

**UNIVERSIDADE FEDERAL DE ITAJUBÁ**

**PBLE04 – PROJETO DE MODULADOR CONFIGURÁVEL EM  
FPGA**

**PARTE 1 – MODULAÇÃO PCM**

Lincoln Wallace Veloso Almeida – 2018018715

Bruno de Mello Duarte – 2016010988

Ítalo Barbosa Barros – 2018008924

Gabriel Medeiros Cardoso – 2018014574

Itajubá, 13 de setembro de 2021

A primeira etapa do projeto de rádio definido por software (SDR), consistirá na modulação PCM (Pulse-Code Modulation), onde o sinal é modulado através de três etapas: Amostragem, Quantização e codificação.

O processo de amostragem, trata-se da etapa em que o sinal analógico presente em um tempo contínuo será amostrado em tempos discretos a cada intervalo fixo de tempo, intervalo esse o qual denomina-se tempo de amostragem  $t_s$ , que é calculado tomando em conta a frequência de amostragem escolhida para aquele sinal. Para que o sinal possa ser amostrado de forma a ser possível de ser recuperado posteriormente sem que suas características se percam é necessário que a frequência de amostragem escolhida obedeça ao critério de Nyquist, que diz que para que essas características sejam preservadas de forma satisfatória no momento da amostragem é necessário que a frequência de amostragem seja ao menos 2x maior que a componente de maior frequência presente no sinal.

Logo temos:

$$f_s \gg 2 * f_m$$

Sabe-se que o sinal é dado por:

$$m(t) = 3 \cos(20\pi t) + 2 \cos\left(40\pi t + \frac{\pi}{6}\right)$$

Logo, conclui-se que  $f_m = 20[\text{Hz}]$

Com a finalidade de se preservar as características do sinal modulante  $m(t)$  decidiu-se realizar a amostragem a uma frequência 40x maior que  $f_m$  logo:

$$f_s = 40 * f_m = 800[\text{Hz}]$$

O que nos dá um período de amostragem de:

$$t_s = \frac{1}{f_s} = \frac{1}{800}$$

$$t_s = 0,00125 [\text{s}]$$

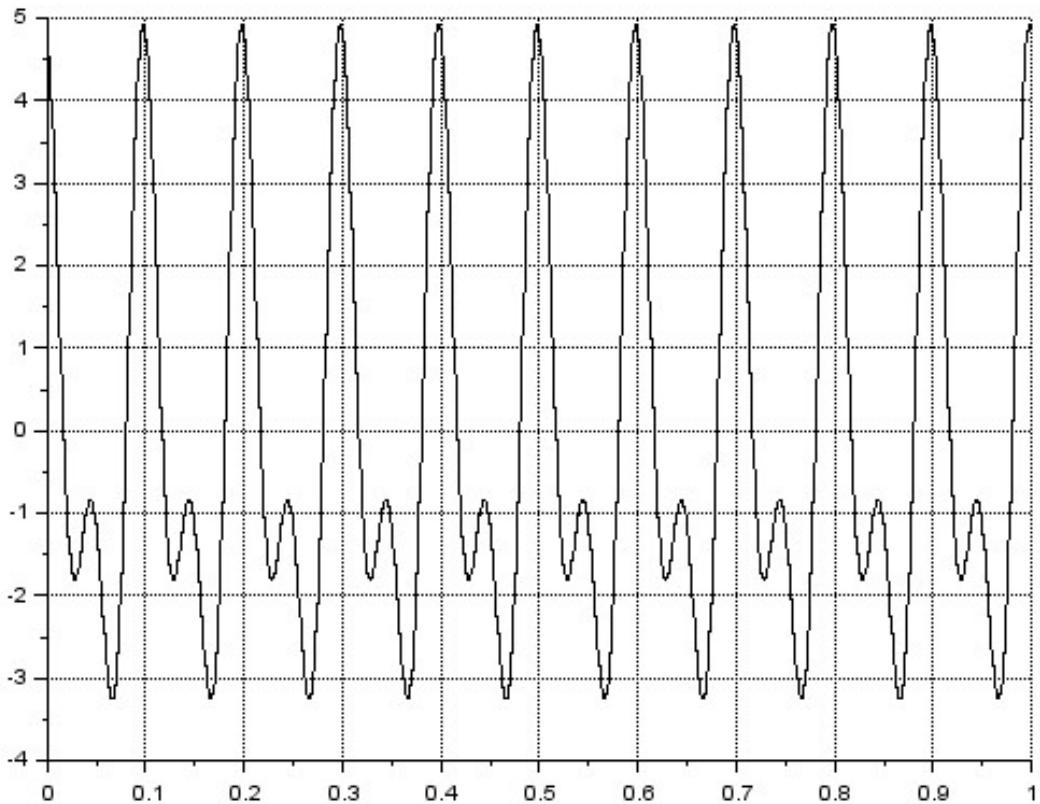
Quanto ao tempo total de amostragem, será utilizado 1[s].

Utilizando o Scinotes (script do scilab) temos:

```

fm = 20;
fs = 40*fm;
ts = 1/fs;
t = 0:ts:(1-ts);
m = 3*cos(20*%pi*t) + 2*cos(40*%pi*t + (%pi/6));
plot2d2(t,m);

```



No processo de quantização, define-se primeiramente a resolução digital, ou seja, o número de bits **n** que se deseja para o projeto e então tem-se o número de níveis de quantização para o sinal modulante. De acordo com as especificações do projeto serão utilizados 8 bits por amostra portanto:

$$L = 2^n = 2^8 = 256$$

O que resulta em intervalos de tensão a serem amostrados com base no valor de máximo e mínimo da tensão do sinal, ou seja:

$$\Delta V = \frac{V_{Max} - V_{min}}{L - 1}$$

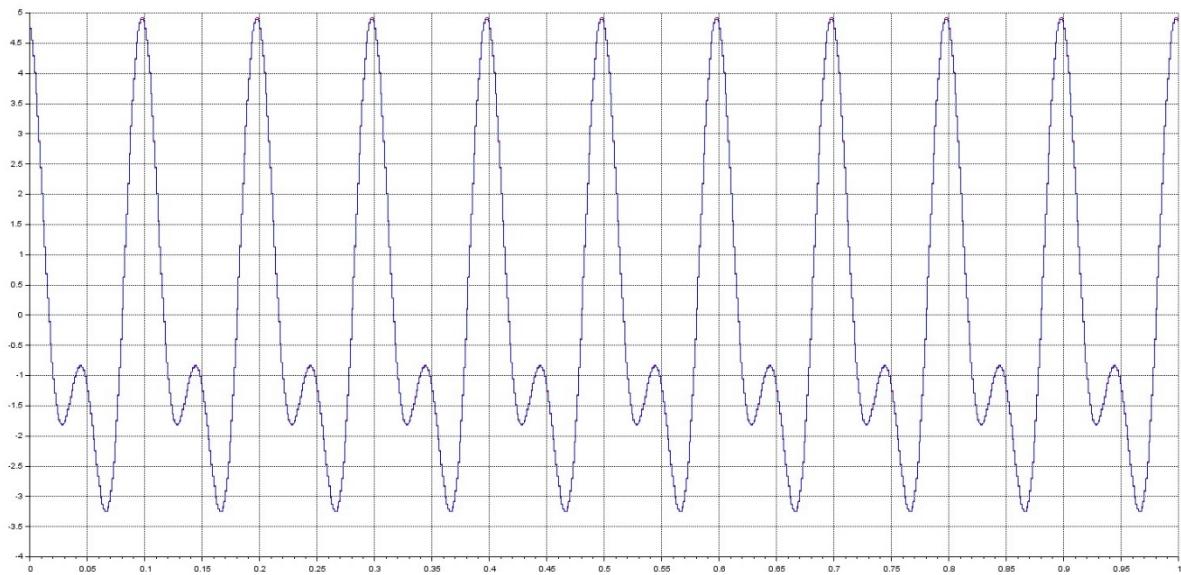
Logo, temos a implementação deste cálculo utilizando o Scilab:

```

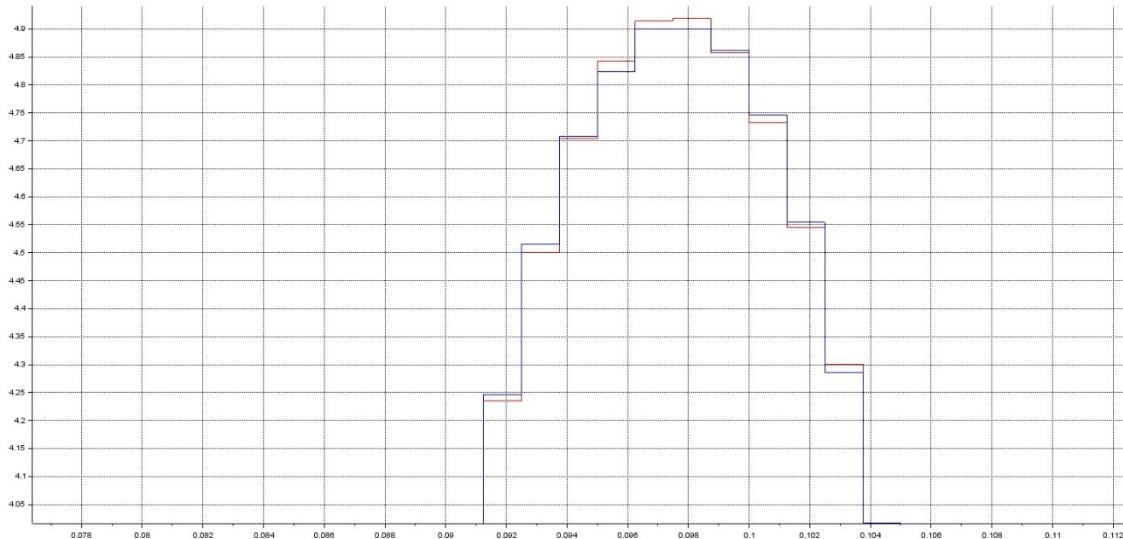
n = 8;
L = 2^n;
mmax = max(abs(m));
mq = m/mmax;
cod = mq;
d = L/L;
q = d*[0:L-1];
q = q -((L-1)/2)*d;
for i = 1:L
    mq(find(((q(i)-d/2)<=mq)&(mq<=(q(i)+d/2))))=
    q(i).*ones(1,length(find(((q(i)-d/2)<=mq)&(mq<=(q(i)+d/2)))));
    cod(find(mq==q(i)))=(i-1).*ones(1,length(find(mq==q(i)))));
end
mq = mq*mmax;
plot2d2(t,mq);

```

Obtendo-se, portanto, o sinal:



Realizando-se um zoom para verificar e comparar os sinais amostrados e quantizados temos:



Na última etapa, tem-se então a codificação dos  $n$  bits onde  $n$  é o número de bits por amostra e  $L$  o número de níveis quantizados no qual:

$$n = \log_2 L$$

Para isso, utiliza-se da implementação da rotina do Scilab, presente abaixo, onde se realiza a varredura do sinal e a cada amostra é atribuído um valor de 0x00 à 0xFF de acordo com a amplitude da amostra:

```
c = zeros(length(m),n);
for i = 1:length(m)
    for j = n:-1:0
        if(fix(cod(i)/(2^j))==1)
            c(i,(n-j)) = 1;
            cod(i) = cod(i)-2^j;
        end
    end
end
```

Logo, o valor codificado do sinal estará salvo na variável **c** que será utilizada posteriormente no projeto. Abaixo encontra-se um link onde é possível visualizar essa variável.

[variável c](#)

Obtivemos então o vetor codificado em 8 bits, a partir da aplicação do PCM no sinal modulante. Para dar prosseguimento ao objetivo de recuperar o sinal analógico original será necessário a representação das amostras obtidas com PWM através da aproximação da amostragem ideal.

Através da comparação do valor absoluto de cada amostra com uma onda dente de serra, a geração do PWM é feita. O funcionamento da saída se dá da seguinte forma: enquanto a onda dente de serra for menor que o valor da amostra a saída terá nível lógico alto e quando não for menor a saída terá nível lógico baixo, produzindo assim uma modulação em largura de pulso.

A geração do PWM na FPGA será realizada comparando a amostra e o valor em um contador incrementado por um clock, levando em conta que o contador deverá resetar para zero quando ultrapassar seu valor máximo, significando que se deve atualizar a amostra utilizando a próxima em seguida.

Para definir o valor da frequência do clock que servirá para incrementar o contador se usará da seguinte relação desta variável com a frequência de amostragem do sinal, que por sua vez é igual a 800 Hz.

$$fs = \frac{f_{clk}}{cmax + 1}$$

Onde  $cmax$  é o valor máximo do contador, cujo valor deve respeitar a seguinte inequação:  $cmax > 2^{N+1}$ , em que N é o número de bits de quantização utilizado. Esta inequação existe para garantir que o sinal PWM gerado não possua nunca um duty cycle superior a 50% da frequência de amostragem. Como neste trabalho utilizou-se 8 bits de quantização, tem-se que:  $cmax > 2^{8+1} \Rightarrow cmax > 2^9 \Rightarrow cmax > 512$ . A fim de obter um valor da frequência do clock redondo, escolheu-se um valor de  $cmax$  igual a 999 que satisfaz a inequação. Desta forma, tem-se que:

$$fs = \frac{f_{clk}}{cmax + 1} \Rightarrow f_{clk} = fs(cmax + 1) = 8000(999 + 1) \Rightarrow f_{clk} = 800kHz$$

Após a definição dos parâmetros, é possível iniciar o desenvolvimento da lógica em Verilog para replicar o circuito desejado para gerar o PWM. Para evitar falhas e comprovar o funcionamento do módulo geral, foi decidido programar um módulo para cada “componente” do circuito completo, ou seja, programou-se um módulo separado para o contador, a amostragem e o comparador. Por fim, juntou-se ambos para o módulo geral do PWM.

