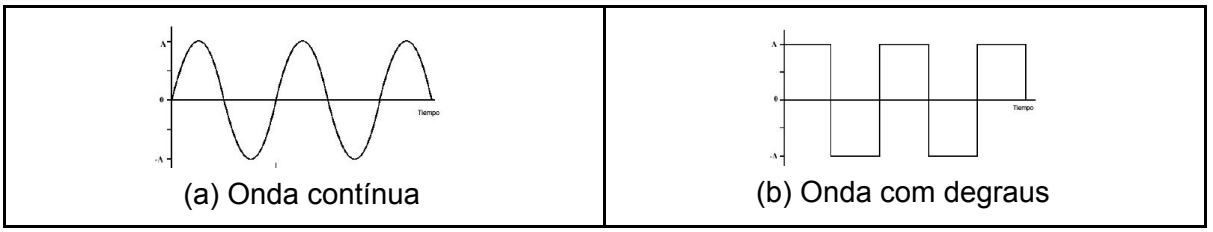
# Sistemas Digitais

Atualmente é possível dividir os diversos sistemas eletrônicos em dois tipos: sistemas digitais e sistemas analógicos. Enquanto o último pode ser descrito como um sistema que é contínuo ao longo do tempo e magnitude, o primeiro é o oposto, uma vez que apresenta valores discretos e degraus entre eles. Os sistemas analógicos apresentam formas de ondas contínua (Figura 1a), enquanto os sistemas digitais, ondas com degraus e certa granularidade (Figura 2a).




(a) Onda contínua (b) Onda com degraus

Figura 1: Exemplo de ondas

O ser humano está envolto a dados analógicos, uma vez que eles fazem parte do cotidiano. Por exemplo, a voz do ser humano é um dado contínuo, pois representa uma medida física no tempo. Inicialmente, todas as tecnologia eram baseadas em representações e dados analógicos, que aos poucos foram sendo substituídos por sistemas digitais.

Os sistemas analógicos apresentam diversos problemas como a dificuldade associada a armazenação dos dados, pois é necessário armazenar o dado na forma de onda. Há outros problemas relacionados ao processo a transmissão dos dados, os quais podem se deteriorar (isso pode ocorrer inclusive durante o processos de leitura/escrita) e os instrumentos analógicos consomem mais energia e são muito suscetíveis a ruídos.

Diferente dos sistemas analógicos (SA), nos sistemas digitais (SD), os dados são armazenados em forma de bit, ou seja, informação binária em que apenas duas representações são possíveis. Eles sofrem menos com ruídos durante sua transmissão ou durante o processo de leitura/escrita, além dos instrumentos digitais não consumirem tanta energia e são menos suscetíveis a ruídos. A única desvantagem dos SDs sobre SAs é o custo envolvendo a construção de ambos, a qual o primeiro é muito mais caro, apesar de que essa diferença vem caindo nos dias atuais. Os dados digitais são bem mais simples de se manipular do que os analógicos, por isso oferece uma gama maior de opções de aplicação.

Os SDs utilizam o sistema binário de numeração e uma vez que todos os dados são utilizados em forma de bit (0 - zero e 1 - um). Há outros sistemas de numeração, como o normalmente usado pelo ser humano, o decimal, e outros como o hexadecimal usado em sistemas computacionais para por exemplo, facilitar a visualização de endereços, além de outros. Em qualquer um deles, é possível realizar a conversão de um para outro. Como os sistemas digitais usam o conceito de bits, torna-se necessário a definição de circuitos adaptados a esse contexto.

A soma binária funciona similar a adição com números decimais. Utiliza-se o mesmo mecanismo de “vai um”, onde ocorre a adição de mais um na próxima dezena e subtração

de dez do resultado da soma atual quando atinge-se ou ultrapassa um valor específico. No caso da soma decimal, esse valor é dez, e na binária, $2_{10}$ ($10_2$). Da mesma forma, a subtração binária utiliza o mecanismo de “pegar um” da subtração decimal. Para realizar a multiplicação basta realizar uma sucessão de somas.

Uma outra forma de realizar a subtração é por meio da soma de um número negativo. A conversão de um número binário para negativo envolve o complemento de 2. O complemento de 2 é a soma de 1 ao complemento de um, que por sua vez, é a inversão de todos os bits de um número original. O uso dessa técnica facilita os circuitos digitais uma vez que se torna necessário a construção somente do somador e do complemento de 2 para a efetivação da soma, subtração e multiplicação.

A eletrônica digital e propriamente os sistemas digitais, se baseiam em alguns circuitos básicos padronizados. Para tais circuitos dá-se o nome de portas lógicas. Esse conceito foi introduzido pelo trabalho *Symbolic Analysis of Relay and Switching*, proposto pelo engenheiro Claude Elwood Shannon. Ele se baseou na álgebra de Boole, a qual foi introduzida por George Boole (1815-1864) em 1854.

Basicamente, uma porta lógica possui pelo menos uma entrada (1...n), e uma saída. As portas lógicas mais básicas são descritas na Tabela 1.

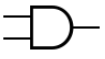
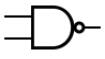
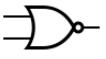
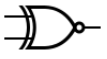
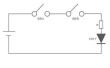
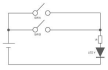
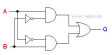
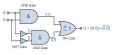
| A | B | AND | OR | NOT(A) | NAND | NOR | XOR | XNOR |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| Símbolo | |  |  |  |  |  |  |  |
| Circuito Físico | |  |  |  | - | - |  |  |
| Notação | | $A.B$ | $A+B$ | $\overline{A}$ | $\overline{A.B}$ | $\overline{A+B}$ | $A \oplus B$ | $A \odot B$ |

Tabela 1: Portas lógicas

A única porta lógica com uma entrada é a NOT. As portas XOR e CON possuem duas entradas. As demais possuem pelo menos duas entradas. As portas lógicas podem ser descritas como:

- AND - Todas as **n** entradas devem estar ativadas (estado 1) para que ative a saída.
- OR - Pelo menos uma das **n** entradas deve estar ativada para que ative a saída.
- NOT - Inverte o valor da sua entrada, se estiver ativada a saída estará desativada (estado 0 - zero), se a entrada estiver desativada a saída estará ativada.
- NAND - É a porta AND invertida, ou seja, é necessário que pelo menos uma porta não esteja ativada para que a saída seja ativada.

- NOR - É a porta OR invertida, ou seja, é necessário que todas as suas **n** entradas entradas estejam desativadas para ativar a saída.
- XOR - Para que sua saída seja ativada é necessário que nenhuma das suas duas entradas assumam o mesmo valor.
- XNOR ou CON - Para que sua saída seja ativada é necessário que todas as suas entradas assumam o mesmo valor.

