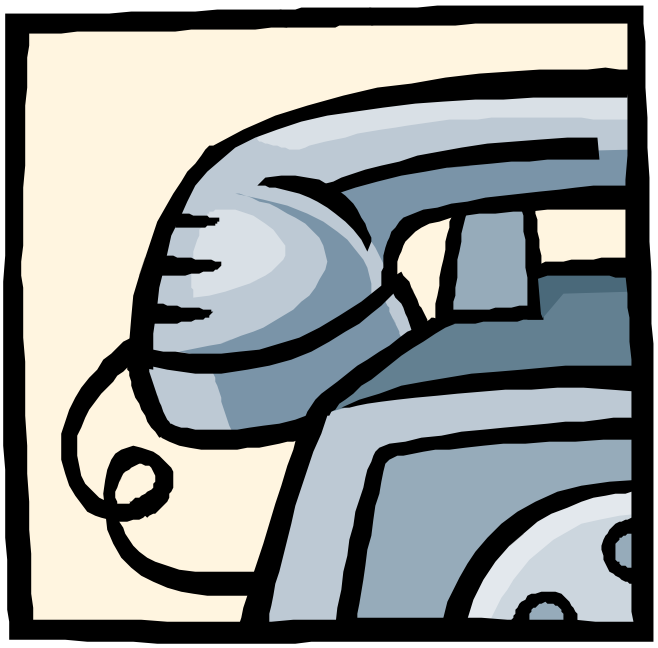
Mestrado em Engenharia Electrotécnica e de Computadores

Projecto de Circuitos e Sistemas Digitais (2000/01)

Gerador digital de tons DTMF *(Dual-Tone Multi-Frequency*) para codificação de dígitos na linha telefónica



Trabalho realizado por:

Luís Filipe Moreira

Ricardo Almeida

Projecto de Circuitos e Sistemas Digitais

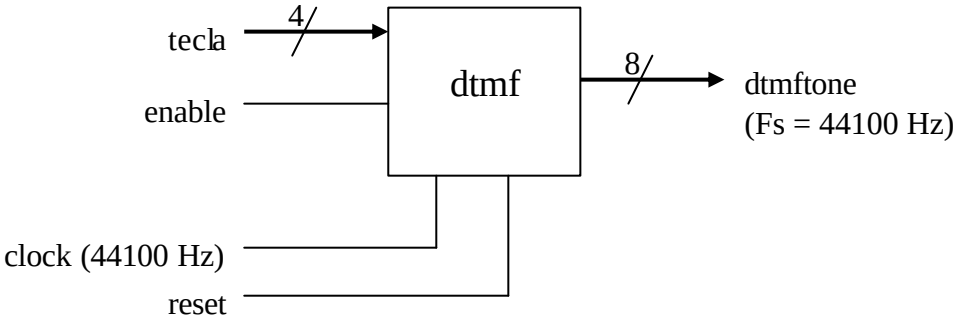
Introdução

O objectivo do trabalho consiste em desenvolver um gerador digital de tons DTMF *(Dual-Tone Multi-Frequency*) para codificação de dígitos na linha telefónica.
Usando um acoplamento acústico adequado, este sistema permite efectuar a marcação de números de telefone recorrendo a telefones que utilizem este sistema de sinalização. O sistema foi realizado num sistema de prototipagem construído com um FPGA XILINX e associado a outros circuitos de interface como teclados ou conversores de sinal.

Descrição do projecto

Interface

A interface pretendido para o sistema é o seguinte:



As entradas e saídas apresentadas na figura têm a funcionalidade seguinte:

tecla: codifica em 4 bits a tecla premida, de acordo com a tabela 1.

tecla [3:2]/tecla [1:0]	00	01	10	11	F1 (Hz)
00	1	2	3	A	697
01	4	5	6	B	770
10	7	8	9	C	852
10	*	0	#	D	941
F2 (Hz)	1029	1336	1477	1633	

Tabela 1. Codificação dos dígitos com a entrada tecla[3:0] e frequências associadas a cada dígito.

enable: activa com 1 a geração do tom DTMF definido pela entrada **tecla**. Quando **enable** é zero, a saída é colocada com zero.

clock: sinal de relógio, activo no flanco ascendente. A frequência indicada para **clock** de 44100 Hz e igual à frequência de amostragem do sinal de saída, facilitará a

construção de ficheiros de áudio (.WAV) com resultados das simulações que podem ser "tocados" num PC com carta de áudio.

reset: sinal de reset assíncrono activo com o nível lógico alto. Enquanto activado, a saída **dtmftone** deverá manter-se com zero.

