para Q

14.4 FLIP FLOP T

Um flip flop T é um circuito que possui duas entradas: uma para o sinal de clock, e outra para uma entrada T. Ele também possui duas saídas, uma Q, e outra negada, Q'. Veja seu esquemático, para as versões posedge (sensível a borda de subida) e negedge (sensível a borda de descida), além da versão com set e reset, cujo funcionamento é idêntico aos do flip flop D.

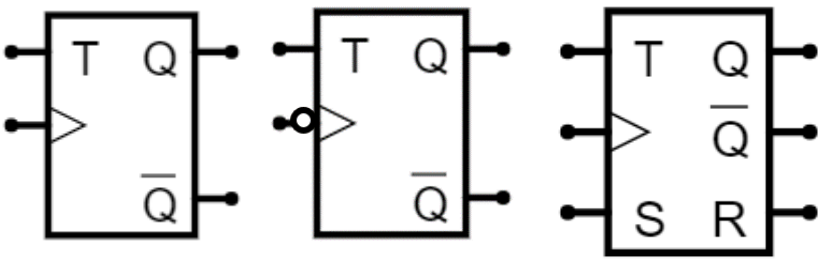


Figura 118: flip flop T

Sempre que o clock subir (ou descer), o valor de Q será invertido caso T possua sinal 1. Veja a tabela-verdade para esse flip flop:

Tabela 40: tabela-verdade para o flip flop T.

$Clock$	T	Q	Q'	Comentários
0	0	0/1	0/1	Sem mudança
0	1	0/1	0/1	Sem mudança
1 (s/d)	0	0/1	0/1	Sem mudança
1 (s/d)	1	Q'	Q	O valor de Q é invertido

14.5 FLIP FLOP JK

O flip flop JK um circuito que possui três entradas: uma para o sinal de clock, uma para o J, e outra para K. Ele também possui duas saídas, uma Q, e outra negada, Q'. Veja seu esquemático, para as versões posedge (sensível a borda de subida) e negedge (sensível a borda de descida), além da versão reset, cujo funcionamento é idêntico aos do flip flop T.

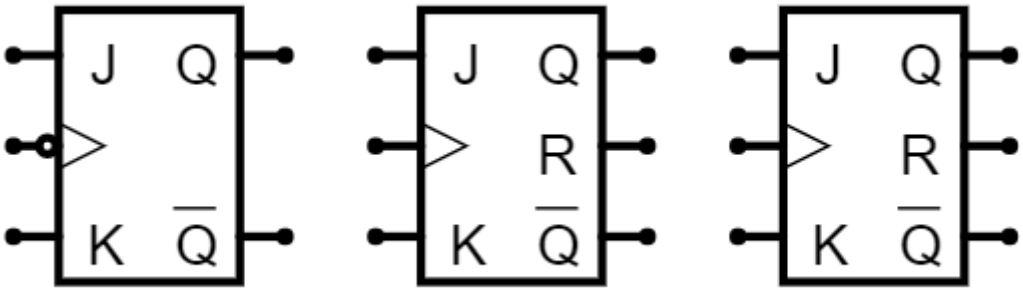


Figura 119: flip flop JK.

Seu funcionamento depende do comportamento das entradas: se J e K forem 0, então o flip flop está inativo independente do comportamento do clock. Se J e K forem 1, então o flip flop funciona como um flip flop T. Já se J for igual a 0 e K for igual a 1, então a próxima subida (ou descida) do clock fará com que seja feito um reset. Se J for igual a 1 e K for igual a 0, então a próxima subida (ou descida) do clock fará um set.

O flip flop JK também pode possuir um reset e set assíncronos.

Tabela 41: tabela-verdade para o flip flop JK.

$Clock$	J	K	Q	Q'	Comentários
0	0	0	0/1	0/1	Sem mudança
0	0	1	0/1	0/1	Sem mudança
0	1	0	0/1	0/1	Sem mudança
0	1	1	0/1	0/1	Sem mudança
1 (s/d)	0	0	0/1	0/1	Sem mudança

1 (s/d)	0	1	0	1	Set
1 (s/d)	1	0	1	0	Reset
1 (s/d)	1	1	Q’	Q	O valor de Q é invertido

15 CIRCUITOS SEQUENCIAIS NOTÁVEIS

15.1 REGISTRADORES SIMPLES

Um registrador é um elemento de memória que armazena uma dada quantidade de bits. Podemos fazer um registrador simples de n bits usando n flip flops do tipo D. Esse circuito funcionará como uma memória, que copia os valores dos n bits de entrada para uma saída sempre que o clock subir ou descer.

O registrador pode ser feito associando-se n flip flops do tipo D com um clock comum. Esse clock pode ser mantido ou negado, e ditará o momento em que o valor recebido nos pinos D de cada flip flop pode ser passado para os pinos Q, que contém a saída.