Qualquer notação em forma de Álgebra de Boole, pode ser transcrita para um circuito lógico, e vice-versa. Ambos podem ser o resultado de uma tabela verdade. A tabela verdade é uma forma de se representar todos os possíveis estados de uma porta lógica ou uma operação lógica. Um exemplo é mostrado na Tabela 1, onde cada porta lógica tem uma saída para todas as possíveis entradas. O número de possíveis combinações das entradas é exponencial, e depende do número de entradas, $2^n$ (2 = quantidade de possíveis valores que a entrada pode assumir; n = número de entradas).

Há algumas formas de fazer a transposição de uma tabela verdade para uma Expressão Booleana. Uma delas é pegar todas as saídas com valor igual 1 e:

1. Unir todas as entradas da mesma linha por uma operação AND, entretanto, todas as entradas com valor zero devem ser negadas.
2. Unir todas as diferentes linhas com saída 1 com operações OR.

Essa operação, apesar de válida, resulta em expressões muito grandes e custosas para serem construídas. Pensando nisso, algumas estratégias de simplificação foram pensadas, sendo elas:

- Postulados:
    a. Adição (OR) : A + 1 = 1; A + 0 = A; A + A = A; A + $\overline{A}$ = 1.
    b. Multiplicação (AND): A . 1 = A; A . 0 = 0; A . A = A; A . $\overline{A}$ = 0.
- Propriedades
    a. Comutação: A + B = B + A; A . B = B . A.
    b. Associativa: A + (B + C) = (A + B) + C = A + B + C; A.(B.C) = (A.B).C = A .B.C
    c. Distributiva: A + (B . C) = AB + AC; A . (B + C) = (A + B) . (A + C)
- Identidades:
    a. De Morgan: $\overline{A.B.C....} = \overline{A} + \overline{B} + \overline{C}....$ ; $\overline{A + B + C....} = \overline{A} \,.\, \overline{B} \,.\, \overline{C}....$ .
    b. Identidade: A + A.B = A; A + $\overline{A}$.B = A + B; (A + B).(A + C) = A + B.C.

Apesar de simplificar a expressão, esse processo é muito custoso e suscetível a erros. Uma abordagem mais simples, rápida e eficiente é o diagrama de Veitch-Karnaugh. A partir de qualquer tabela verdade, é possível construir uma expressão booleana usando o diagrama. Os casos mais comuns são de 2, 3, 4 e 5 variáveis para processo manual, todavia, o computador consegue executar para números de variáveis bem grande.

As portas NAND e NOR ocupam um espaço importante dentro do contexto de eletrônica digital e responsáveis por alavancar a indústria, isso porque qualquer porta lógica pode ser representada por um conjunto de portas NAND ou NOR. Isso faz delas portas universais. Elas são importantes pois podem economizar muito no processo da confecção de um

sistema, visto que é necessário produzir uma única porta em grande escala, o que barateia todo o processo.

Normalmente, as portas lógicas são construídas com transistores com tecnologia CMOS ou TTL. TTL vem de *Transistor-Transistor Logic* e é uma classificação de circuitos integrados. CMOS vem de *Complementary Metal Oxide Semiconductor* e é uma tecnologia diferente. A Figura 1 mostra um exemplo de construção de uma porta NAND com transistores.

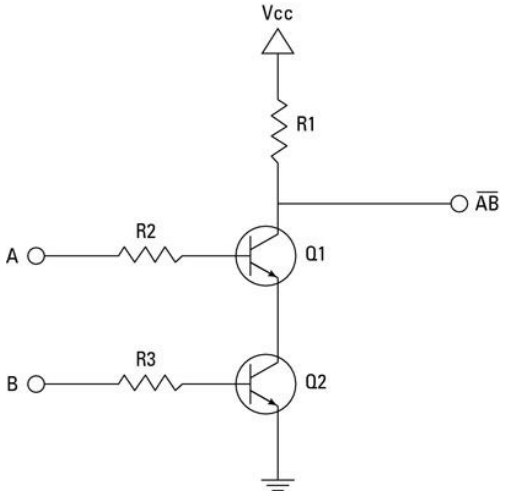



Figura 1: Porta AND implementada com transistors.

Os circuitos digitais podem ser divididos em dois grupos: circuitos combinacionais e circuitos sequenciais. Enquanto nos circuitos combinacionais a saída depende unicamente e exclusivamente das combinações entre as variáveis de entradas, os circuitos sequenciais são aqueles que dependem tanto das variáveis de entrada e/ou do estado anterior, estes que dependem de um pulso denominado clock. O clock é um sinal digital com uma forma de onda periódica, a qual a periodicidade deve ser definida no hardware. O ciclo do clock é dividido em duas partes: 0 ou 1. A transição de subida é a que vai de zero para um e a transição de descida é a que vai de um para zero.

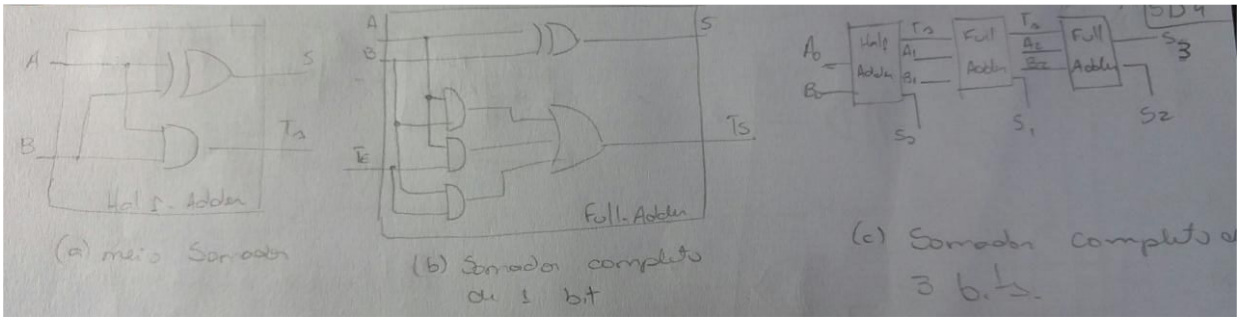
Os circuitos combinacionais são em sua maioria codificadores, decodificadores, circuitos aritméticos, multiplexadores (multiplex), demultiplexadores (demultiplex) e comparadores. Um ponto importante é que não há armazenamento de valores no circuito. Os circuitos combinacionais mais básicos são descritos nos próximos parágrafos.

Os codificadores proporcionam a passagem de um código conhecido para um desconhecido. Por exemplo, a calculadora não conhece os valores 20, 8, 7, 5 ou qualquer outro número decimal, ela conhece números binários, então é necessário passar/codificar o número decimal para o binário. No caso do decodificador, faz-se o inverso, onde um código desconhecido é convertido para um conhecido. Por exemplo, o resultado da soma de dois números em uma calculadora está em binário, logo é necessário decodificá-lo para decimal para o ser humano. O conceito de codificador e decodificador são dependentes do contexto, pois no ponto de vista da calculadora, a passagem de decimal para binário é uma decodificação e de binário para decimal uma codificação. A construção de um decodificador

ou de um codificador pode ser feito por meio da construção da tabela verdade e da simplificação usando o diagrama de Veitch-Karnaugh.