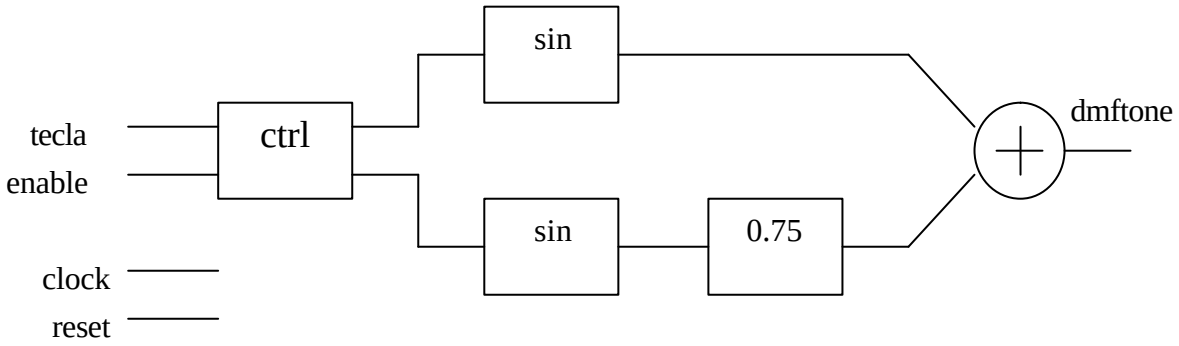
dtmftone: sinal DTMF com 8 bits em complemento para dois, fim de escala e com frequência de amostragem igual à frequência do sinal de relógio (44.1 KHz).

Funcionalidade

O sinal de saída é obtido com a soma de dois sinais sinusoidais, cujas frequências são indicadas na tabela 1. É admitida uma tolerância de ±1.5% nos valores indicados. O sinal com a frequência mais baixa pode ter uma amplitude até -4dB da amplitude do sinal com frequência mais alta (ou 0.631). Para simplificar a implementação pode usar-se o factor 0.75 (-2.5dB) que pode ser facilmente construído usando apenas uma adição e deslocamentos de bits.
O sinal de saída será produzido em complemento para dois com 8 bits e à mesma frequência do sinal de relógio (44.1 KHz). Deverá ser convenientemente preenchida a gama de valores permitidos com 8 bits em complemento para dois.