Primeiramente, foi criado o módulo das amostras, dado por:

```
1  module amostras(
2      input clk,
3      input [13:0] A,
4      output reg [11:0] amostra
5  );
6
7  reg [7:0] mem [0:799];
8  initial begin
9      $readmemb("C:/Users/linco/Desktop/PWM/C.txt",mem);
10 end
11 always @(A) begin
12     amostra = mem[A];
13 end
14
15 endmodule
16
```

Onde a ideia é criar um vetor 800x8 que seja capaz de armazenar os valores, em que cada uma dessas posições serão preenchidas com as amostras codificadas do sinal modulante decorrente do processo de PCM através de um arquivo .txt, onde também o valor de saída deste módulo será controlado pelo valor de entrada “A”, onde a saída será dada pelo valor da posição “A” da memória. Por exemplo, se o valor de “A” é 1, a saída amostra será o valor da memória na posição 1.

Já o módulo do contador é dado por:

```
1  module contador(
2      input clk,
3      output reg [11:0] cont,
4      output reg [13:0] A
5  );
6      initial cont = 12'd0;
7      initial A = 14'd0;
8
9  always @(posedge clk) begin
10    cont <= cont + 12'd1;
11    if(cont >= 12'd999)begin
12        cont <= 12'd0;
13        A <= A + 14'd1;
14        if(A == 14'd799) A <= 14'd0;
15    end
16 end
17 endmodule
```

Como sabemos, o contador é incrementado com o clock, de 1 em 1 e possui valor máximo de 999 e assim que ele passa esse valor ele volta a valer 0 e recomeça todo o processo. Além disso, o contador também deve servir para indicar qual posição da memória, isto é, qual amostra, deve ser comparada para gerar o PWM. Ou seja, quando o contador reinicia deve-se atualizar a amostra utilizando a próxima. Isso é feito incrementando 1 na variável de controle da posição da amostra toda vez que o valor do contador é extrapolado. Como há no total 800

amostras, esse é processo é realizado até chegar neste valor, que então faz com que o valor da posição da amostra volte a 0, reiniciando assim o ciclo.

Assim temos o módulo de comparação, que é dado por:

```
1  module comparador(
2    input [11:0] amostra,
3    input [11:0] cont,
4    output reg pwm
5  );
6
7  always @(*) begin
8    if(amostra > cont) pwm = 1;
9    else pwm = 0;
10   end
11
12 endmodule
```

Cuja função do módulo é comparar os valores de entrada referentes à amostra atual e do contador e, se o valor absoluto da amostra é maior do que o do contador, a saída PWM é igual a 1(nível lógico alto). Caso contrário, a saída PWM é 0 (nível lógico baixo).

Escolheu-se o nível lógico igual 1 por ser o padrão. Desta forma, quando o sinal for recuperado ele terá o seu valor entre 0 e 1 ainda que a forma de onda seja mantida.

Na verdade, o seu valor máximo terá aproximadamente o valor de  $\frac{\text{valor máximo da amostra}}{\text{valor máximo do contador}} * 1 = \frac{255}{999} = 0,255$ , já que a amostra com o valor de 255 só terá o valor lógico 1 no período do PWM durante uma fração de 0,255 do período deste. Ou seja, para se chegar ao valor correto na reconstrução do sinal será necessário aplicar um ganho juntamente com o filtro passa baixa, o que será mais explicado na próxima etapa do trabalho.

Para realizar a interconexão desses três módulos no módulo principal, temos um módulo dado por:

---

```
1  module PWM (
2    input clk,
3    output saida
4  );
5
6  (*keep=1*) wire [11:0] cont;
7  (*keep=1*) wire [11:0] amostra;
8  (*keep=1*) wire [13:0] A;
9  contador C(clk, cont, A);
10 amostras Amos(clk, A, amostra);
11 comparador comp(amostra, cont, saida);
12
13 endmodule
```

Em que a expressão (\*keep=1\*) é feita apenas para melhor visualização dessas variáveis na simulação. A conexão do módulo é realizada da seguinte forma: a saída do módulo contador “A” vai para a entrada do módulo amostras para indicar qual posição da memória e, consequentemente, qual amostra deverá ser comparada. Então essa amostra e a saída “cont” do módulo contador vão para a entrada do módulo comparador onde são comparadas com intuito de gerar o sinal PWM.

O código de simulação é mostrado abaixo, onde se implementa um clock com período de 1250 ns, ou seja, com frequência de 800 kHz:

```

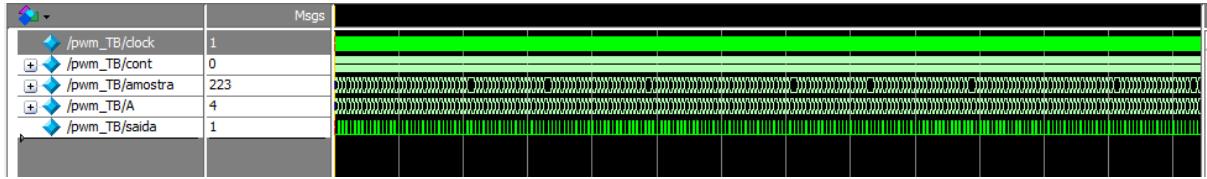
1 `timescale 1ns/10ps
2
3 module PWM_TB;
4   reg clock;
5   reg [11:0] cont;
6   reg [11:0] amostra;
7   reg [13:0] A;
8   wire saida;
9
10  DUT(
11    .clk(clock),
12    .saida(saida)
13  );
14
15  initial begin
16    clock = 0;
17  end
18
19  always begin
20    #625 clock = ~clock;
21  end
22
23  initial begin
24    $init_signal_spy("/PWM_TB/DUT/cont", "cont", 1);
25    $init_signal_spy("/PWM_TB/DUT/amostra", "amostra", 1);
26    $init_signal_spy("/PWM_TB/DUT/A", "A", 1);
27  end
28
29  initial begin
30    #1005000000 $stop;
31  end
32
33
34 endmodule

```

Também é possível ver que o valor definido para a frequência do clock é exatamente o calculado anteriormente:

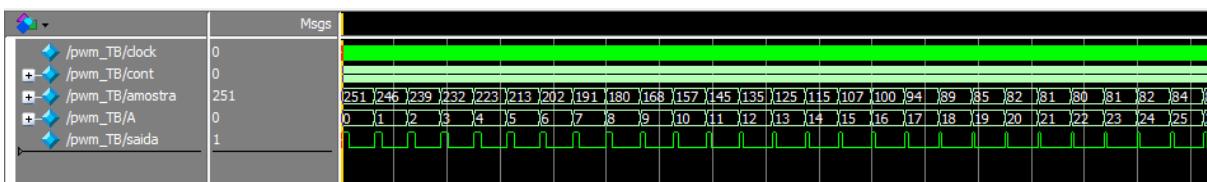
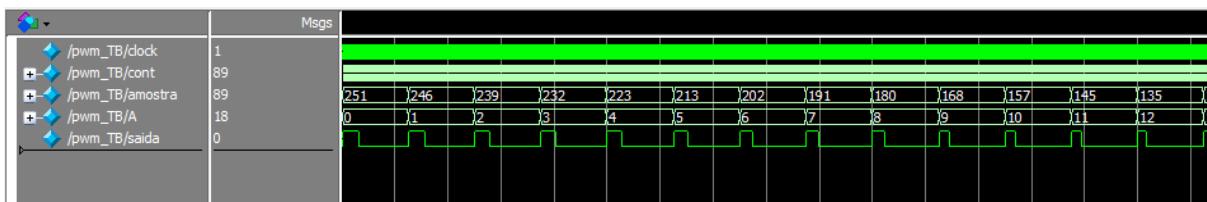
	Clock Name	Type	Period	Frequency	Rise	Fall
1	clk	Virtual	1250.000	0.8 MHz	0.000	625.000

O resultado da simulação é mostrado a seguir:



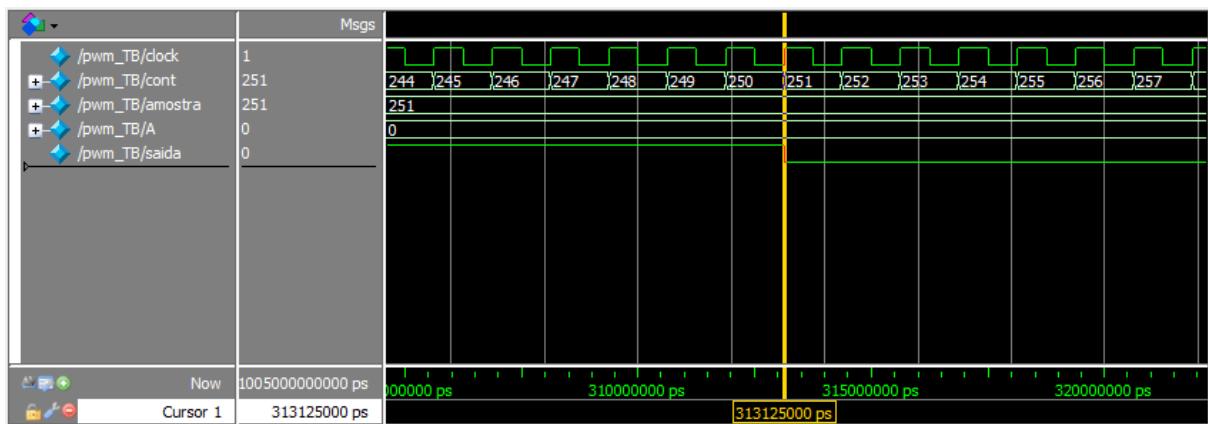
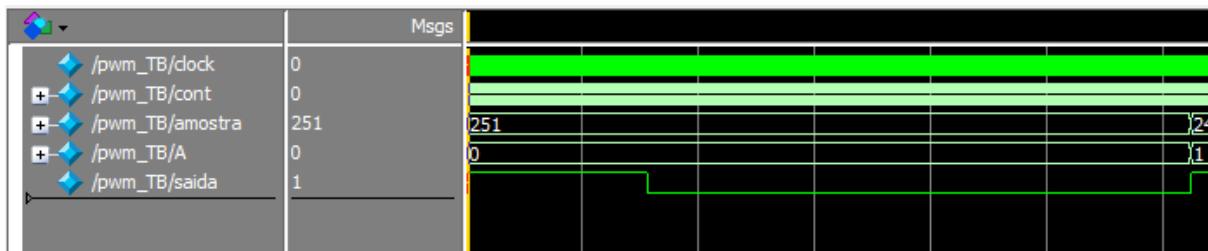
Sabendo que, com a visão original não seria possível realizar a análise, dado que o vetor de saída PWM tem um tamanho relativamente grande, será necessário dar um zoom para poder fazer uma análise mais qualificada.

Assim, após o uso do zoom para visualizar as primeiras 12 amostras do sinal modulante codificado e, em seguida as 25 primeiras, percebe-se que:

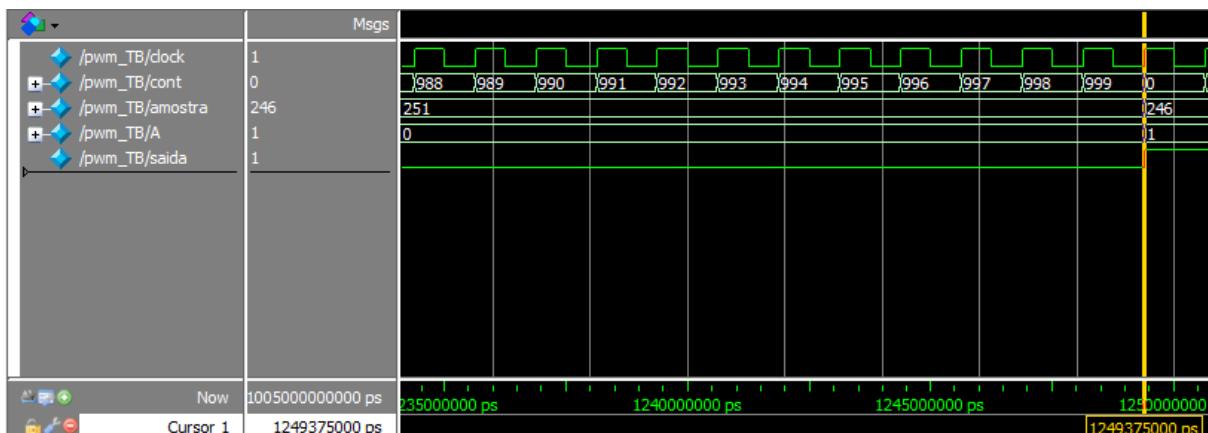


É possível observar que, conforme o valor absoluto da amostra vai diminuindo, o período em que o sinal PWM fica em estado alto também diminui, assim comprovando o que se esperava. Podemos verificar a validade do funcionamento deste módulo analisando o comportamento da onda de PWM em uma única amostra, através da comparação entre a proporção do tempo que a saída fica em nível lógico alto e o tempo total do período da onda PWM para essa amostra em específico e a proporção do valor absoluto da amostra com o valor máximo do contador igual a 999. Em teoria, eles deveriam ser iguais ou ao menos muito próximos.

Escolhendo a primeira amostra para fazer esse teste, tem-se que:



Ou seja, o sinal PWM muda de nível lógico em 0,313125 ms. Por fim, o contador extrapola no seguinte estante:



Ou seja, em 1,249375 ms, o valor ideal seria em 1,25 ms correspondente ao período de amostragem. No entanto, como se utilizou da simulação em gate level, levou-se em conta

desta forma os atrasos inerentes aos componentes utilizados para realizar a simulação além também da própria resolução utilizada.