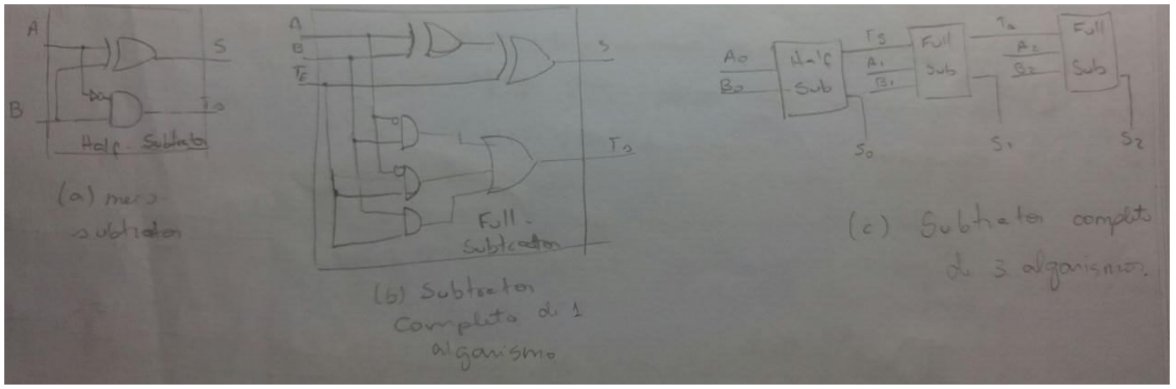
O uso de circuitos aritméticos está relacionado a ULA (Unidade Lógica Aritmética) dos computadores. A ULA envolve operações lógicas (OR, AND, XOR, etc) e operações aritméticas. Os principais sistemas são os somadores completos, que usam os meio-somadores, os subtratores completos, que fazem uso de um meio-subtrator e a fusão de ambos (somador e subtrator completo).

O meio-somador realiza a soma de dois números binários com um algarismo (Circuito 1a). Ele possui duas entradas(primeiro e segundo número) e duas saídas: vai um (carry-out) e o resultado. Já o somador completo realiza a soma de dois números com n algarismos. A diferença do somador completo para o meio-somador é a entrada extra que é o carry in (“recebe um”) e considera o mesmo no cálculo (circuito 1b). Dado o somador completo de 1 algarismo, constrói-se o de n algarismos (Circuito 1c). O meio-somador também é usado no somador completo. É possível fazer o somador completo com somente meio-somadores.



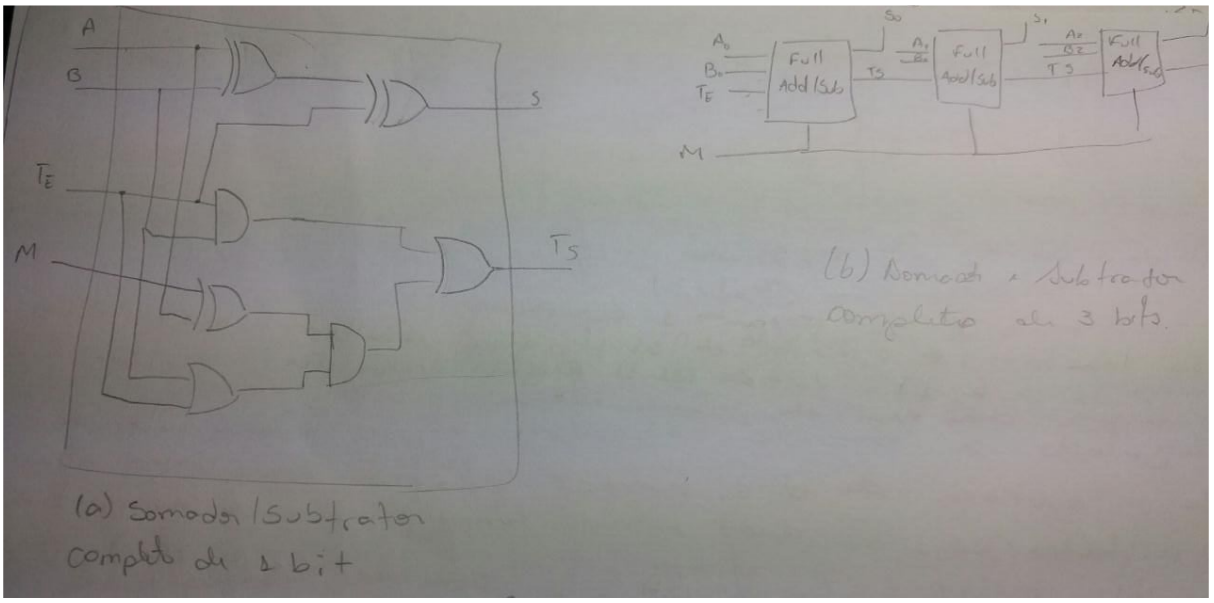

Circuito 1: Somadores

Seguindo a mesma abordagem, o meio-subtrator realiza a subtração de dois números binários com um algarismo. O meio-subtrator possui duas entradas (uma para cada número) e duas saídas, o “pede um” (*carry out*) e a saída (Circuito 2a). O subtrator completo de um algarismo possui uma entrada a mais, a qual possui o *carry in* do anterior (Circuito 2b). O subtrator completo com mais algarismos é uma combinação entre o meio-somador e o somador completo (Circuito 2c).



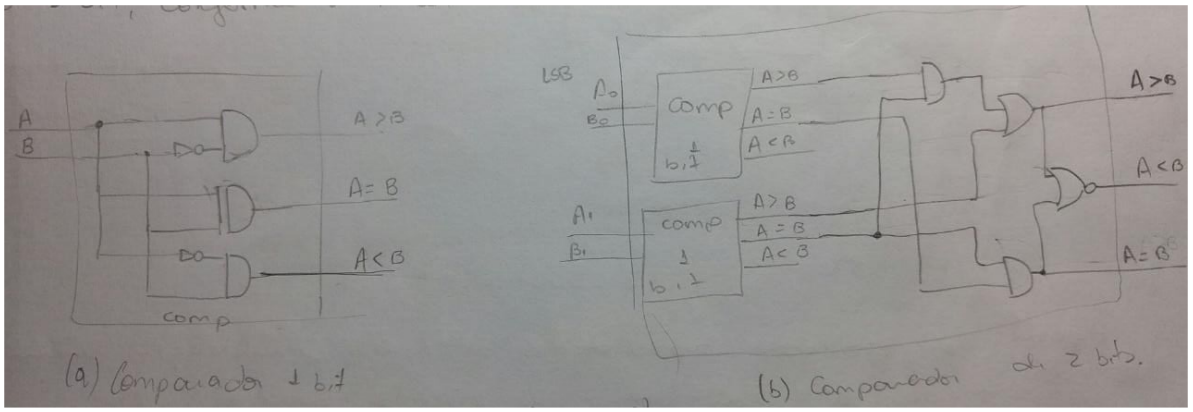

Circuito 2 - Subtratores

Apesar dos sistemas apresentados de soma e subtração serem separados, é possível criar um único sistema que engloba ambos, adicionando somente uma chave M seletora para o tipo de operação (M=0 soma e M=1 subtração). No ponto de vista econômico, isso é extremamente vantajoso , uma vez que o custo reduz, pois agora é feito somente um circuito e o espaço necessário é menor, alŕm de que há algumas sobreposições entre os circuitos. Como é feito para todos os outros circuitos, o processo consiste em criar a tabela verdade e, a partir dela, criar o sistema. O circuito da união de ambos, pode ser visto no circuito 3a. Esse circuito é responsável pela soma/subtração de dois números com somente 1 algarismo. O Circuito 3b contém um exemplo com mais algarismos.