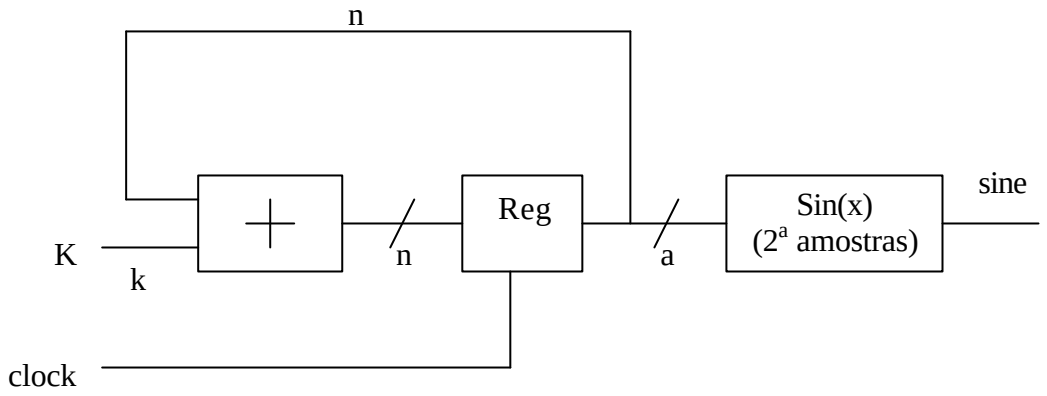
Arquitectura proposta

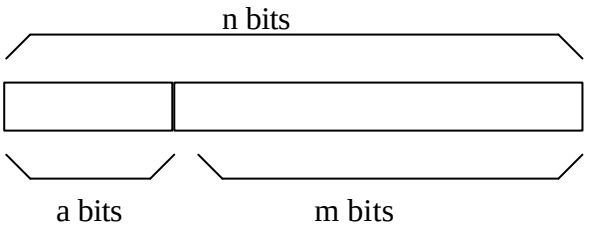
O circuito pretendido pode ser construído usando a seguinte arquitectura:



O sinal DTMF é obtido adicionando dois sinais sinusoidais produzidos por dois geradores de senos operando em paralelo. A unidade **ctrl** converte o código da tecla (tabela 1) nos sinais adequados para os geradores de senos de forma a que produzam sinusoides com as frequências adequadas. O somador final adiciona as amostras dos dois sinais, após multiplicar o sinal de frequência mais baixa por 0.75 (note que 0.75 = 0.5 + 0.25 = 2^-1 + 2^-2).
A geração de um sinal sinusoidal pode ser realizada recorrendo a uma tabela de consulta com amostras de um período completo da função seno e um gerador de endereços para essa tabela. A cadência a que os endereços dessa tabela são percorridos determina a frequência da sinusoide gerada.

Na figura seguinte propõe-se uma arquitectura genérica para um gerador de senos:





A frequência gerada será dada por:

$$fsin = (fclk / 2^m / 2^a) * K$$

onde **fclk** é a frequência do sinal de relógio, **m** e **a** são os números de bits do acumulador (ver figura acima) e **K** uma constante inteira de k bits que determina a frequência produzida.

Fases do projecto

Especificação detalhada do sistema

A primeira especificação a ser feita teria de ser quais os valores de **a** e de **m** que satisfisessem os requisitos atrás enunciados. Para o efeito recorreu-se ao matlab, tendo sido realizado um programa (**ms2.m**, em anexo). O parâmetro de aferição de escolha foi o erro relativo da frequência (que, como referido anteriormente, é ±1.5%); de seguida, foram tentadas várias combinações de **a** e de **m**. Com base nos resultados obtidos, constatou-se que uma tabela do seno com 32 amostras por período (o que equivale a ter a=5) e com amplitudes entre –127 e +127 (equivalente a ter m=7) satisfariam as especificações do problema. Como a frequência do sinal de relógio utilizada é 44100 Hz e como a frequência mais alta pretendida é 1633 Hz (caso mais desfavorável), então, recorrendo à fórmula acima descrita, temos que, para a=5 e m=7 obtem-se um valor de K=151.6727 (como se pode verificar na tabela 2) que deverá ser representado com 12 bits (k=12) - para os diversos valores de K, só se utilizou a parte inteira.

Para as diversas frequências pretendidas obtiverem-se o seguintes valores para K:

Freq	K
697	64.73723
770	71.51746
852	79.13361
941	87.39991
1209	112.2917
1336	124.0874
1477	137.1835
1633	151.6727

Tabela 2. Valores de K para as diferentes frequências.

Entrada do circuito e validação funcional

Foram desenvolvidos os vários módulos em Verilog para cada componente, nomeadamente:

- O módulo **ctrl.v** tem como entradas *tecla* - correspondente à tecla premida - e *enable* - correspondente à entrada enable do circuito e tem como saídas *K1* e *K2*, que são as constantes a utilizar para o incremento no módulo do gerador de seno
- O módulo **relogio.v** gera um sinal de relógio (*clock*)
- O módulo **clock44.v** gera um sinal de relógio - *clkout* - com um período 46 vezes maior ao período do sinal de relógio de entrada - *clkin*; tal foi necessário, pois o sinal de relógio disponível para simulação era de uma frequência 46 vezes superior à frequência pretendida (44100 Hz)
- O módulo **geraseno.v** tem como entradas *K*, *clock*, *reset* e *enable* que correspondem, respectivamente, ao incremento a utilizar, à entrada de relógio, e as teclas de reset e enable do circuito; tem como saída *adress*, que corresponde ao endereço (com 5 posições) a ser usado na tabela de seno (instalada no módulo seno32.v)
- O módulo **seno32.v** é constituído por uma tabela de 32 posições para o seno; tem como entrada *adress* (5 bits) e tem como saída *out* com 8 bits em complemento para 2; esta saída corresponde à frequência determinada pelo endereço gerado no módulo gerseno.v que por sua vez é determinado pela frequência dada pela constante *K*
- O módulo **divisor.v** este módulo serve para multiplicar o sinal de entrada por 0.75; isto é feito deslocando o sinal de entrada para a direita uma e duas vezes e somando estas duas partes. A entrada está representada por *in* e a saída representada por *out*
- O módulo **somador.v** serve para somar os dois sinais resultantes dos dois geradores de seno (representadas por *in1* e por *in2*). A saída deste módulo é o sinal de *dtmf* pretendido
- O módulo **principal.v** serve para correr todos os módulos acima descritos aquando da simulação em Veriwell, iniciando as variáveis *enable*, *reset* e *tecla*; de seguida evoca os outros módulos acima referidos para simular a geração do sinal DTMF pretendido para a tecla seleccionada.

Estes módulos encontram-se todos em anexo.

Após a elaboração destes módulos testou-se a sua validade para todos os valores possíveis de *tecla* (0 a 15); os valores do sinal dtmf obtido foram então guardados num ficheiro *.dat, sendo depois lidos usando o matlab e procurando os valores das frequências obtidas. Os valores obtidos encontram-se na tabela 3.

tecla	F1 (Hz)	F2 (Hz)
0	944	1335
1	689	1025
2	689	1335
3	689	1477
4	765	1025
5	765	1335
6	765	1477
7	850	1025
8	850	1335
9	850	1477
A	689	1626
B	765	1626
C	850	1626
D	944	1626
#	944	1477
*	944	1025

Tabela 3. Valores de F1 e de F2 para as diferentes teclas (valores obtidos após simulação em Veriwell).

Os valores obtidos estão dentro das especificações do problema; assim sendo, demos os módulos desenvolvidos em Verilog como correctos. Além disto foram criados ficheiros posteriormente ouvidos através do matlab com a função **sound**; para tal foi necessário converter os dados obtidos num sinal com amplitudes entre -1 e +1.

Síntese lógica e montagem do circuito completo

Neste passo foram criados os componentes a serem utilizados na simulação do circuito através do Xilinx. Como base para esses componentes utilizaram-se os módulos desenvolvidos em Verilog (excepto os módulos **relogio.v** e **principal.v** que serviram unicamente para simular em Veriwell o circuito desenvolvido em linguagem Verl«ilog) e devidamente validados (os módulos descritos no ponto anterior). Estes componentes foram criados numa libraria com o nome **teste** (nome igual ao nome dado ao projecto).
Estes componentes entretanto criados foram incorporados num circuito cujo circuito esquemático se encontra em anexo. Foi necessário introduzir um gerador de sinal de relógio (componente existente na libraria do Xilinx), bem como uma entrada simuladora de um teclado de quatro teclas (componente existente na libraria Feupix), bem um DAC (também existente na libraria do Xilinx) - o DAC foi alterado de modo a que o sinal resultante (do componente **somador**) pudesse ser convertido de complemento para 2 em um sinal binário "normal"; essa alteração consistiu em inverter o bit mais significativo.

Simulação pós-síntese

Com o circuito desenhado, iniciou-se a simulação do mesmo. Para tal utilizou-se novamente o Xilinx. Os resultados da simulação verificados foram o relógio obtido (a

ver se a frequência era na realidade 44100 Hz), os valores de *K* que entravam nos geradores de seno bem como os sinais de dtmf à entrada do DAC.