De qualquer forma, as proporções ficam da seguinte forma:

$$\frac{\text{valor da amostra}}{\text{valor máximo do contador}} = \frac{251}{999} = 0,2512$$

$$\frac{\text{tempo em nível lógico alto}}{\text{período da onda para a amostra}} = \frac{0,313125 \text{ ms}}{1,249375 \text{ ms}} = 0,2506$$

Ou seja, houve uma diferença entre as proporções de apenas 0.0006, um valor relativamente bem baixo, comprovando assim a eficácia do módulo proposto.

Após a realização da implementação no Quartus, será implementada a mesma ideia no no Scilab.

Onde os mesmos valores das variáveis definidas anteriormente serão empregados, sem a aplicação de um valor “físico” em relação às operações realizadas para chegar ao resultado.

Inicialmente, o modo pelo qual o vetor das amostras do sinal modulante codificado não o permite fazer operações com cada uma delas em específico, dado que cada bit no total de 8 para cada amostra ocupa uma posição na matriz em específico. Para resolver esse problema faremos a conversão de cada uma das amostras binárias em números decimais, de forma que cada amostra ocupe apenas uma posição no vetor.

Isso é realizado da seguinte forma:

```
--> for i = 1:800
>   Bin(i) = c(i,1)*(2^7) + c(i,2)*(2^6) + c(i,3)*(2^5) + c(i,4)*(2^4) + c(i,5)*(2^3)+c(i,6)*(2^2)+c(i,7)*(2^1)+c(i,8)*(2^0);
> end
```

$$(cmax + 1) * \text{Número de amostras} = 1000 * 800 = 800000 \text{ amostras},$$

dado que cada amostra terá uma comparação com um valor do contador que vai de 0 até 999, constrói-se um vetor de zeros chamado de PWM para armazenar cada um dos 800000 valores que ainda serão calculados.

```
PWM = zeros(1,800000);
```

Agora aplica-se o algoritmo para fazer a comparação entre o valor das amostras e o contador. Desta forma será usado um “for” de 1 a 800 para varrer cada uma das amostras e outro “for” de 1 a 1000 para varrer todos os valores possíveis do contador.

Além disso, é feito a comparação entre o valor do contador e o valor da amostra no índice indicado pelo for através de um “if”. Falta apenas a concatenação de cada operação de comparação com os valores abstraídos do PWM, pois, quando extrapolado o valor do contador passa-se para a seguinte posição do vetor de amostras e assim começa um novo ciclo de obtenção dos valores do PWM. Para resolver isso, usa-se de uma variável auxiliar a fim de corrigir a posição do PWM, isto é, conforme o índice da posição do vetor de amostras aumenta em 1 o índice de posição do vetor de PWM deverá ser incrementado em 1000, justamente o valor de extração do contador. Este algoritmo é apresentado a seguir:

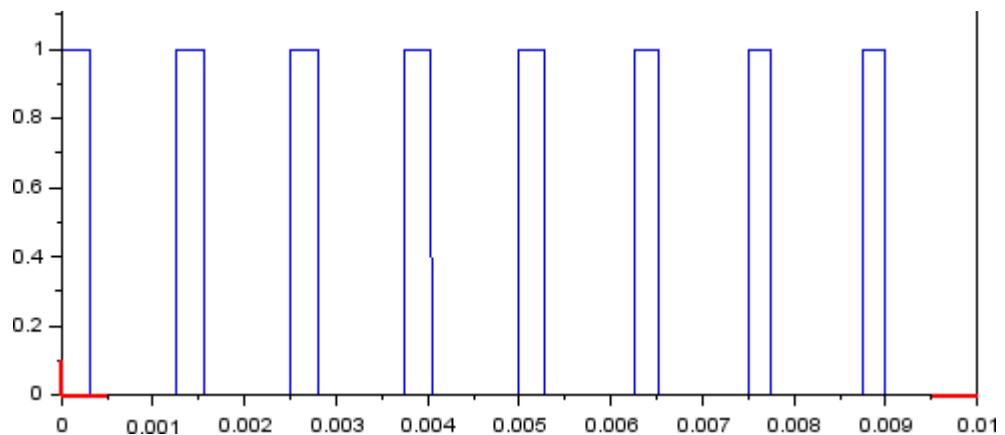
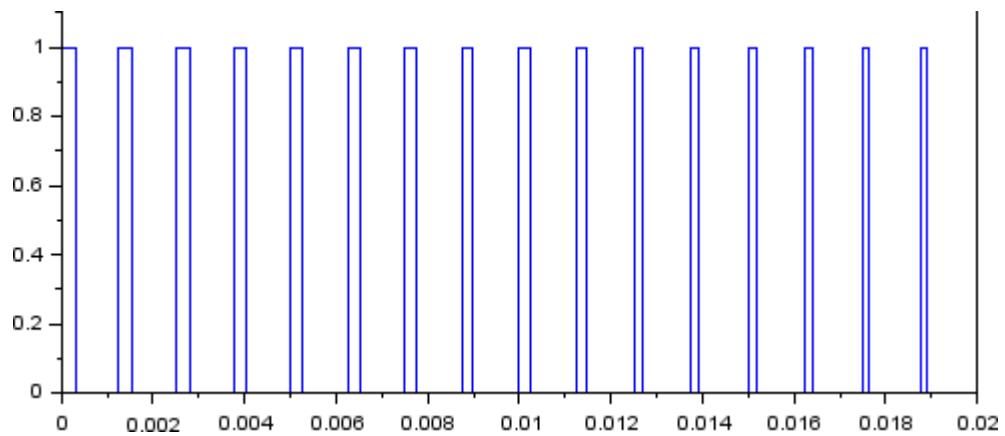
```
--> for i=1:800
> for cont = 1:1000
> j = i-1;
> aux = cont + 1000*j;
> if(cont < Bin(i))
> PWM(aux)=1;
> else
> PWM(aux)=0;
> end
> end
> end
```

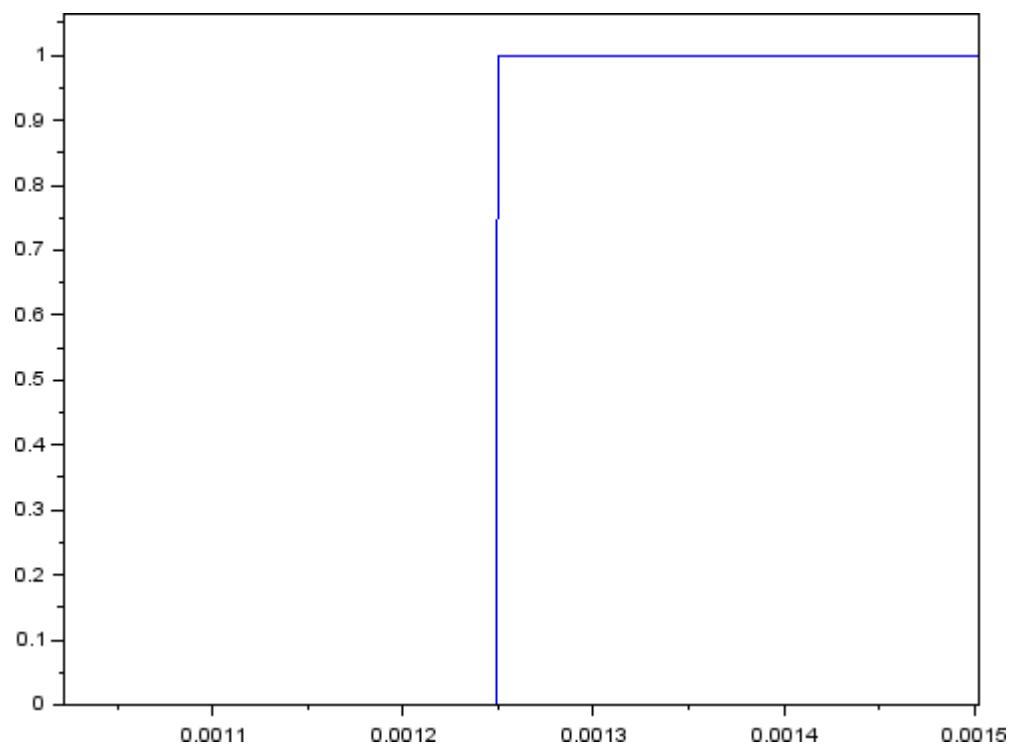
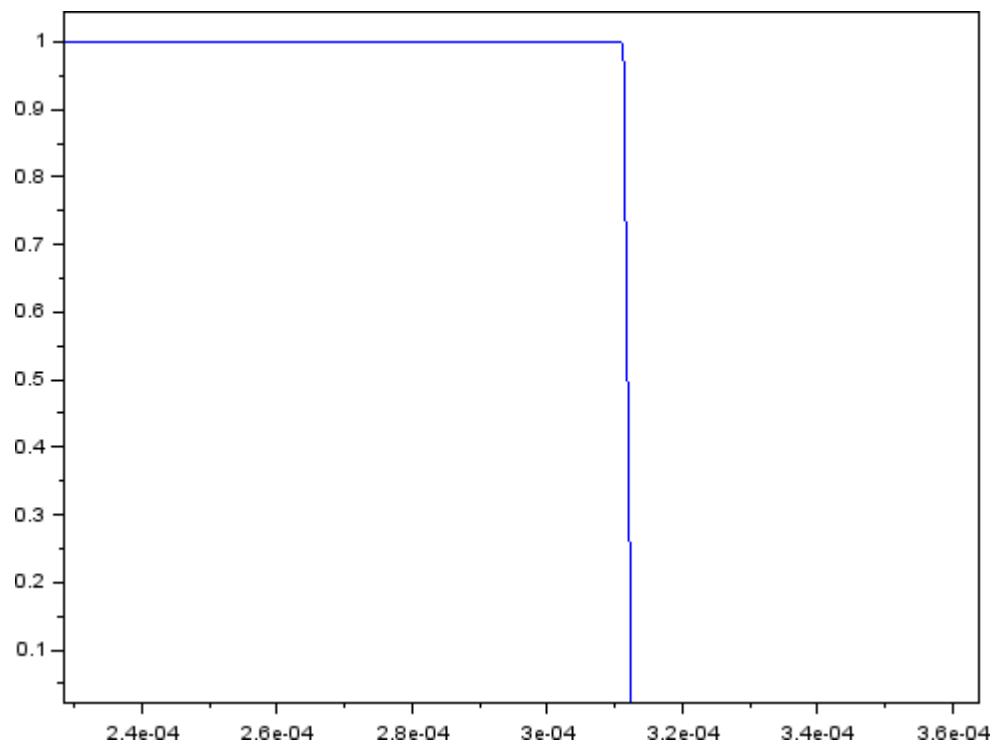
```
fclk = 1000*800;

tclk = 1/fclk;

t = 0:tclk:1-tclk;

plot(t,PWM)
```





Dessa forma, as proporções ficam da seguinte forma:

$$\frac{\text{valor da amostra}}{\text{valor máximo do contador}} = \frac{251}{999} = 0,2512$$

$$\frac{\text{tempo em nível lógico alto}}{\text{período da onda para a amostra}} = \frac{0,0003125}{0,00125} = 0,25$$

Concluímos que, houve uma leve diferença entre as proporções de 0.0012, um valor relativamente baixo, comprovando também assim a eficácia do algoritmo implementado. Além disso, pode-se notar também a correspondência entre o PWM implementado com o Quartus com o implementado no Scilab, com ambos dando resultados praticamente idênticos.

Uma vez que o sinal PWM foi obtido a partir da aplicação do PCM no sinal modulante é necessário agora filtrar esse sinal com um filtro passa-baixa a fim de converter as amostras para o sinal analógico correspondente ao sinal modulante original, ou seja, a aplicação do filtro passa-baixa sobre o sinal PWM retorna ao sinal original. Essa recuperação do sinal analógico baseia-se no teorema da amostragem e para o caso do filtro passa baixa, o respeito ao teorema de Nyquist, que já foi reportadamente realizado neste trabalho.

Antes de estabelecer os parâmetros para criar a função de transferência do sinal é necessário primeiro entender o sinal PWM de entrada para esse filtro e a sua relação com a amplitude do sinal original. O modo pelo qual se projetou o sinal PWM foi o seguinte: um comparador comparava o valor absoluto de cada amostra obtida pelo PCM, um total de 800, com um contador que ia de 0 a 999. Se o valor da amostra fosse maior do que o do contador a saída PWM era igual a 1 (nível lógico alto), se contrário a saída era igual a 0 (nível lógico baixo). Essa escolha do valor das saídas foi feita pois corresponde a um padrão regular para um PWM, pensando em um caso real.