Circuito 3 - Somador e subtrator completos

Um outro tipo de circuito combinacional é o comparador de magnitude. O comparador de 1 bit possui duas entradas (2 números de 1 bit) e três saídas: menor (o primeiro número é menor que o segundo), igual e maior (Circuito 4a). O comparador de mais bits pode ser obtido a partir do comparador de 1 bit, conforme é mostrado no Circuito 4b.

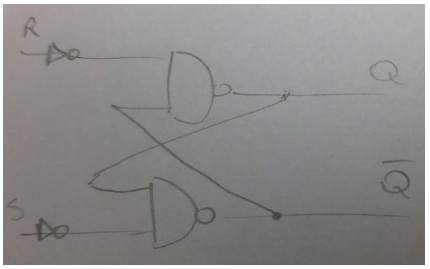



Circuito 4 - Comparadores

O multiplexador, ou Mux, é um circuito com várias entradas, um conjunto de chaves de seleção e uma única saída. O demultiplexador é o inverso, possui uma entrada, e várias saídas, além das chaves seletoras. No caso de mux, a saída é a transmissão de uma das entradas e do demux é o valor de uma das entradas na saída. A entrada ou saída escolhida é de acordo com a combinação das chaves seletoras. O esquema de ambos os circuitos são baseados em portas AND simulando todas as possibilidades de entrada. A quantidade de saídas (mux) ou entradas (demux) deve ser igual ou inferior a $2^n$, onde **n** é o número de chaves seletoras (n). Pode-se usar a união do mux e do demux para a transmissão de dados. Para essa finalidade, basta que tenhamos um bloco no transmissor e um bloco oposto no receptor. Dependendo da ordem dos blocos, pode-se ter uma transmissão paralela (demux - mux) ou uma transmissão série (mux - demux). Para que funcione, as chaves seletoras deve estar sincronizadas.

Os principais componentes de circuitos sequenciais são latches, flip-flops, registradores e memórias. Um ponto importante desse tipo de circuito é que há armazenamento de valores no circuito. Os latches são os elementos de memória mais simples, pois são capazes de armazenar o valor de somente um bit. A saída do circuito é o valor armazenado no circuito, o qual pode ser alterado de acordo com a entrada.

O latch mais simples é o RS. Ele possui duas entradas de saída, e para assumir um dos estados, é necessário uma combinação da entrada. Esse tipo de latch, não possui clock. Os estados possíveis são $Q = 0$ e $\overline{Q} = 1$ e o oposto. Ele recebe o nome RS, pois quando S (de *Set*) assume o valor 1, a saída Q assume valor 1. Quando R (de *Reset*) assume o valor 1, a saída Q assume valor zero. A mudança do valor de saída depende exclusivamente da alteração dos estados R e S. O Circuito 5 apresenta o circuito mais simples



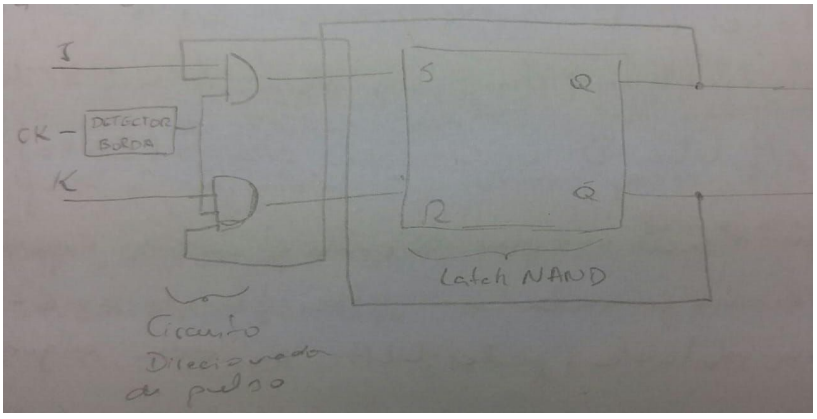

Circuito 5 - Latcher RS

A saída desse circuito pode ser definida de acordo com a seguinte tabela.

| S | R | Q | ~Q | Funcionamento |
|---|---|---|---|---|
| 0 | 0 | $Q_{ant}$ | $\sim Q_{ant}$ | Valor armazenado não muda |
| 0 | 1 | 0 | 1 | *RESET*: valor armazenado passa para zero |
| 1 | 0 | 1 | 0 | *SET*: Valor armazenado passa para um |
| 1 | 1 | - | - | (Imprevisível) Não utilizado |

A principal diferença entre um latcher e um flip-flop está no ponto em que o primeiro tem sua saída afetada pela alteração da entrada, enquanto o segundo somente é ativado quando há uma transição do sinal de controle (borda). O flip-flop necessita de um detector de borda. Ele pode ser feito usando uma porta AND com sua entrada curto-circuitada, contudo invertida, a qual gera um atraso e faz com que o detector funcione. Tanto o Latcher quanto o flip-flop são conhecidos como circuitos lógicos biestáveis. A partir deste ponto, será usado flip-flop com clock e detector de bordas.

Um dos problemas do flip-flop RS é o estado imprevisível quando R=S=1. Como alternativa, criou-se o flip-flop JK. Este não apresenta um estado ambíguo. Ele é construído adicionando-se uma porta NAND a entrada S e outro R, com a saída ~Q como uma entrada da NAND ligada a entrada S, e a Q, da NAND da R, além do clock com detector de borda ligado as duas portas NAND (Circuito 6). Esse contador é bastante usado em contadores binários.




Circuito 6 - Flip-flop JK

Há momentos onde é interessantes setar a saída para 1 e para zero, independente dos valores de J, K e Q. Para isso, criou-se as entradas *Preset* (PR) e *Clear*(CLR). Ambas são ligadas a porta NAND do latch RS, ou seja, elas só afetam o circuito quando assumem valor igual a zero. Elas são entradas assíncronas, isso porque, independente do valor do clock e valores de J e K, quando PR=0, a saída do Flip-Flop passa para 1, e quando CLR=0, a saída assume zero. Quando PR=CLR=1, o funcionamento do flip-flop se comporta normalmente. Apesar de PR e CLR poderem assumir simultaneamente valor igual a 1, o mesmo não se aplica ao valor zero, pois resulta em uma situação não permitida.