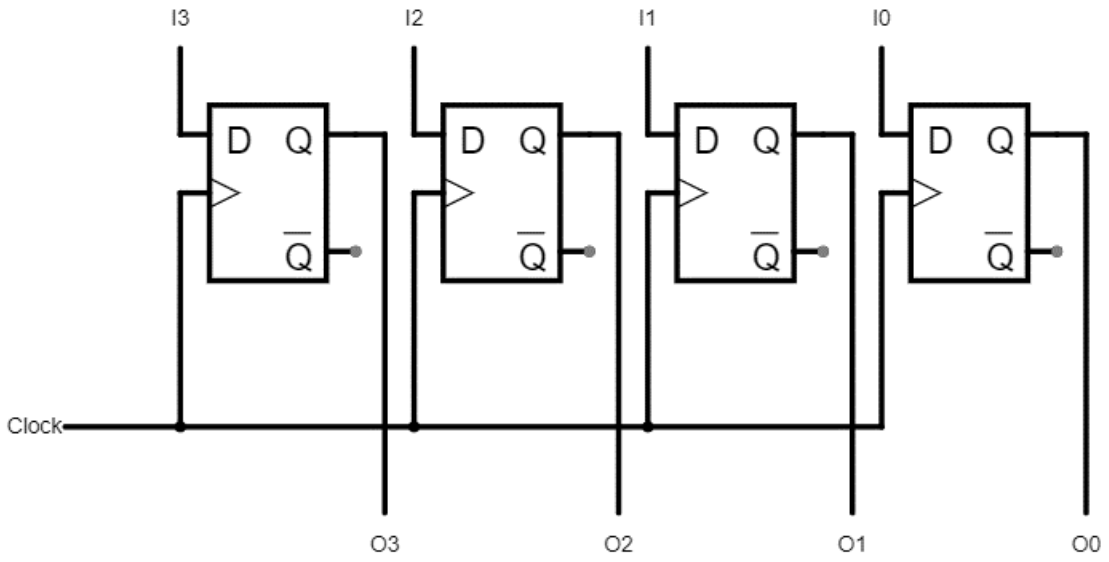


Figura 120: registrador de 4 bits.

15.2 REGISTRADORES DE DESLOCAMENTO

Um registrador de deslocamento é um circuito que armazena os n últimos estados de uma variável. Esse armazenamento é feito na borda de subida ou de descida do clock.

Para fazer esse circuito, devemos associar n flip flops do tipo D de forma a possuírem um clock e um sinal de reset em comum. Então, a saída Q de um flip flop deve ser ligada com a entrada D do outro flip flop. Cada uma das saídas Q equivale a um estado passado. A entrada D mais à esquerda equivale ao estado presente, e deve ser ligado na variável que se deseja monitorar.

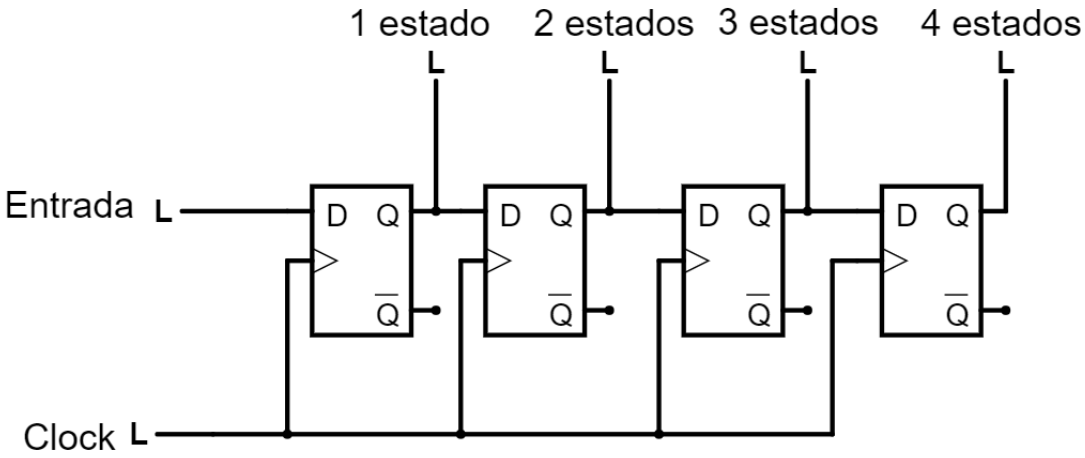


Figura 121: registrador de deslocamento de 4 bits.

16 O QUE VEM A SEGUIR

Percorremos uma longa jornada até aqui. Compreendemos o funcionamento do transistor, elemento fundamental na eletrônica, o que nos levou às portas lógicas, que possuíam modelos matemáticos para compreensão e aplicação. Isso nos permitiu montar diversos circuitos combinacionais, como multiplexadores e calculadoras rudimentares. Em seguida, conhecemos os circuitos sequenciais, aqueles que possuem elementos de memória, e construímos algumas coisas com eles.

É chegado o momento de encerrar este livro. No futuro, o retomaremos abordando sistemas digitais e técnicas gerais para o projeto de sistemas mais complexos com elementos sequenciais e combinacionais.

Espero que este livro tenha sido de leitura prazerosa para você, nos vemos em breve.

O autor.

UM BREVE COMENTÁRIO...

Espero que essa apostila lhe tenha sido útil. Espero, também, que ela tenha chegado as suas mãos da forma que foi feita para chegar: gratuitamente.

Este material faz parte de um projeto que eu, estudante de Engenharia da Computação, mantenho em meu canal no YouTube, visando apresentar a Computação de forma mais simples e voltada para o público leigo.

Se você considera meu trabalho merecedor de qualquer tipo de recompensa, aceito contribuições em criptomoedas, PayPal e Patreon. Veja detalhes em www.fabricadenoobs.com.br/colaborar, ou por meio do QR Code abaixo.

Muito obrigado!