Entretanto, o que deve ser considerado para esse sistema é que desta forma, quando o sinal for recuperado ele terá o seu valor entre 0 e 1, ainda que a forma de onda seja mantida. Na verdade, o valor máximo do sinal recuperado, sabendo que cada amostra PCM possui 8 bits e portanto esta só pode ir até 255, terá aproximadamente o valor de:

$$\frac{\text{valor máximo da amostra}}{\text{valor máximo do contador}} * 1 = \frac{255}{999} = 0,255$$

já que a amostra com o valor de 255 só terá o valor lógico 1 no período do pwm durante uma fração de 0,255 do período deste. A mesma lógica pode ser aplicada para achar o valor mínimo do sinal recuperado. Primeiro se acha o valor mínimo do vetor de amostras PCM, que dá:

```
--> amin = min(Bin)
amin =
43.
```

Onde se obteve o mínimo valor do vetor das amostras convertidas em números decimais. Ou seja, o valor mínimo do sinal recuperado terá aproximadamente o valor de:

$$\frac{\text{valor máximo da amostra}}{\text{valor máximo do contador}} * 1 = \frac{43}{999} = 0,043$$

já que a amostra com o valor de valor lógico 1 no período do pwm durante uma fração de 0,043 do período deste. Concluindo então, quando o sinal for recuperado ele terá o seu valor aproximadamente entre 0,043 e 0,255, contrastando com o sinal original onde seu máximo e mínimo são:

```
--> mmax = max(abs(m))
mmax =
4.9193602

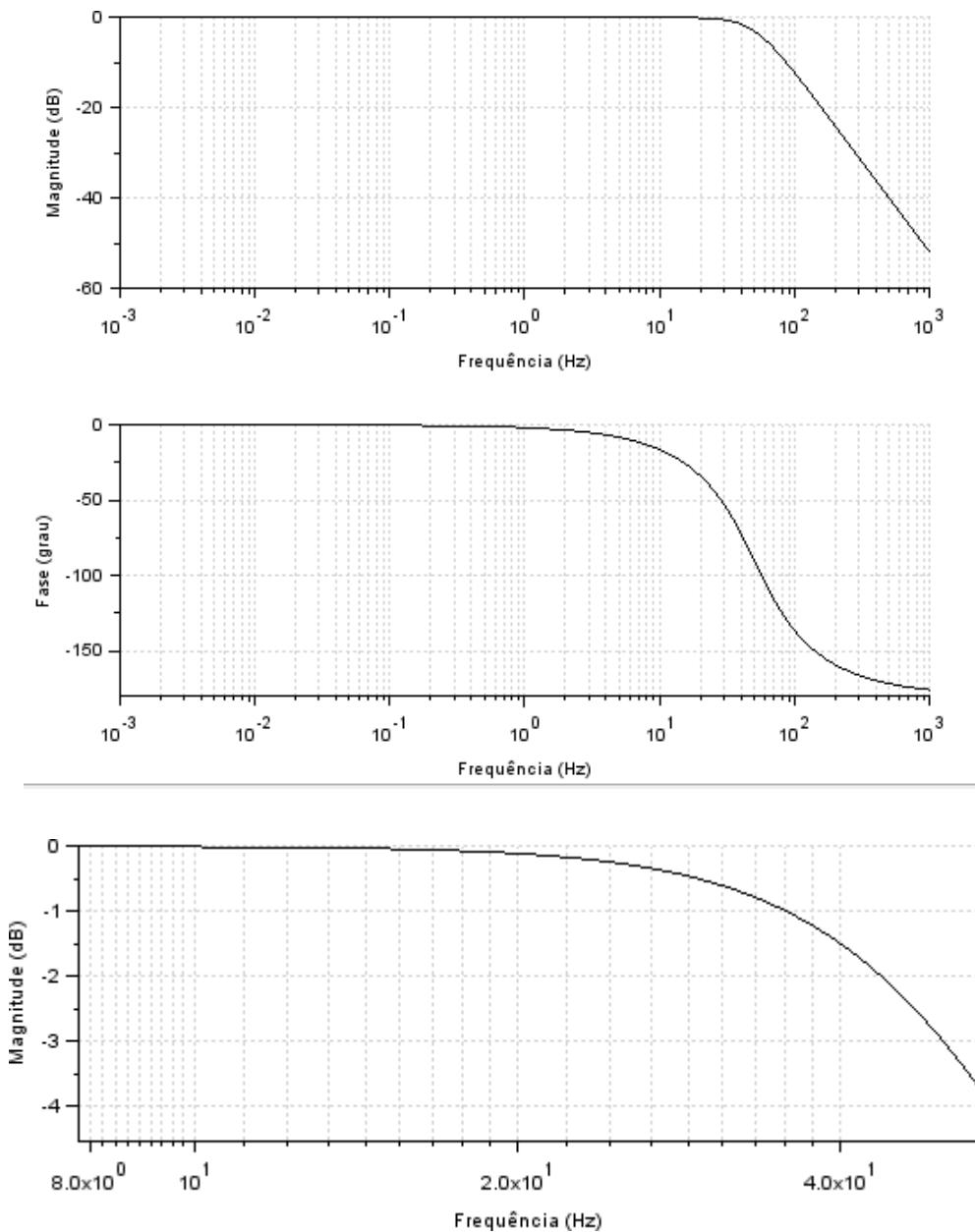
--> mmin = min(m)
mmin =
-3.2496454
```

Ou seja, o sinal vai de aproximadamente -3,25 até 4,92.

As soluções para esse problema serão discutidas posteriormente. Já para a construção do filtro passa baixa, será usado a função pronta do Scilab chamada de analpf() onde se passa como parâmetros principais a ordem do filtro, o tipo dele e a frequência de corte. Como se quer recuperar o sinal modulante cujas componentes possuem respectivamente frequências de 10Hz e 20Hz, pode-se escolher como frequência de corte 50Hz que se conseguirá excluir uma ampla faixa de altas frequências e produzirá uma perda de ganho ínfima para as frequências de interesse. Quanto à ordem do filtro, escolhe-se o de segunda ordem a fim de produzir uma maior atenuação do sinal conforme se passa da frequência de corte, e do tipo butterworth que consegue produzir um bom resultado com relação à resposta transitória. Pode-se analisar o filtro de uma melhor forma visualizando o seu diagrama de Bode, do seguinte modo:

```
--> fpb = analpf(2, 'butt', [0 0], 2*pi*50)
fpb =
98696.044
-----
2
98696.044 + 444.28829s + s

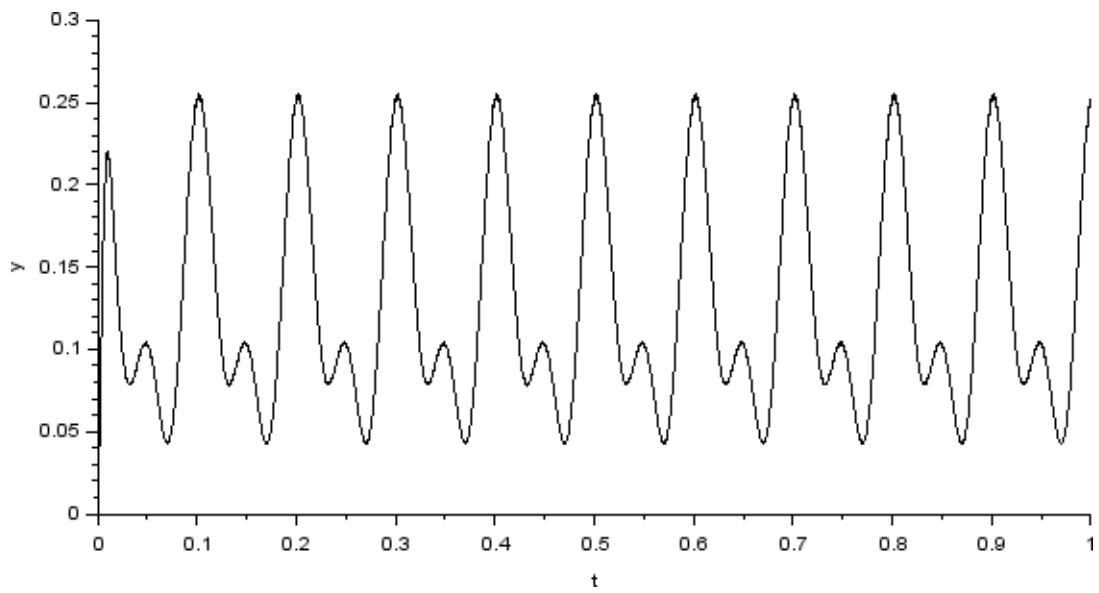
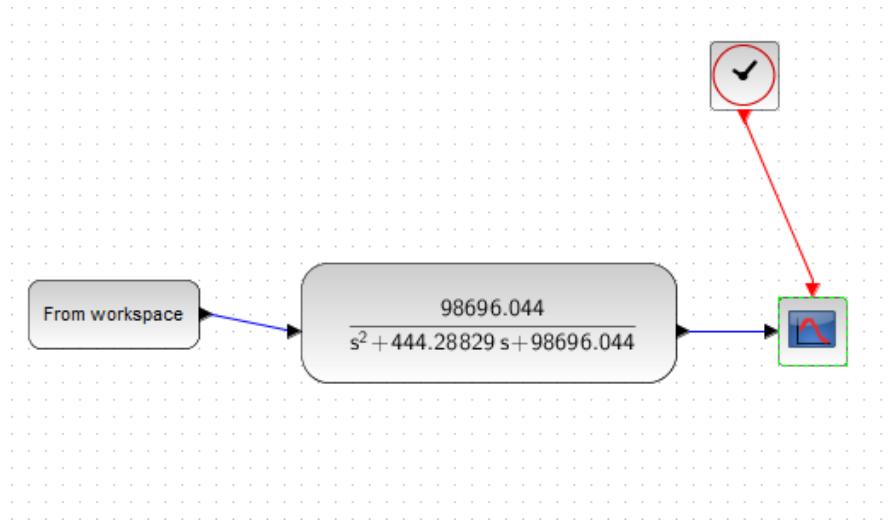
--> bode(fpb)
```



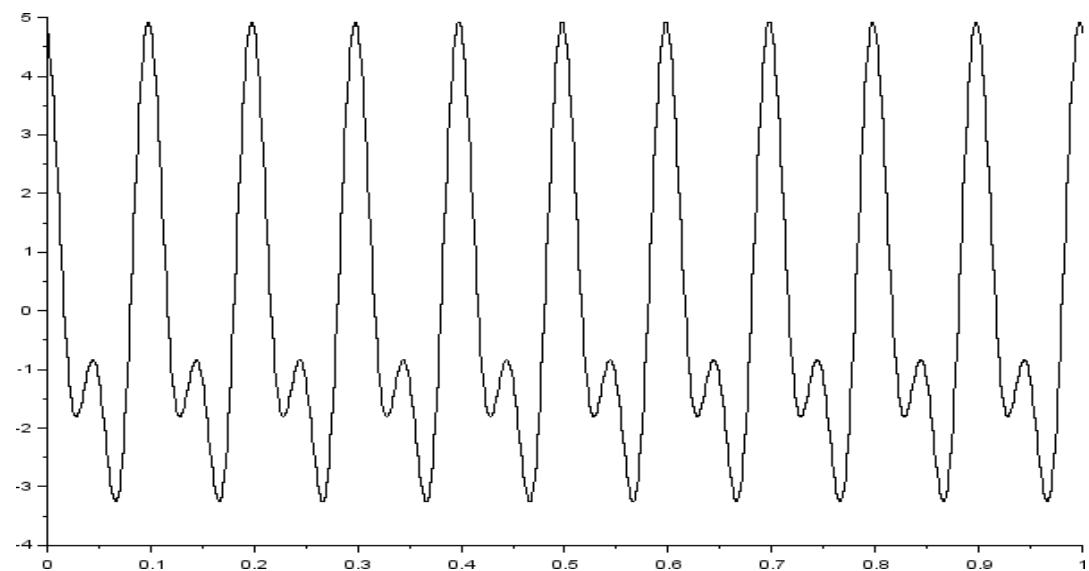
Onde pode-se ver que a magnitude em 10Hz e 20Hz é muito próxima a 0dB, ou seja, sematenuação do ganho.

Com o sinal PWM e a função do filtro em mãos já é possível reconstruir o sinal através de Xcos do Scilab, onde é necessário utilizar o comando struct no console para passar os valores de cada amostra do sinal pwm e do vetor temporal para o bloco from workspace do xcos. Após isso, basta organizar os blocos junto com o clock e o osciloscópio para medir o sinal resultante. Isso é feito da seguinte forma:

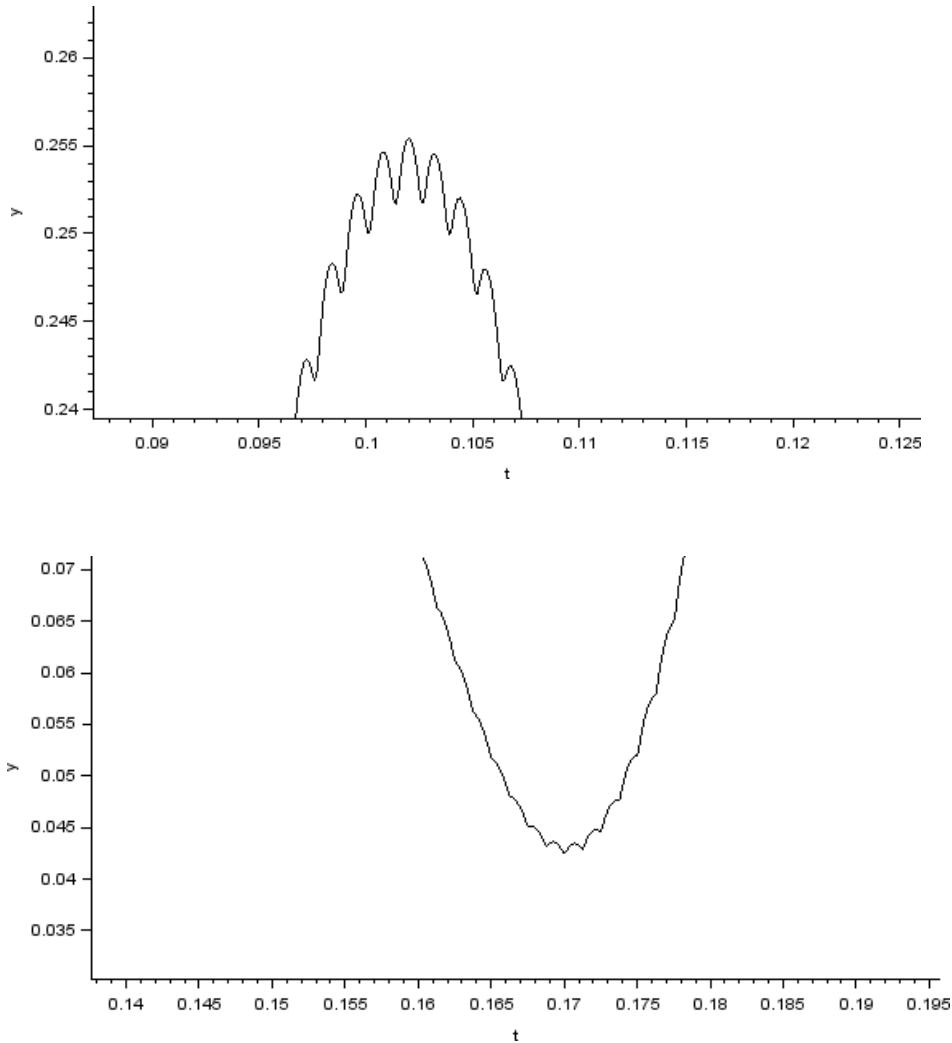
```
--> pwm = struct('time',t,'values',PWM);
```



Onde pode se comparar com a onda original, dada por:



Percebe-se que de fato, tirando os primeiros momentos de estabilização, a onda recuperada possui a mesma forma da onda original, mas como já explicado anteriormente, uma menor amplitude. Os valores máximo e mínimos podem ser conferidos, dando um zoom na onda reconstruída:



Onde percebe-se que o valor máximo é em volta de 0,255 e o mínimo muito próximo de 0,043, justamente os valores calculados anteriormente para a aplicação de pwm com o contador indo de 0 a 999 e saída de 0 ou 1. Além disso, percebe-se também que a onda reconstruída, quando é dado um zoom maior, não possui um aspecto linear “limpo”, mas um aspecto senoidal, resultante dos métodos aplicados para a sua criação.