O flip-flop JK como foi descrito, ainda possui a característica de que quando o detector de borda é ativado, o circuito funciona como um circuito combinacional, pois há a passagem das entradas J, K e a retro-alimentação. Isso pode ser percebido pois uma configuração diferente de JK gera uma saída diferente enquanto a entrada do clock estiver ativada. O detector de borda ameniza o problema mas não o soluciona. Como uma forma de contornar esse problema, o flip-flop JK mestre-escravo foi proposto. São dois flip-flops JK interligados, onde o clock de um é o inverso do outro.

As entradas PR e CLR ficam no segundo flip-flop JK. Quando o primeiro flip-flop é ativado, há a passagem da entrada JK de Q e ~Q. Contudo, o segundo flip-flop não é ativado e as

saídas Q e ~Q do flip-flop anterior não é passado. Somente quando o primeiro flip-flop é desativado, o segundo é ativado, passando assim as saídas do primeiro JK para o segundo. Enquanto não há passagem/variação do clock, J e K podem variar que a saída não será afetada, somente quando o clock é alterado e o detector de bordas é ativado que os valores de J e K são levados em conta. Quando J e K são iguais a zero, o estado anterior é mantido, J=0 e K=1 o valor de Q se altera para zero, quando J=1 e K=0, então Q=1 e quando J=K=1, então Q=~Q.

O flip-flop tipo D é uma adaptação do JK mestre-escravo. O flip-flop tipo D curto-circuita as entradas J e K, com uma entrada inversa a outra. Ele é muito usado em registradores, uma vez que o valor de D é transferido para a saída Q. A letra D vem de dados (*data*). Eles são usados em registradores.

Os registradores são os responsáveis por armazenar informação na CPU. Essa informação é armazenada na CPU temporariamente para que seja usada e manipulada pela ULA ou por alguma outra instrução, ou até mesmo para ser movida para uma memória fora da CPU. Ela fica no topo das memórias do sistema. Ela é a mais veloz (menor tempo de acesso), menor capacidade de armazenamento e maior custo. São usados como caches na CPU.

A partir dos registradores, é possível construir os registradores de deslocamento, ou *shift-register*. O shift-register é composto por vários flip-flops interligados para armazenar os dados. A cada pulso de clock, o dado de um registrador é passado ao próximo e assume-se o valor do anterior. Ele pode ser construído com flip-flops JK mestre-escravo com as saídas Q e ~Q ligadas as entradas J e K do próximo, respectivamente. Entretanto, o primeiro flip-flop precisa ser do tipo D. Uma outra forma de ser construído é com somente flip-flops tipo D com a saída Q de um ligada a entrada D do próximo. Um dos usos de shift-register é a construção de conversores série-paralelo e paralelo-série.

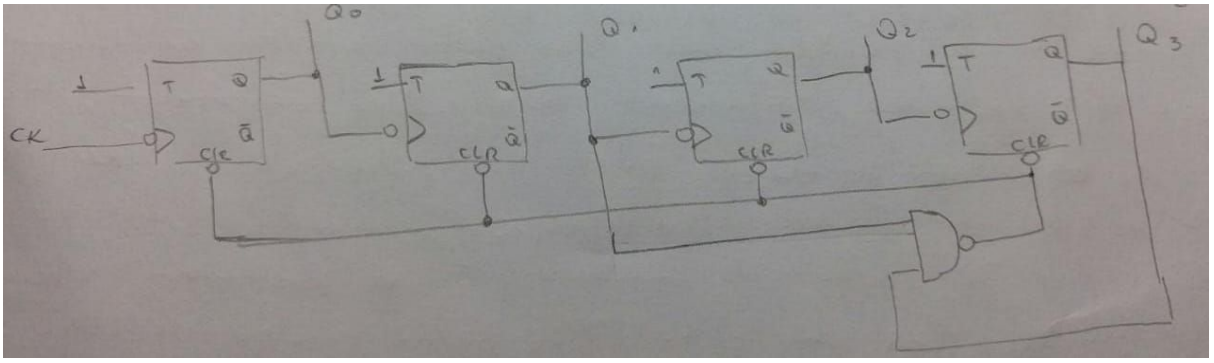
Além dos conversores, é possível realizar multiplicação e divisão por 2 com o shift-register. Quando um bit é deslocada para direita, realiza-se uma divisão por 2 ($1010_2$ ($10_{10}$) -> $0101_2$ ($5_{10}$)). E, com algumas adaptações, pode se realizar a multiplicação fazendo o deslocamento para esquerda ($0011_2$ ($3_{10}$) -> $0110_2$ ($6_{10}$)).

Outro circuito sequencial são os contadores. Estes variam os seus estados sob o comando de um clock, de acordo com uma sequência pré-definida. Eles podem ser usados para contagens diversas, divisão de frequência, modificação de frequência e tempo, geração de forma de onda e para conversão analógico-digital.

Os contadores podem ser divididos em duas partes: contadores assíncronos e contadores síncronos. Nos contadores assíncronos, os flip-flops funcionam de forma assíncrona, pois os pulsos de clock entre os flip-flops não é sincronizada. A entrada de clock só ocorre no primeiro flip-flop, para os demais, a entrada do clock só é feita a partir da saída do anterior. Já nos contadores síncronos, os clocks de todos os flip-flops são ligados ao mesmo pulso de clock e operam juntos e sincronizados.

Partindo do contador assíncrono, é possível criar um contador de pulso (ou binário). Este apresenta em sua saída o sistema binário em sequência. O flip-flop usado neste circuito é uma adaptação do flip-flop JK mestre-escravo. Essa adaptação é feita curto-circuitando as