Implementação

Este foi o último passo a ser desenvolvido em Xilinx, pois no circuito acima representado foi utilizado um único sinal de relógio correctamente distribuído pelos vários *flip-flops* (tal foi possível introduzindo um *buffer* à saída do sinal de relógio do circuito). Prescindiu-se, assim, da validação temporal (tal como aliás vem referido no guião do trabalho). O resultado deste passo foi a criação de um ficheiro - teste.bit - que foi testado num circuito de interface já existente no laboratório.

Conclusões

Os objectivos do trabalho foram alcançados.
Este trabalho permitiu-nos a familiarização com duas ferramentas de concepção de circuitos digitais para nós desconhecidas: o Verilog e o Xilinx.

Bibliografia

A bibliografia utilizada restringiu-se aos manuais de Verilog e Xilinx fornecidos na página do docente.

Anexo

Módulos em Verilog

module CTRL (tecla, enable, K1, K2);

```
input [3:0] tecla;
input enable;
output [11:0] K1, K2;
reg    [11:0] K1, K2;

        always @(tecla or enable)
        begin
        if (enable)
        begin
                case ( tecla )
                        0://0000
                        begin
                                K1 = 64; K2 = 112;
                        end
                        1://0001
                        begin
                                K1 = 64; K2 = 124;
                        end
                        2://0010
                        begin
                                K1 = 64; K2 = 137;
                        end
                        3://0011
                        begin
                                K1 = 64; K2 = 151;
                        end
                        4://0100
                        begin
                                K1 = 71; K2 = 112;
                        end
                        5://0101
                        begin
                                K1 = 71; K2 = 124;
                        end
                        6://0110
                        begin
                                K1 = 71; K2 = 137;
                        end
                        7://0111
                        begin
                                K1 = 71; K2 = 151;
                        end
                        8://1000
                        begin
                                K1 = 79; K2 = 112;
                        end
                        9://1001
                        begin
                                K1 = 79; K2 = 124;
                        end
```

```
                        10://1010
                        begin
                                K1 = 79; K2 = 137;
                        end
                        11://1011
                        begin
                                K1 = 79; K2 = 151;
                        end
                        12://1100
                        begin
                                K1 = 87; K2 = 112;
                        end
                        13://1101
                        begin
                                K1 = 87; K2 = 124;
                        end
                        14://1110
                        begin
                                K1 = 87; K2 = 137;
                        end
                        15://1111
                        begin
                                K1 = 87; K2 = 151;
                        end
                endcase
        end
        else
        begin
          K1 = 0; K2 = 0;
        end
        end
endmodule
```

---

module RELOGIO(clock);

```
output clock;
reg clock;
initial
        #5 clock = 1;
always
        #50 clock = ~clock;
endmodule
```