Comprovado então o funcionamento do filtro, pode-se partir então para resolver o problema da redução da amplitude do sinal recuperado, onde para se chegar ao valor correto na reconstrução do sinal será necessário aplicar um ganho juntamente com o filtro passa baixa. Esse ganho será nada mais do que um multiplicador, isto é, um controlador proporcional.

Esse ganho simplesmente multiplicará o valor de saída das amostras do pwm por um fator x, da seguinte forma, com o exemplo do maior e menor valor das amostras:

$$\begin{cases} 0,255 * x = y \\ 0,043 * x = z \end{cases}$$

Essas multiplicações resultam em y e z, constantes maiores do que 0 que por sua vez serão os valores que serão aplicados no filtro passa-baixa.

Para o caso original realizado anteriormente, esse ganho era simplesmente 1, resultando que y e z fossem respectivamente iguais a 0,255 e 0,043.

Bom, uma abordagem inicial poderia ser definir o valor de y como o máximo valor do sinal modulante, igual a 4,919 e daí calcular o fator x. Fazendo isso, chega-se a:

$$x = \frac{4,919}{0,255} = 19,29$$

Entretanto, utilizando esse fator para a segunda equação do sistema, obtém-se:

$$0,043 * 19,29 = 0,83,$$

que é muito diferente do valor mínimo da onda modulante, igual a -3,25. O problema pelo qual fará essa abordagem nunca funcionar é que é impossível fazer o valor mínimo resultante ser negativo, igual ao valor mínimo da onda modulante, pois o ganho à priori deve ser positivo. Isso indica que, para recuperar exatamente o sinal original, além de aplicar um ganho na saída PWM das amostras, será também necessário aplicar um offset na onda resultante, a fim de subtrair desta um valor a ser determinado com o objetivo de fazer que a onda também possua valores negativos da mesma forma que a original, situação que não era possível na primeira abordagem visto que a saída do pwm é limitada de 0 a 1, e para as amostras utilizadas, possui um mínimo de 0,043.

Para corrigir esse problema, a diferença entre o valor máximo e mínimo da onda resultante deverá ser igual ao da onda original, dada por:  $4,919 - (-3,25) = 8,169$ . Assim, poderá ser aplicado o offset exatamente para os pontos que se desejam. Desta forma, o sistema de equações atualizado é dado por:

$$\begin{cases} 0,255 * x = y \quad (1) \\ 0,043 * x = z \quad (2) \\ y - z = 8,169 \end{cases}$$

Subtraindo a equação (1) pela (2), tem-se que:

$$0,255x - 0,043x = y - z = 8,169$$

$$\Rightarrow 0,212x = 8,169 \Rightarrow x = 38,53$$

E eventualmente:

$$y = 0,255 * 38,53 = 9,825$$

$$z = 0,043 * 38,53 = 1,65$$

Indicando que os valores máximo e mínimo da onda resultante serão de 9,825 e 1,657 respectivamente. Para calcular o offset a ser aplicado, basta pegar um ponto da onda original e subtraí-lo do ponto equivalente na onda resultante. Fazendo isso para o ponto mínimo da onda modulante, tem-se que:

$$m_{min} - z = \text{offset} \Rightarrow \text{offset} = -3,25 - 1,657 = -4,907$$

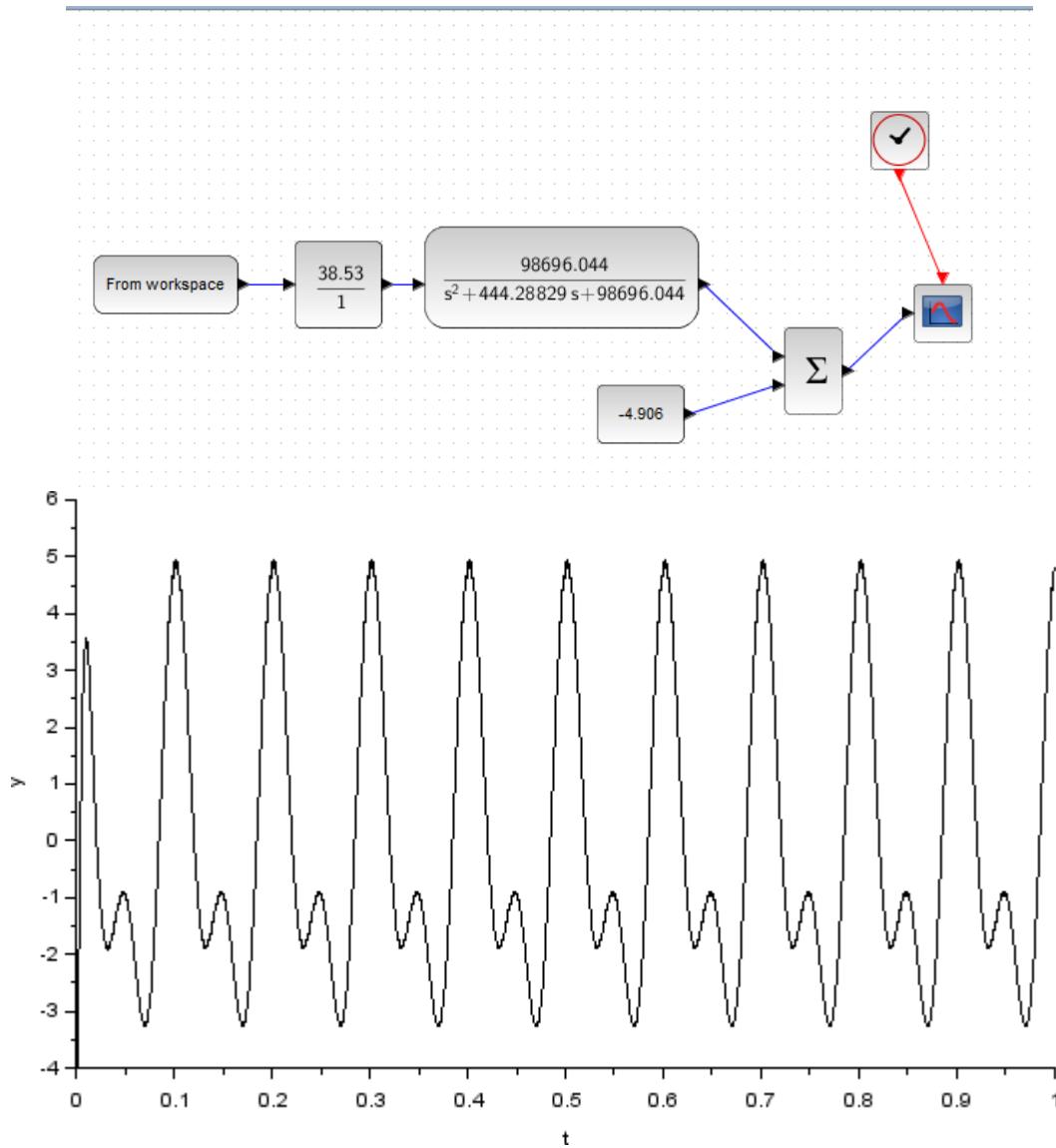
Isso poderia ser feito com qualquer ponto escolhido da onda original, que deveria dar o mesmo valor de offset. Repetindo o procedimento agora com o valor máximo da onda modulante, tem-se que:

$$m_{max} - y = \text{offset} \Rightarrow \text{offset} = -4,919 - 9,825 = -4,906$$

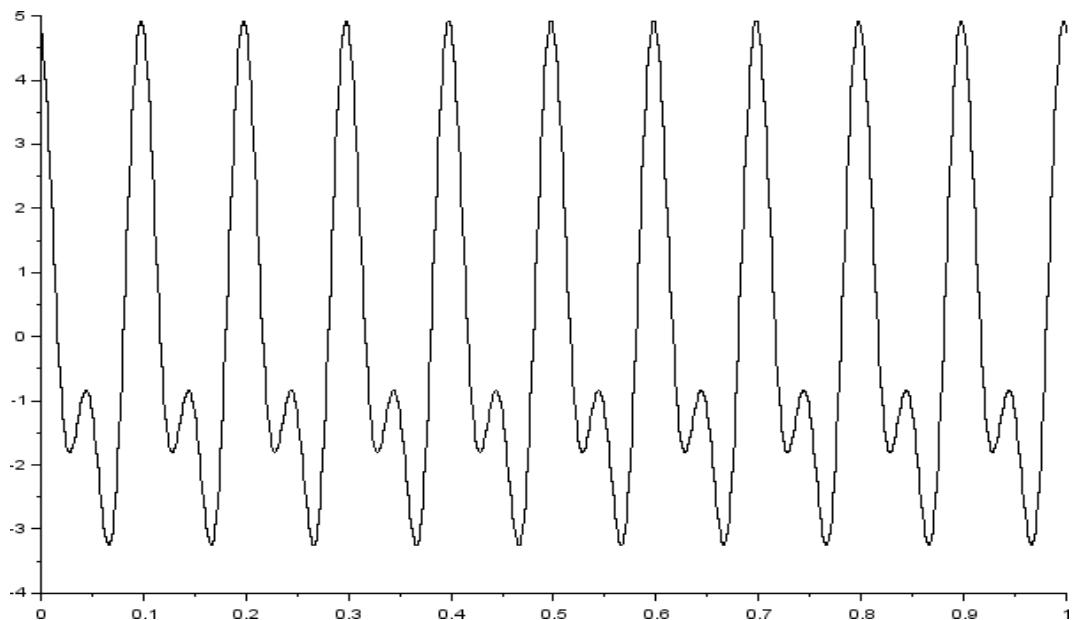
Ou seja, valores iguais (a diferença mínima se dá devido a aproximações nos cálculos e arredondamentos).

Desta forma, o controlador proporcional colocado no sistema terá um ganho de 38,53 e se aplicará um offset de -4,906 na onda resultante.

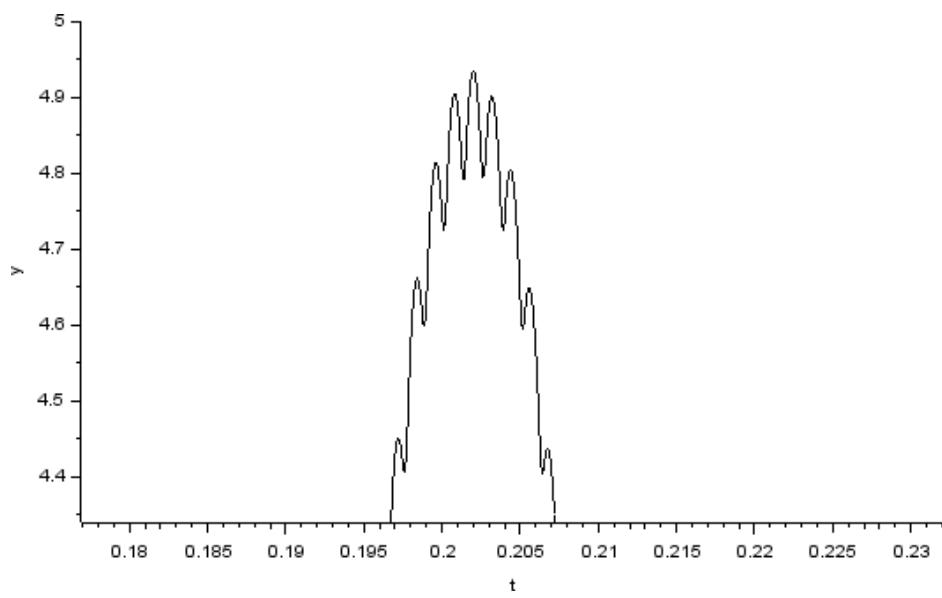
A nova simulação no Xcos fica da seguinte forma então:

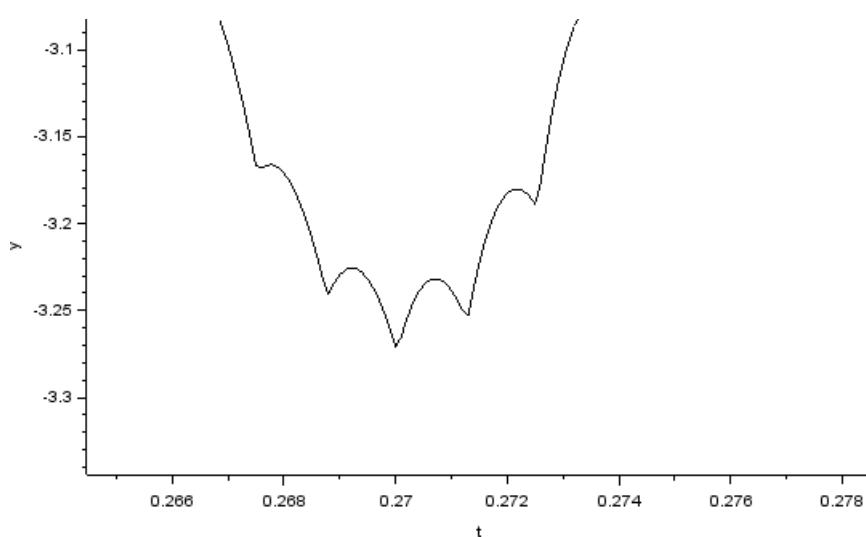
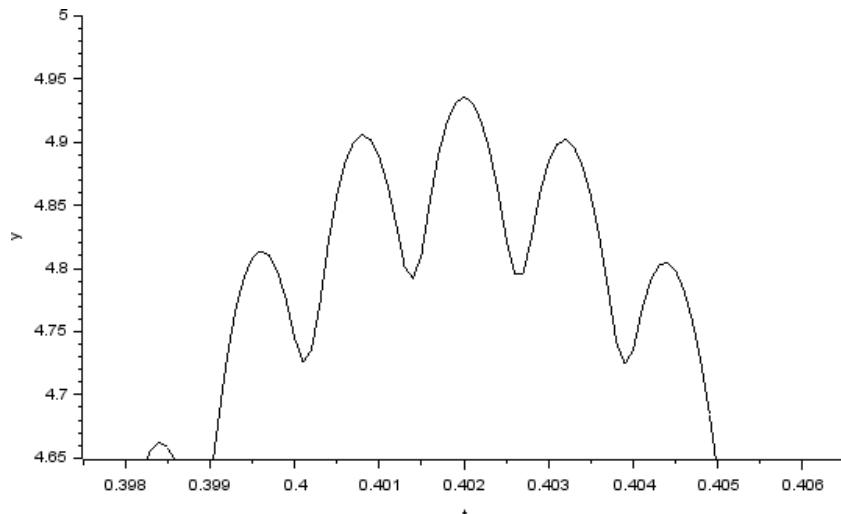


Que pode ser novamente comparada com a onda original, dada por:



Onde pode se ver uma clara correspondência entre uma e outra, que pode ser ainda maisconfirmada conferindo os pontos de máximo e mínimo da onda recuperada, dados por:



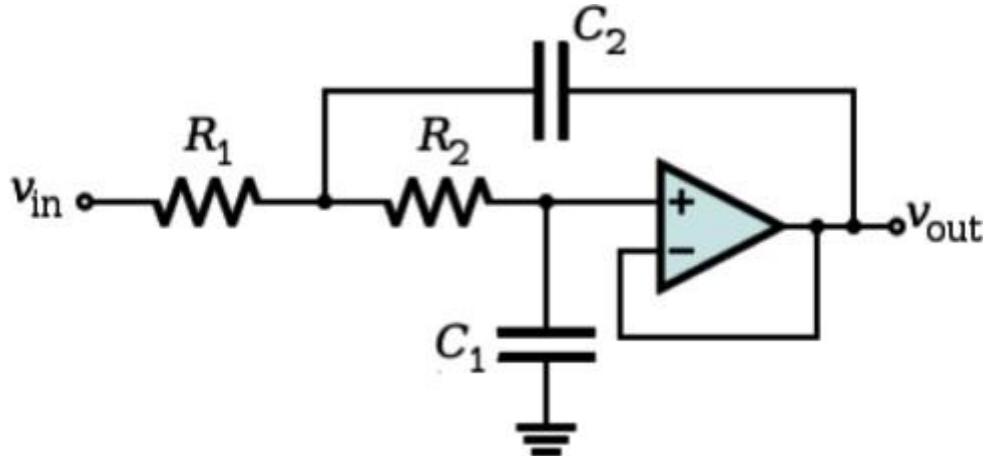


Onde pode-se ver que o valor máximo da onda recuperada é de aproximadamente 4,935 e o mínimo é de -3,263, ou seja, valores muito próximos daqueles da onda modulante original. O erro produzido, na cada de 0,015, é resultante das aproximações realizadas nos cálculos dos fatores do ganho e do offset, além também do erro existente no filtro.

Com isso, pode-se confirmar a eficácia do sistema em recuperar o sinal original através da aplicação de um filtro passa-baixa no sinal de saída do pwm juntamente com um controlador proporcional e um offset.

No entanto, é necessário calcular o valor das componentes para fazer o filtro passa-baixa e escolher valores comerciais para estes. Deve-se fazer o mesmo para o cálculo do circuito de amp op para fazer o ganho proporcional.

O projeto começa definindo o filtro passa-baixa de segunda ordem do tipo Butterworth dado pelo seguinte circuito:



A função de transferência no domínio da frequência é:

$$H(s) = \frac{(2\pi f_c)^2}{s^2 + 2\pi f_c Q s + (2\pi f_c)^2}$$

Onde  $f_c$  é a frequência de corte, escolhida como 50Hz e  $Q$  é o fator de qualidade que determina o formato de resposta do filtro e é igual a 0,707 para o tipo Butterworth. Desta forma, tem-se que:

$$H(s) = \frac{(2\pi 50)^2}{s^2 + \frac{2\pi 50}{0,707} s + (2\pi 50)^2} = \frac{98.696,044}{s^2 + 444,355 s + 98.696,044}$$

Ou seja, a função é praticamente idêntica a aquela obtida utilizando o comando analpf() do Scilab.

Além disso, a frequência de corte  $f_c$  do filtro é dada por:

$$f_c = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}}$$

Fazendo  $R_1 = R_2$  e  $C_2 = 2C_1$ , tem-se que:

$$f_c = \frac{1}{2\pi\sqrt{2R_1 C_1}} = 50 \Rightarrow R_1 C_1 = \frac{1}{2\pi\sqrt{2} * 50} = 0,002251s$$

Escolhendo  $C_1$  com o valor comercial de  $150\text{nF}$  segue que:

$$R_1 = \frac{0,002251}{150 * 10^{-9}} = 15.006,66\Omega$$

Cujo valor comercial mais próximo é de  $15\text{k}\Omega$ , logo  $C_1 = 150\text{nF}$  e consequentemente:  $C_2 = 2C_1 = 300\text{nF}$ , que é um valor comercial. Desta forma tem-se que:

$$\begin{aligned} R_1 &= 15\text{k}\Omega \\ R_2 &= 15\text{k}\Omega \\ C_1 &= 150\text{nF} \\ C_2 &= 300\text{nF} \end{aligned}$$

O fator de qualidade Q do filtro é dado por:

$$Q = \frac{1}{2\pi f_c C_1 (R_1 + R_2)} = \frac{1}{2\pi 50 * 150 * 10^{-9} (30000)} = 0,7073$$

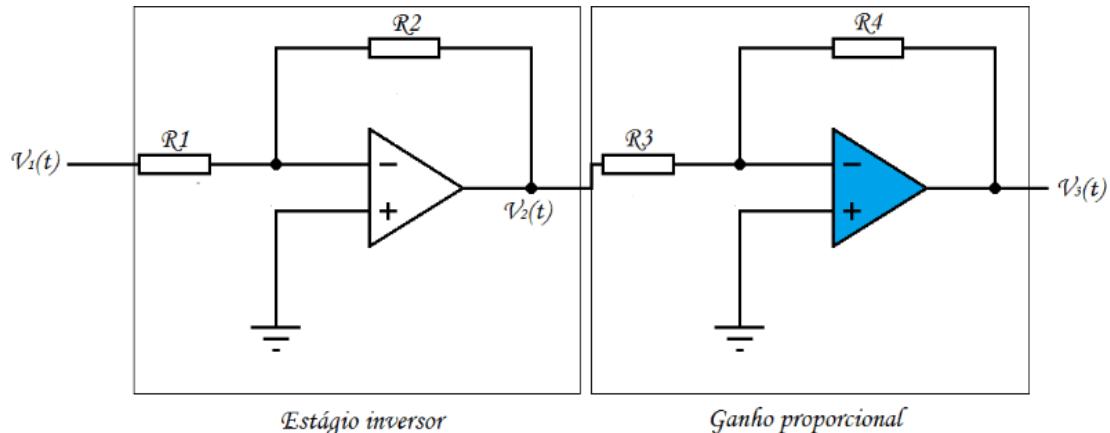
Ou seja, conseguiu-se um fator de qualidade quase ideal com os valores selecionados. A nova função de transferência é dada por:

$$\begin{aligned} H(s) &= \frac{1}{1 + C_1 (R_1 + R_2) s + C_1 C_2 R_1 R_2 s^2} = \frac{1}{0,000010125s^2 + 0,0045s + 1} \\ &= \frac{98.765,43}{s^2 + 444,45s + 98.765,43} \end{aligned}$$

Agora é necessário projetar o controlador proporcional cujo ganho deve ser de 38,53. Ele pode ser feito a partir de dois ampops inversores em série, onde o ganho é dado pela fórmula de:

$$v_0 = -\frac{R_2}{R_1} v_i$$

onde observa-se que o sinal de saída possui polaridade inversa em relação ao sinal de entrada e por isso o nome de inversor. O primeiro ampop inversor serve apenas para fazer a polaridade do sinal de saída igual à da entrada, onde escolhe desta forma  $R_2 = R_1$ . Já o segundo é onde aplica-se o ganho de fato. O esquemático desse circuito é dado por:



Para o primeiro, o ganho deve ser igual a -1, logo escolhe-se  $R2 = R1 = 10k\Omega$ , que é um valor comercial.

Já para o segundo, o ganho deve ser de -38,53. Escolhendo  $R4 = 62k\Omega$  tem-se que:

$$\frac{v_o}{v_i(\text{primeiro amp op})} = -38,53 = -\frac{R_4}{R_3} \Rightarrow R_3 = \frac{62k}{38,53} = 1,61k\Omega$$

Cujos valores comerciais mais próximos são de  $1,6k\Omega$ .

Desta forma:

$$v_o(t) = -\frac{62k}{1,6k} (\text{primeiro amp op}) = -38,53 * (-v_i(t))$$

$$v_o(t) = 38,53v_i(t)$$

Ou seja, para o circuito controlador proporcional, tem-se os seguintes componentes:

$$\begin{cases} R_1 = R_2 = 10k\Omega \\ R_3 = 1,6k\Omega \\ R_4 = 62k\Omega \end{cases}$$

Por fim, para o offset basta aplicar uma tensão de -4,906V em série com a onda resultante do filtro. Isso pode ser feito realizando um circuito divisor de tensão, alimentado por uma fonte de -10 V e composto por resistências de  $39 k\Omega$ ,  $10 k\Omega$  e  $51 k\Omega$  em série, onde a saída do sinal é dada depois da resistência de  $39 k\Omega$  e  $10 k\Omega$  e antes da de  $51 k\Omega$ . Com isso, a tensão de offset é igual a:

$$V_{offset} = \frac{-10(39k + 10k)}{(39k + 10k + 51k)} = -4,9V$$

Ou seja, para o circuito de offset, tem-se os seguintes componentes, com uma fonte de tensão de -10 V de alimentação:

$$R_1 = 10k\Omega$$

$$R_2 = 39k\Omega$$

$$R_3 = 51k\Omega$$

Os valores tabelados comerciais para os resistores e comerciais foram abstraídos das seguintes tabelas:

### Resistores Comerciais

1.0ohm	1.1ohm	1.2ohm	1.3ohm
1.5ohm	1.6ohm	1.8ohm	2.0ohm
2.2ohm	2.4ohm	2.7ohm	3.0ohm
3.3ohm	3.6ohm	3.9ohm	4.3ohm
4.7ohm	5.1ohm	5.6ohm	6.2ohm
6.8ohm	7.5ohm	8.2ohm	9.1ohm

Para obter os demais valores basta multiplicar por:  $10, 10^2, 10^3, 10^4, 10^5, 10^6$ ,