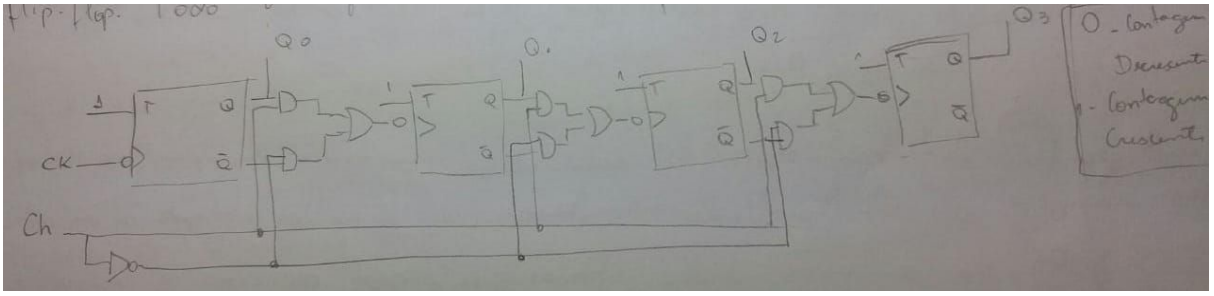
entradas JK. Isso faz com que ambos assumam o mesmo valor. Quando ambos JK assumem o mesmo valor, o flip-flop nunca é forçado assumir um novo valor como é feito no flip-flop tipo D. Quando JK ou T assumem valor zero, a saída Q se mantém, quando T assume um, a saída Q é invertida. Para a construção de contador, primeiro a entrada T de todos os flip-flops são ligadas em estado igual a um, e a saída Q de cada flip-flop é ligada invertida na entrada do pulso de clock do próximo flip-flop. O primeiro flip-flop, o qual é ligado ao gerador de pulso de clock, é o bit menos significativo e o último, o mais significativo. O primeiro é o que mais varia e oferece ciclos mais rápidos, o segundo oferece ciclos duas vezes mais lentos que o primeiro, pois necessita da variação do anterior para mudar seu estado. Os contadores assíncronos, geralmente, são usados para realizar contagens de números zero até $2^{n-1}$, onde n é o número de flip-flops. Entretanto, é possível construir outros contadores que vão de zero até um número qualquer ou de um número qualquer até outro qualquer, desde que estes obedeçam a ordem numérica. Para isso, é necessário criar uma adaptação do contador de pulso normal. Ela é feita por meio da entrada CLEAR e PRESET. Quando a sequência atinge o valor desejado, ativa-se a entrada CLEAR e PRESET. A checagem se caso o valor desejado foi atingido é feito por portas NAND com as suas entradas sendo as saídas dos flip-flops. Liga-se diretamente as saídas Q quando o seu bit corresponde for 1 e inverte-se ou nem se coloca quando o seu bit for zero. O Circuito 7 exemplifica esse processo para um contador de década ($0_{10}$ à $9_{10}$), onde o circuito deve ser zerado quando a contagem atingir $10_{10}$ ou $1010_2$.




Circuito 7 - Contador de década

A partir de algumas alterações, é possível construir um contador decrescente. Isso pode ser feito de duas formas: (i) a saída associada ao contador passa a ser a ~Q do flip-flop e o restante do circuito permanece igual; ou (ii) usa-se as entradas PRESET para setar todos os flip-flops e injeta-se a saída ~Q ao clock do próximo. Neste caso, a saída continua sendo Q.

O ideal dentro de qualquer sistema é sobrepor o uso de componentes e pensando nisso, pode-se criar um contador assíncrono crescente e decrescente em um mesmo circuito. Para que isso ocorra, adiciona-se uma chave para definir qual o tipo de contagem. As saídas Q e ~Q são ligadas cada uma AND com a outra entrada sendo a chave de seleção, a qual é invertida a AND da saída ~Q. Ambas as ANDs são ligadas a uma porta OR, que por sua vez, é ligada ao clock do próximo flip-flop. Todo o esquema do circuito pode ser visto no Circuito 8.

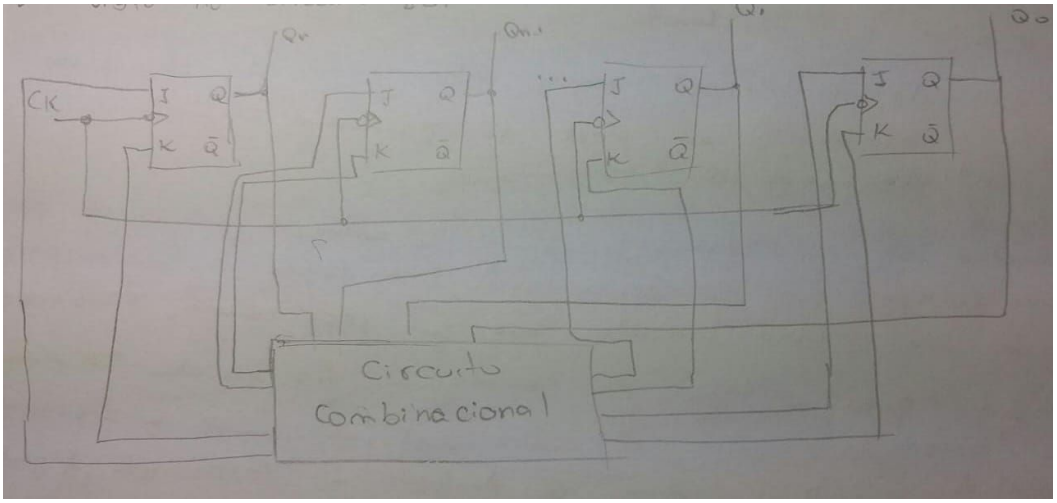



Circuito 8 - Contador crescente e decrescente com chave seletora

Os contadores assíncronos são mais usados para contagens sequenciais, por exemplo de 0 a 10, de 5 a 16 e assim sucessivamente. Para contagens de sequência *aleatórias* (ela é pré-definida, todavia não sequencial) usa-se o contador síncrono. Para esse caso, usa-se o flip-flop JK mestre-escravo sem nenhuma adaptação, entretanto, há a adição de um circuito combinacional que gerencia as entradas J e K de cada flip-flop, uma vez que:

1. Quando Q=0 e deseja-se manter, basta fixarmos J=0, independente do valor de K
2. Quando Q=0 e deseja-se mudar para um, basta fixarmos J=1, independente do K
3. Quando Q=1 e deseja-se manter, basta fixarmos K=0, independente do valor de J
4. Quando Q=1 e deseja-se mudar para zero, basta fixarmos K=1, independente do J

Para a construção de um contador síncrono de uma sequência qualquer, até mesmo 0 a 9, deve-se criar a tabela verdade da sequência desejada para cada entrada J e cada K. O circuito esquemático pode ser visto no Circuito 9. É o circuito combinacional que dita a ordem da sequência baseado nas saídas Qs.




Circuito 9 - Contador síncrono

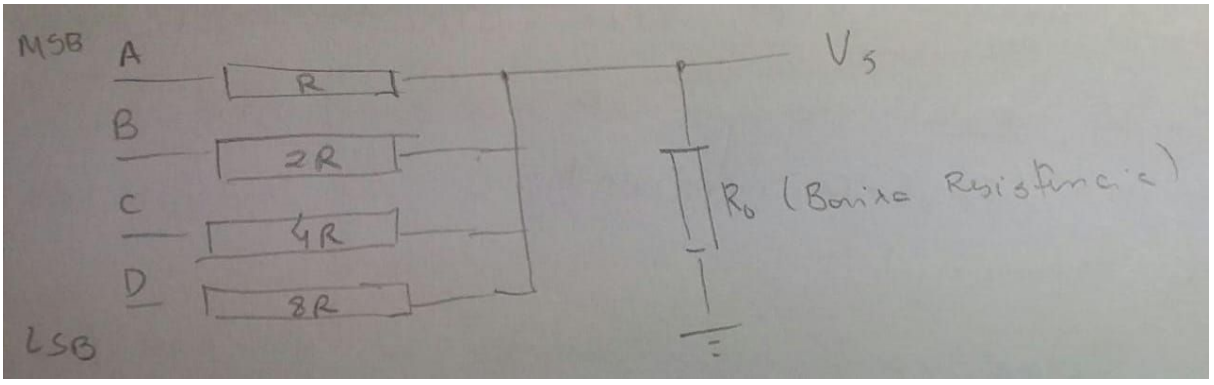
Os circuitos sequenciais tem em sua composição elementos de circuitos combinacionais e elementos de memória. O estado atual influência nos resultados do circuito, conforme dito. Os circuitos combinacionais já foram explorados, mas os elementos de memória ainda não. Esses elementos de memória são dispositivos que armazenam informações. Essas

informações codificadas digitalmente podem representar diversos tipos, como letras, números, caracteres, comando de instruções, endereço, além de outros tipos de dados.