---

module CLOCK44 (reset, clk, clkout);

```
input  clk, reset;
output clkout;
reg    clkout;
integer valor;


always @ (posedge clk or posedge reset)
begin
        if (reset)
        begin
```

```
                valor = 0;
                clkout = 1;
        end
        else
        begin
                valor = valor + 1;
                if (valor >= 46)
                begin
                        valor = 0;
                        clkout = ~clkout;
                end
                else
                if (valor == 23)
                begin
                        clkout = ~clkout;
                end
        end
end
endmodule
```

```
module GERASENO (K,clock,reset,enable,adress);

        parameter m=7;
        parameter k=12;

        input [k-1:0] K;
        input clock, reset, enable;
        output [4:0] adress;

        reg [4:0] adress;
        reg [k-1:0] n;

        always @ (posedge clock or posedge reset)
        begin
                if (reset)
                        begin
                                n=0;
                                adress=0;
                        end
                else
                        begin
                        if (enable)
                                begin
                                        n=n+K;
                                        adress = n[11:7];
                                end
                                else
                                begin
                                        n=0;
                                        adress=0;
                                end
                        end
        end
endmodule
```

```
module SENO32 (in,out);
        input [4:0] in;
        output [7:0] out;
        reg [7:0] out;

        always @ (in) begin
                case (in)
                        0: out = 8'd0;
                        1: out = 8'd25;
                        2: out = 8'd49;
                        3: out = 8'd71;
                        4: out = 8'd90;
                        5: out = 8'd106;
                        6: out = 8'd117;
                        7: out = 8'd125;
                        8: out = 8'd127;
                        9: out = 8'd125;
                        10: out = 8'd117;
                        11: out = 8'd106;
                        12: out = 8'd90;
                        13: out = 8'd71;
                        14: out = 8'd49;
                        15: out = 8'd25;
                        16: out = 8'd0;
                        17: out = 8'd-25;
                        18: out = 8'd-49;
                        19: out = 8'd-71;
                        20: out = 8'd-90;
                        21: out = 8'd-106;
                        22: out = 8'd-117;
                        23: out = 8'd-125;
                        24: out = 8'd-127;
                        25: out = 8'd-125;
                        26: out = 8'd-117;
                        27: out = 8'd-106;
                        28: out = 8'd-90;
                        29: out = 8'd-71;
                        30: out = 8'd-49;
                        31: out = 8'd-25;
                endcase
        end
endmodule
```

```
module DIVISOR (in, out);
input [7:0] in;
output [7:0] out;

        assign out = {in[7],in[7:1]} + {{2{in[7]}},in[7:2]};   // como se está a multiplicar por 0.75 não há
                                                               //risco de overflow

endmodule
```

```
module SOMADOR (dtmf, in1, in2);
input [7:0] in1, in2;
output [7:0] dtmf;
reg [8:0] temp;

always @ (in1 or in2)

 temp = ({in1[7],in1} + {in2[7],in2});

assign dtmf = temp[8:1];

endmodule
```

```
module PRINCIPAL;


        reg [3:0] tecla;
        wire [7:0] amp_freq_L;
        wire [7:0] dtmftone;
        wire [4:0] adress1, adress2;
        wire [7:0] point_f1,point_f2;
        wire [11:0] k1,k2;
        wire clk,clk1;
        reg reset,enable;


        integer fp;

        initial
        begin: stop_at

                fp = $fopen("dados.dat");

                #1500000;

                $fclose(fp);
                $stop;

        end

        initial
                begin: init
                reset=1;
                #1
                reset=0;
                enable=1;
                tecla = 1;
                #10000
                tecla=0;

        end



        always @(negedge clk)
```

```
	begin
		$fwrite(fp, "%d\n", {~dtmftone[8],dtmftone[7:1]});
	end


	CTRL ctrl1(tecla,enable,k1,k2);
	RELOGIO rel1(clk1);

		CLOCK44 rel2(clk1,clk);

	GERASENO seno1(k1,clk,reset,enable,adress1);
	GERASENO seno2(k2,clk,reset,enable,adress2);

	SENO32 sinal1(adress1,point_f1);
	SENO32 sinal2(adress2,point_f2);

	DIVISOR div1(point_f1,amp_freq_L);

	SOMADOR soma(dtmftone,point_f2,amp_freq_L);
endmodule
```

Ficheiro matlab

```
% Usage: ms( nba, step, no_samples)
% nbstep = numero de bits / step (m)
% no_samples = numero de amostras por periodo
```

function ms2(nbstep, no_samples)

```
freqs = [697, 770, 852, 941, 1209, 1336, 1477, 1633];

for j=1:8
  clear freq freqt step;

  freq = freqs(j);

  step = freq / ( 44100 / no_samples );

  fprintf('Step=%15.10f,  freq = %15.10f\n', step, freq );

  step = fix( step * 2^nbstep) / 2^nbstep;

  freqt = 44100 / no_samples * (step);

        stept = fix( step * 2^nbstep);

        fprintf('Step truncado=%15.10f,  freq truncada = %15.5f,  erro=%6.4f\n', step, freqt, abs((freqt-
freq)/freq)*100 );

  fprintf('Step truncado=%d\n\n',stept);
end
```

Esquemático do circuito