### Capacitores Comerciais

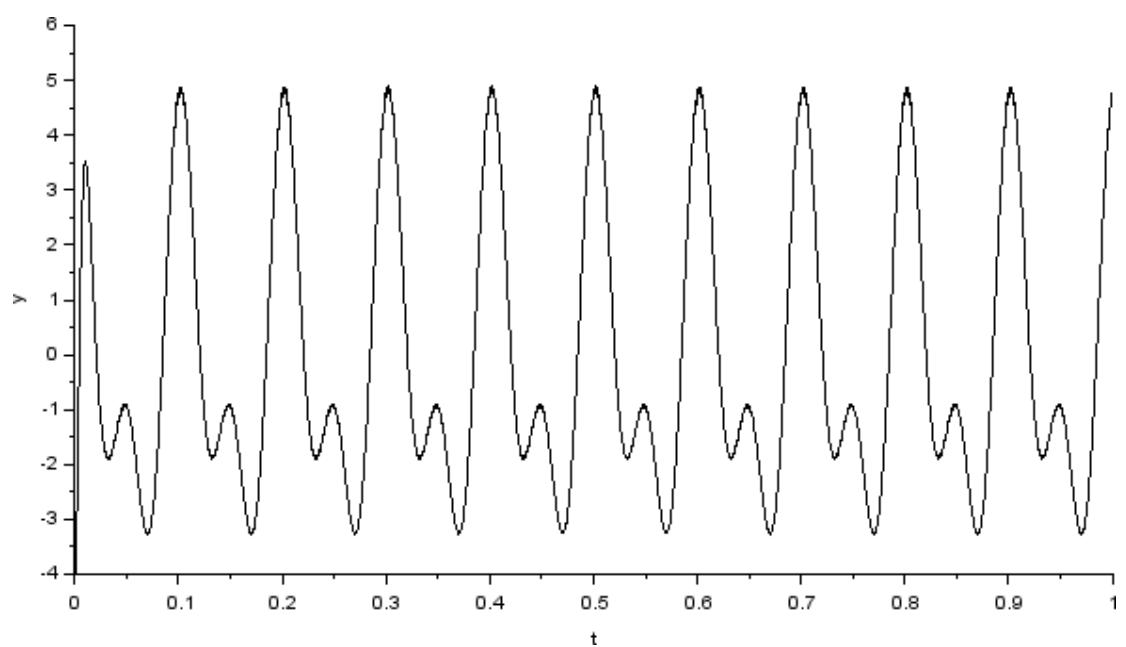
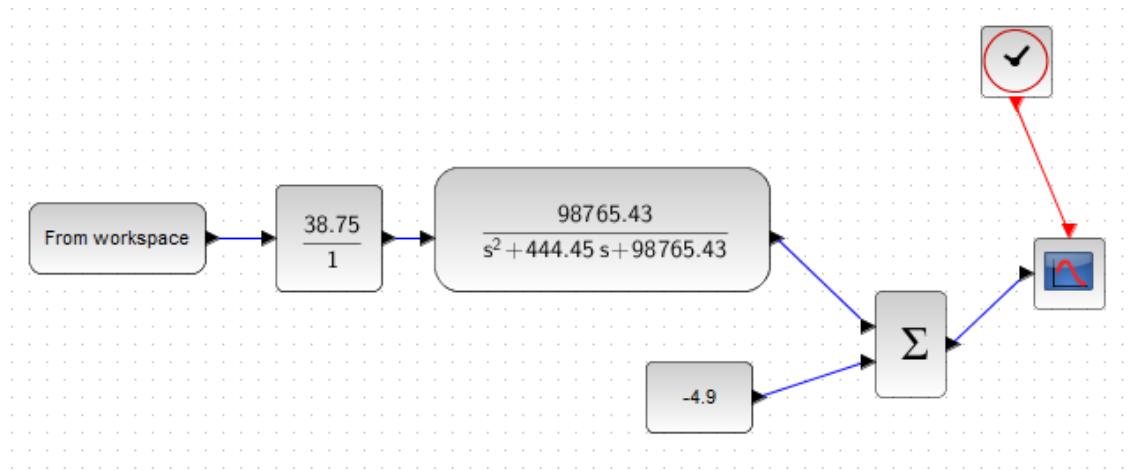
1.0F	1.1F	1.2F	1.3F
1.5F	1.6F	1.8F	2.0F
2.2F	2.4F	2.7F	3.0F
3.3F	3.6F	3.9F	4.3F
4.7F	5.1F	5.6F	6.2F
6.8F	7.5F	8.2F	9.1F

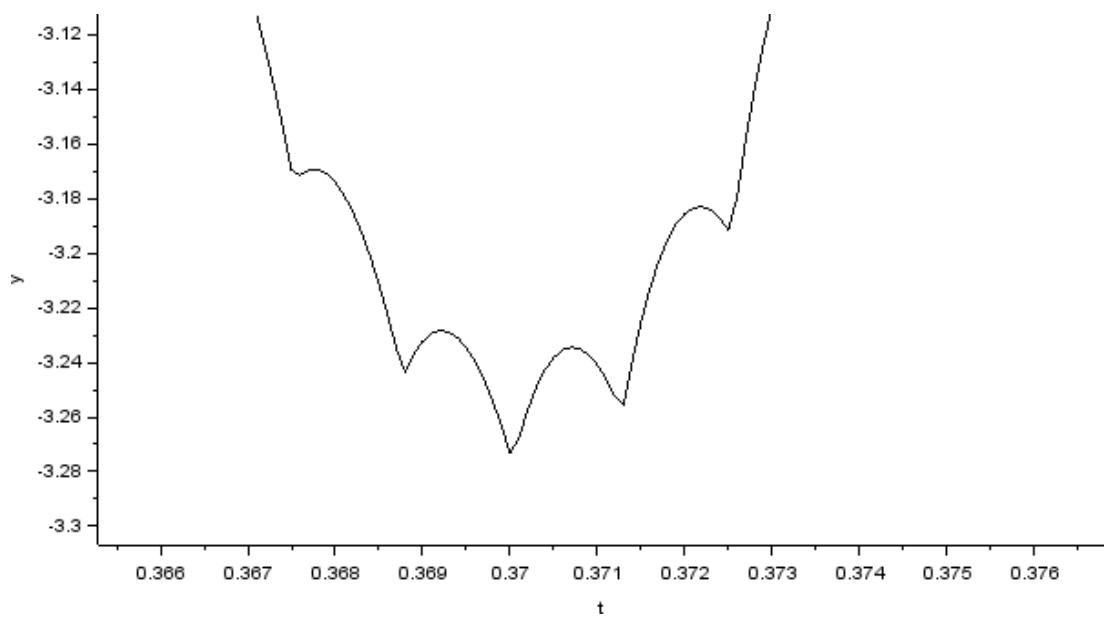
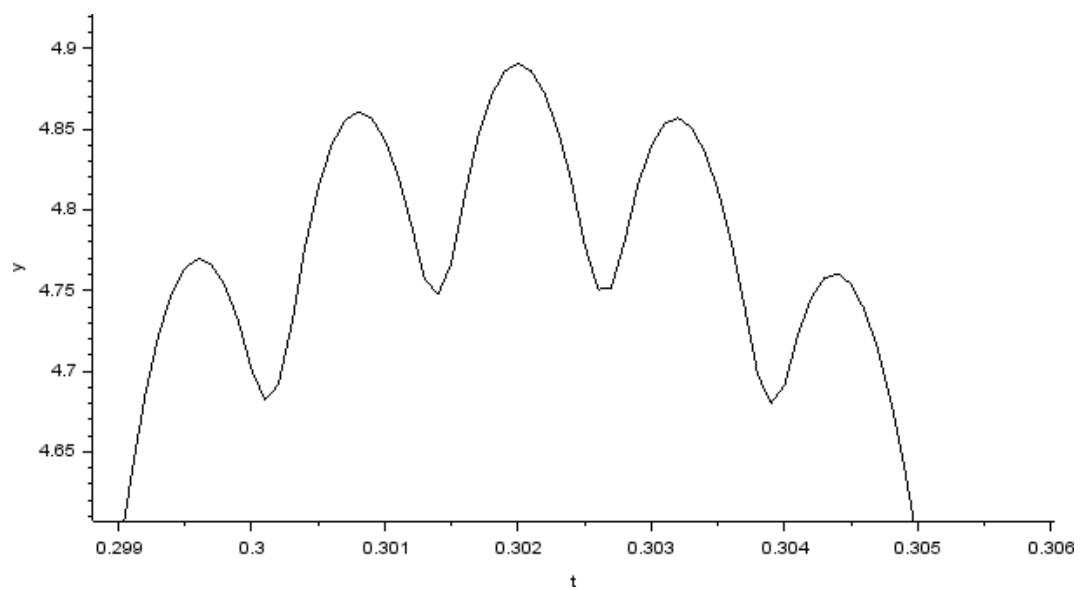
Para obter os demais valores multiplique pelos seus submultiplos: mili, micro, nano e pico.

Do site:

<http://www3.eletronica.org/dicas-e-hacks/valores-comerciais-de-resistores-capacitores-indutores-e-fusiveis>

Por fim, pode-se aplicar esses valores novos de acordo com os componentes escolhidos e verificar se a resposta continua a mesma através da simulação pelo xcos. Fazendo isso, tem-se que:



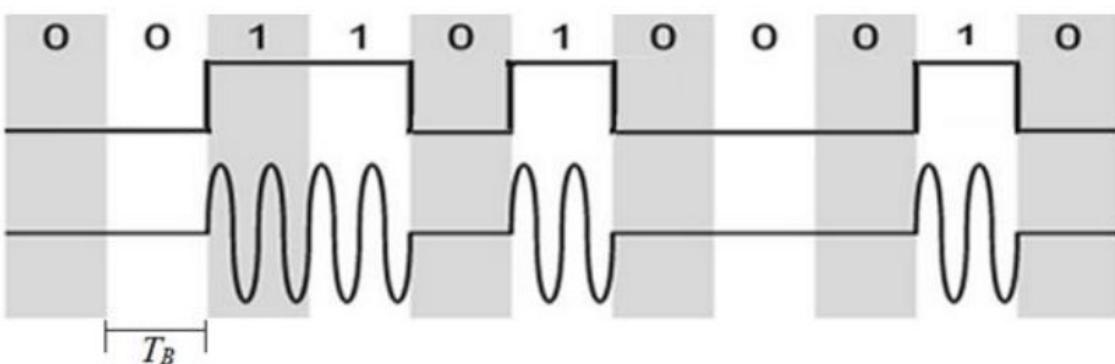


Onde percebe-se que o sinal recuperado é de fato muito próximo ao sinal original, com valores de máximo e mínimo de 4,9 e -3,27 respectivamente, além de manter o formato da onda.

Para a técnica de modulação BASK foram escolhidas duas amostras do sinal PCM criado a partir dos métodos já citados na etapa 01. No entanto, foi utilizada uma frequência de amostragem para o sinal PCM de 400[Hz] ao invés de 800[Hz], como feito originalmente. A razão da escolha de uma menor frequência para produzir o sinal PCM se deve ao fato da geração de menos amostras totais, além de aumentar o período de amostragem de cada bit e como será visto no decorrer deste relatório, com um valor menor de frequência de amostragem do sinal PCM menor será a banda de passagem do sinal modulado o que implica em uma frequência de amostragem menor para o sinal modulado de acordo com o teorema de amostragem passa faixa. Os motivos escolhidos são vantajosos pois acabam gerando uma economia computacional em relação ao tamanho das variáveis usadas.

Devido a este motivo que se escolhe fazer a modulação de apenas 2 amostras do sinal PCM e não todas as 400 que são geradas, já que o objetivo desta etapa é verificar o funcionamento da modulação BASK e caso quisesse modular todas as amostras provavelmente o software não aguentaria passar por todas as etapas.

Em relação a técnica de modulação digital BASK, ela terá o seguinte funcionamento: quando o bit for 1 a resposta modulada será:  $m(t) = 1 * \cos(2\pi f_c t)$ , onde  $f_c$  é a frequência da portadora, igual a 10[MHz], e quando o bit for 0 a resposta modulada será:  $m(t) = 0$ . A forma de onda modulada indica os bits transmitidos. Um exemplo desta modulação é mostrado abaixo:



Foram escolhidas duas amostras consecutivas do sinal PCM que apresentam uma variação de 0 para 1 bit a bit, que são justamente as amostras 84 e 85, visto a seguir:

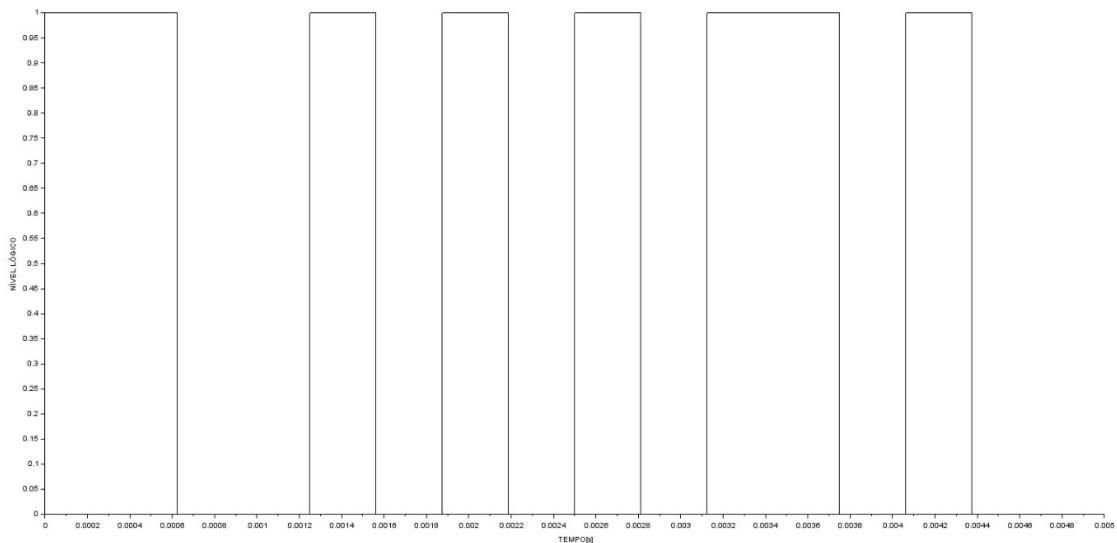
<b>84</b>	1	1	0	0	1	0	1	0	
85	1	0	1	1	0	1	0	0	

Construção da sequência no Scilab:

```

1 x = [1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0];
2 bp = 1/(400*8);
3 bit = [];
4
5 for n=1:1:length(x)
6     if x(n) == 1;
7         se = ones(1,1000);
8     else
9         se = zeros(1,1000);
10    end
11    bit = [bit se];
12 end
13 t1 = 0:bp/1000:(bp*length(x))-bp/1000;
14 plot2d(t1,bit), xlabel('TEMPO[s]'), ylabel('NÍVEL LÓGICO')

```



A sequência binária pode ser vista claramente. Cada bit vai possuir um período igual a:

$$Período_{bit} = \frac{1}{f_{bit}} = \frac{1}{Número_{amostras\ PCM} * Número_{bits}} = \frac{1}{400 * 8} = 0,0003125[\text{s}]$$

Existem 16 bits na sequência binária, logo o tempo de duração total será de 0,005[s].