Os elementos de memória, principalmente os latches e Flip-Flops ocupam um espaço importante na indústria, principalmente pois possibilita a implementação de máquinas de estados finitas. Isso possibilitou novas áreas de evoluírem, principalmente as de hardware descrito por Hardware (HDL), as quais utilizam as máquinas de estado finito no processo de síntese (conversão de HDL para o sistema físico).

A estrutura básica de uma memória consiste em uma barramento de endereços, barramento de controle, ambos entrada, e um barramento de dados que pode tanto ser usado para escrita, quanto para leitura. Há memórias não-voláteis (não perdem os dados quando corta-se a energia), como a ROM que já vem escrita de fábrica, a PROM que vem virgem de fábrica e só pode ser escrita uma única vez, e a EPROM que pode ser apagada com luz ultravioleta e EEPROM que eletricamente apagável, ambas podem ser reescritas, e memória voláteis como a RAM, que pode ser de dois tipos SRAM (usualmente usada em caches e não perde os dados com o passar do tempo) e a DRAM (precisa de um refresh, pois perde o dados com tempo devido a sua natureza capacitiva).

Como estamos rodeados de dados analógicos, como sinais de voz ou tensão em um circuito, faz-se necessário a conversão entre sinais anaçógicos e digitais. Os conversores digitais-analógicos, geralmente, convertem um sinal binário para um analógico (nível de tensão, por exemplo). Tem-se na saída a mesma informação que na entrada contudo em nível de tensão. A forma mais simples de se fazer isso é com resistores. O nível lógico 1 de cada bit oferece a mesma tensão, então usa-se diferentes resistores, onde a maior resistência é do bit menos significativo, uma vez que ele é o que menos contribui com o nível de tensão, enquanto o bit mais significativo tem a menor resistência, uma vez que é o que mais contribui com o nível de tensão (Circuito 10),

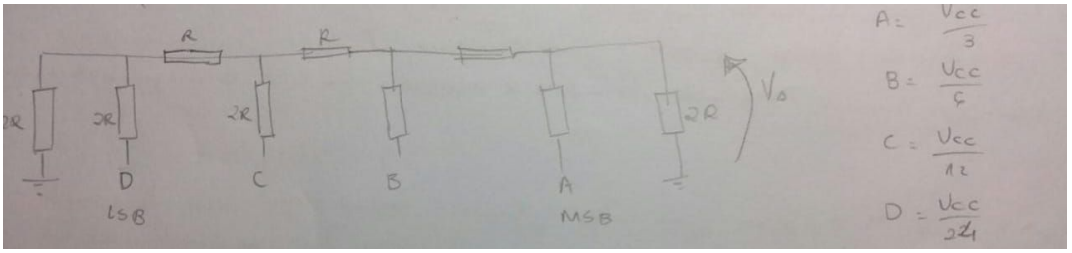



Circuito 10 - Conversos simples D/A

$V_s$ = Vcc. R'/R (soma-se todas e R' resistor no qual terá a tensão de saída).

Outra forma de realizar a conversão é com a rede R-2R. Nesse circuito, resistores de 2R são colocados em paralelo e conectados por resistores R, conforme mostra o circuito 11. Em alguns casos, pode ser interessante colocar uma porta AND ligada a cada bit antes de

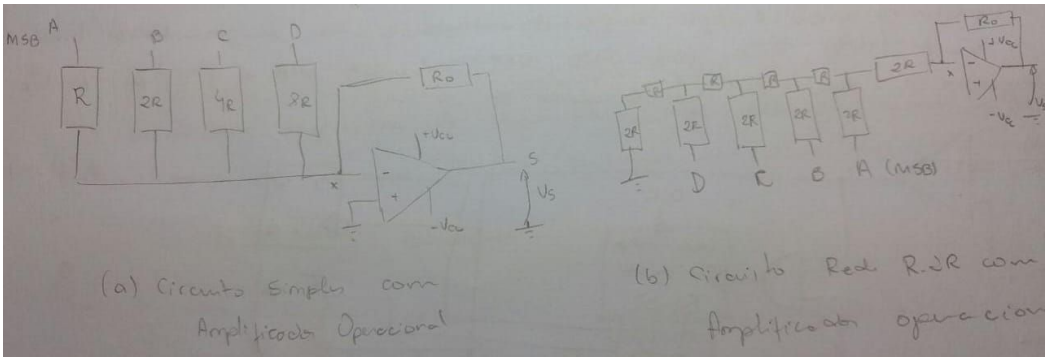
entrar no circuito. Isso favorece ao ponto que isola a impedância de saída do circuito que será ligado ao conversor, fornecendo um sinal de entrada bem definido.




Circuito 11 - Rede R-2R

Nem sempre a saída (tensão) de um circuito digital é suficiente para um circuito analógico. Para esses casos, usa-se o amplificador operacional, o qual apresenta alta impedância (capacidade do circuito de resistir ao fluxo de uma determinada concorrente elétrica quando se aplica uma certa tensão elétrica em seus terminais) de entrada e baixa impedância de saída. Além disso, ele também pode ser usado como comparador de tensão, uma vez que quando a tensão nas duas entradas do amplificador (1 e 2) a saída é zero. Quando a tensão da entrada inversora é superior e não-inversora, a saída será -Vcc, e caso contrário (tensão da entrada inversora menor que a não inversora), a saída será +Vcc. O uso do amplificador proporciona duas principais vantagens: (i) oferecer uma tensão de saída com fator de proporcionalidade qualquer, independente da tensão fixada para nível 1 (alterar ganho); e (ii) melhor acoplamento do conversor com outros circuitos, pois o operacional isola a impedância do conversor com a carga.

Os dois circuitos apresentados podem ser adaptados com o amplificador operacional, conforme mostra o circuito 12.



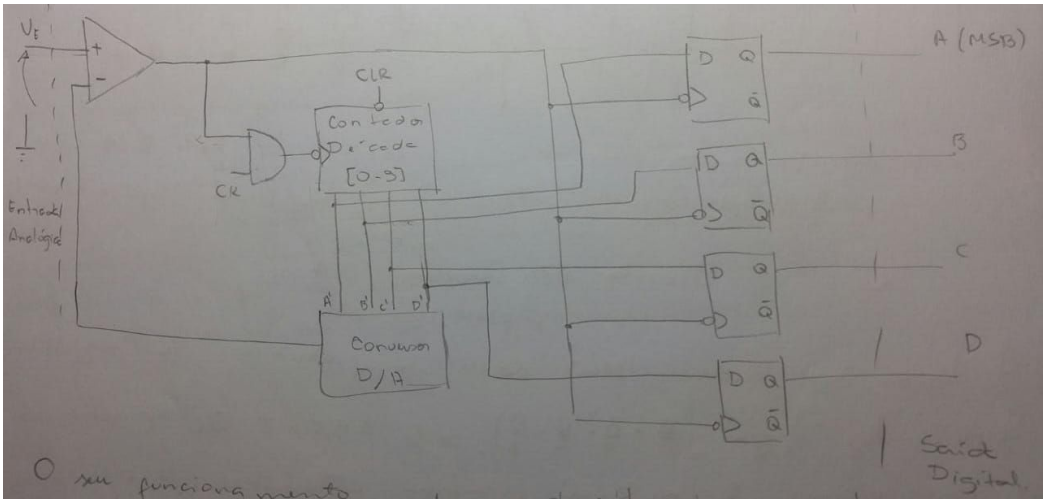

Circuito 12 - Conversores D/A com amplificador operacional

A tensão do circuito 18a pode ser calculado da seguinte forma: $Vs=-(V.R_0/R)*(A+B/2+C/4+D/8)$, onde A, B, C e D são os valores do bit (zero ou um), $R_0$ ou R' é o mesmo já apresentado, V é a tensão de nível 1 e R é a resistência base de R, 2R, etc. O ponto X apresenta baixo potencial, pois o amplificador operacional tem a característica básica um elevado ganho, daí ser conhecido como terra virtual, uma vez que esse baixo potencial será praticamente o mesmo da entrada não-inversora. A tensão de saída é limitada pela tensão de alimentação do amplificador operacional. Pode-se aumentar a granularidade dos conversores, basta ir adicionando grupos de bits com diferentes ordens de grandeza, por

exemplo os 4 primeiros na ordem de $10^0$, os próximos 4 de $10^1$ e assim sucessivamente. Para se calcular a tensão de saída para esse caso: usa-se $Vs = -(R_0/2R)*(V_1 + V_{10}/10 + ....)$, onde calcula-se separadamente a tensão de saída em cada um e soma-se tudo, dado a proporção de cada um.

Para converter qualquer código para analógico, basta primeiro converter para binário (código BCD 8421) e depois realizar a conversão digital-analógico.

O circuito conversor Analógico-Digital (A/D) é um pouco mais complexo e envolve um amplificador operacional, um contador binário de década (0-9), uma porta AND, um conversor D/A e flip-flops tipo D, conforme é mostrado no Circuito 13.




Circuito 13 - Conversor A/D

O seu funcionamento pode ser descrito da seguinte forma:

1. No estado inicial, a saída do amplificador operacional é a tensão positiva, proporcionando sinal 1 para a porta AND, ativando a portas AND em conjunto com o clock.
2. A cada pulso de clock, o contador aumenta, consequentemente, muda a entrada do conversor D/A, que, com o aumento do contador, aumenta a saída de tensão do conversor.
3. Quando a saída de tensão do conversor se equipara a de entrada ($V_E$), a saída do amplificador zera, com isso, o contador de década para a contagem (a saída da porta AND fica zerada). A saída do contador, a qual é proporcional a $V_E$, é aplicada aos flip-flops.
4. Há uma variação na borda do flip-flop, o qual é ativado com as entradas D oriunda do contador. Esses bits são salvos nos flip-flops e têm-se a tensão de entrada em forma binária nas saídas Qs dos flip-flops.

Com somente um contador de década, pode haver um erro muito alto. Para evitar isso, ao invés de usar somente um contador, utiliza-se dois (ou mais) contadores de década ($0_{10}$-$99_{10}$). Quanto maior o intervalo, mais precisa a conversão. Ao se usar somente um

contador, a saída do conversor D/A terá somente 10 degraus de 0 a até a tensão de nível lógico 1. Ao adicionar mais um contador, adiciona-se mais degraus para cada degrau.

As principais aplicações do conversores, é de voltímetro, gerador de forma de onda, gerador de rampa digital, gerador de forma de onda triangular, ou qualquer outra forma de onda.

O processo de projetos de sistemas digitais envolvem dois passos: (i) capturar o comportamento desejado para o circuito lógico; e (ii) converter o comportamento em um circuito. A diferença entre o projeto de um circuito combinacional e o sequencial está na forma de captura do comportamento do circuito. Enquanto o primeiro usa tabelas-verdade ou uma equação, o segundo usa máquina de estados finitos.

Os sistemas digitais são a base de praticamente todos os sistemas atuais. Entretanto, atualmente, os sistemas digitais modernos são projetados com abstrações e abordagens diferentes das que eram usadas, onde usava-se os elementos descritos aqui. Entre as novas tecnologia destacam-se o uso de microcontroladores, dispositivos lógico programáveis (PLDs), e sistema em um chip (SoC). Hoje o projeto de sistemas digitais estão seguindo o caminho dos PLDs, principalmente com FPGAs. Isso possibilitou uma abstração maior no projeto. Ela possibilita que novas abordagens evoluam, como os SoCs. No caso de microcontroladores, diminuí-se a complexidade por parte do hardware e atribui ao software. Por exemplo, o Código 1 apresenta o código em VHDL para o somador completo de um bit.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY fulladd IS
        PORT ( Cin, x, y : IN STD_LOGIC ;
                s, Cout : OUT STD_LOGIC ) ;
END fulladd ;
ARCHITECTURE LogicFunc OF fulladd IS
BEGIN
        s <= x XOR y XOR Cin ;
        Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y) ;
END LogicFunc ;
```

Código 1: Código em VHDL de uma somador completo de 1 bit

Os sistemas digitais hoje são a base do nosso dia-a-dia. Eles estão presentes desde o microondas até os nosso carros e aplicações específicas. Esses sistemas evoluíram com o passar dos anos, o que favoreceu que novas formas de projetos fossem criadas (FPGAs). Essas novas formas aceleraram o processo de criação de softwares e tornaram o processo mais barato. É por esse motivo que hoje temos celulares mais potentes, carros inteligentes e novas tecnologias, como os sistemas ciber-físicos (convergência entre computação, comunicação e controle, sendo a integração da computação com processos físicos) e a internet-das-coisas.

Disponível em: https://github.com/plsilva/Files_DigitalSystems
http://www.ebah.com.br/content/ABAAAAxxgAK/sistemas-digitais?part=6