Para realizar a obtenção do sinal PWM na modulação BASK, é necessário primeiramente fazer a amostragem deste sinal modulado. Para descobrir qual é o valor da frequência de amostragem, faz-se necessário saber o valor da banda ocupada pelo sinal modulado com a sequência binária utilizada, com isso será possível encontrar o valor da frequência de amostragem resultante.

Para determinar a banda ocupada na modulação BASK, faz-se necessário encontrar o valor do espectro de magnitude do sinal. Para realizar tal ação, é utilizada a lógica da modulação BASK,

feita para cada bit da sequência binária. Sendo assim, o resultado da concatenação da aplicação da modulação para cada bit será o sinal modulado total m.

Foi escolhida uma frequência de amostragem significativamente grande o suficiente para realizar a amostragem de maneira bem sucedida, levando em contra que a frequência da portadora é de 10[MHz]. Tal escolha se faz necessária pois se deseja visualizar o espectro de magnitude da forma mais limpa possível, com isso é evitado perdas de informações acarretadas por uma frequência de amostragem ruim. Aplicando o Teorema de Nyquist, é necessária escolher uma frequência de no mínimo 20[MHz]. Portanto, levando em conta que o período do bit é igual a 0,0003125; foi escolhida uma frequência de amostragem seguindo a seguinte fórmula:

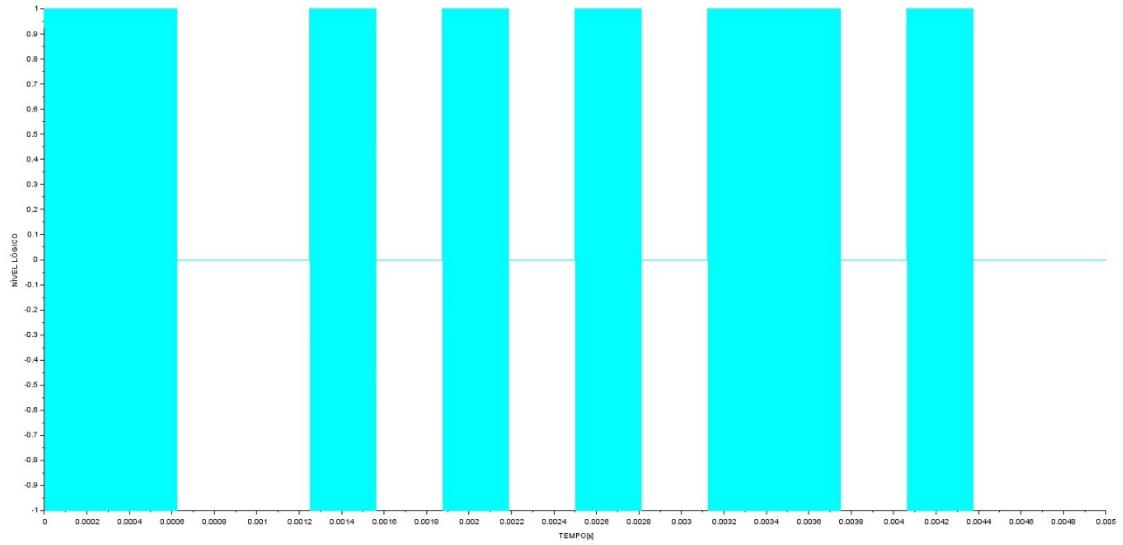
$$f_s = \frac{100000}{p_b} = \frac{100000}{0,0003125} = 320[\text{MHz}]$$

Com isso, pode-se notar claramente que se obteve um valor 32 vezes maior que o da portadora.

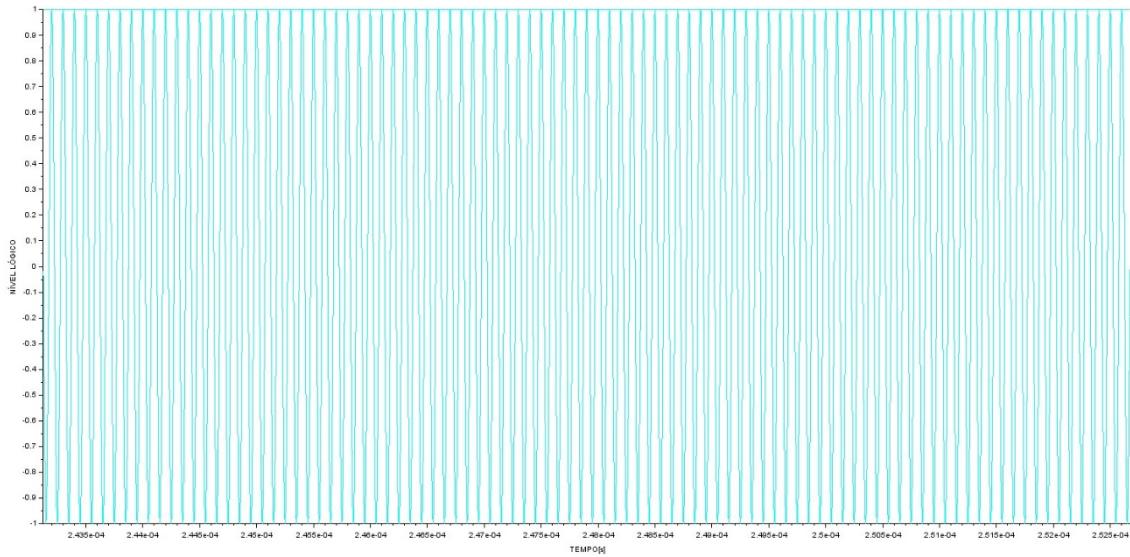
Vale ressaltar que esta não é a frequência de amostragem final, e sim uma frequência provisória para gerar uma boa visualização de como é o sinal modulado e o seu espectro de amplitude. A frequência de amostragem final será determinada aplicando-se o teorema de amostragem passa-faixa.

Agora já se faz possível visualizar o sinal modulado BASK:

```
18 t2 = 0:bp/100000:bp - bp/100000;
19 m = [];
20 for (i=1:1:length(x))
21   if(x(i)==1)
22     y=1*cos(2*pi*10e6*t2);
23   else
24     y=0;
25   end
26   m = [m y];
27 end
28 t3 = 0:bp/(100000):(bp*length(x))-bp/100000;
29 plot2d(t3,m);
```



Visualizando melhor a região onde o bit representado é 1:



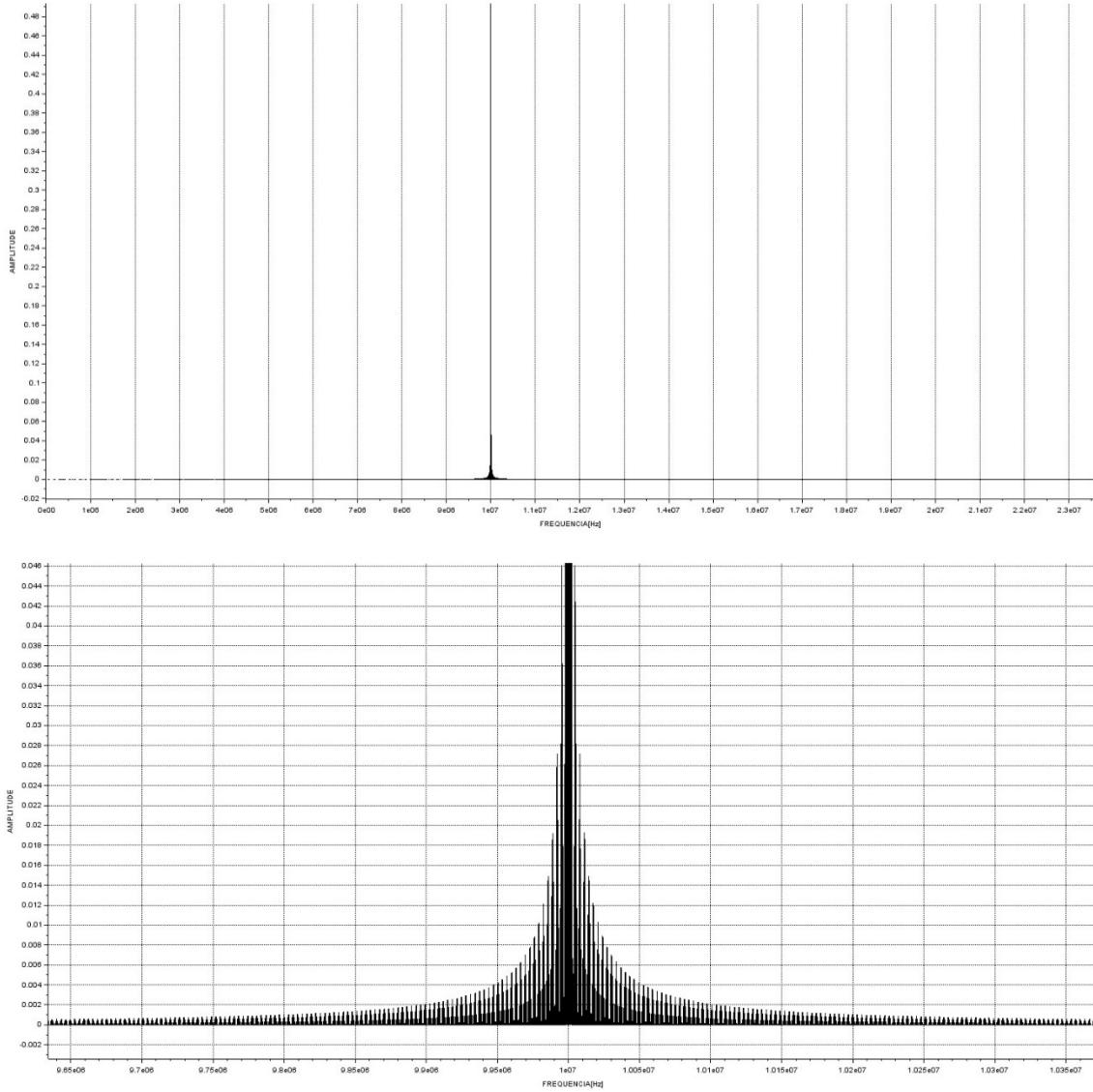
Nota-se claramente os cossenos dados por:  $m(t) = 1 * \cos(2\pi * 10 * 10^6 * t)$

O espectro de amplitude é obtido da seguinte forma: tendo as magnitudes do sinal, sendo elas em função das frequências, tem-se que o vetor de amplitude é dado pelo módulo da aplicação da fft sobre o sinal modulado m completo multiplicado por 2 e dividido pelo número de amostras do próprio sinal, no entanto, o vetor de frequência é construído com base nos múltiplos da frequência fundamental, que por sua vez é definida como sendo o inverso do período de observação do sinal, indo até o número de amostras do sinal -1. O resultado obtido fica:

```

31 N = length(m);
32 Amp = (2*abs(fft(m))/N);
33 f = 0:1/(bp*length(x)): (N-1)*1/(bp*length(x));
34 plot2d3(f,Amp), xlabel('FREQUENCIA[Hz]'), ylabel('AMPLITUDE')

```



Fazendo a análise do espectro de magnitude, nota-se de forma clara o espectro modulado do sinal, ou seja, a mínima frequência não parte de zero, mas fica centrada a frequência da portadora, que por sua vez, corresponde à 10[MHz], ocupando uma certa banda. Para determinar a frequência de amostragem final do sinal modulado, não é necessário aplicar o teorema de Nyquist, mas sim o teorema de amostragem passa faixa, onde com ele será permitida uma escolha de frequência de amostragem bem menor que a da outra alternativa, além de não gerar sobreposição no espectro e distorção na forma de onda.

Com o teorema da amostragem passa-faixa faz-se possível recuperar o sinal  $m(t)$ , levando em conta que este foi amostrado com uma frequência  $f_s$  dada por:

$$f_s = \frac{2 * f_{max}}{k}$$

$k$  é dado pelo valor arredondado para baixo, seguindo a divisão abaixo:

$$k = \frac{f_{max}}{B}$$

Fazendo a análise da última figura apresentada, pode-se perceber que a frequência máxima aproximada pela qual se há uma amplitude não desprezível do sinal é a de 10,1[MHz]. Partindo

da mesma lógica, a banda ocupada pelo sinal, contendo as amplitudes não desprezíveis, pode ser definida começando a partir de 9,9[MHz] e terminando em 10,1[MHz]. Com isso, é obtido um resultado de  $(10,1[\text{MHz}] - 9,9[\text{MHz}]) = 200[\text{kHz}]$ .

É importante estipular um valor de banda ocupada igual a 20 vezes a banda que o sinal modulado de fato ocupa, com o objetivo de acomodar com sucesso a transição do filtro e também reduzir a distorção gerada no PWM. Tem-se  $k$  igual a:

$$k = \frac{f_{max}}{20 * B_{real}} = \frac{10,1 * 10^6}{20 * 200 * 10^3} = 2,525$$

Como  $k$  é dado como sendo o valor arredondado para baixo de seu valor original, então  $k = 2$ . A frequência de amostragem final é:

$$f_s = \frac{2 * f_{max}}{k} = \frac{2 * 10,1 * 10^6}{2} = 10,1[\text{MHz}]$$

Optou-se por arredonda a frequência de amostragem calculada para 10[MHz], ficando assim idêntica a frequência da portadora, assim desta forma, durante a construção do novo sinal modulado BASK, se terá uma amostra por período da portadora, desta forma é evitada a existência de números quebrados para o sinal resultante, que por sua vez acarreta em um vetor das amostras do sinal modulado com dimensão diferente do vetor de tempo da sequência binária usando o novo período de amostragem. A resolução de tal problema se daria completando o valor do sinal modulado com zeros, obtendo assim uma dimensão igual à do vetor de tempo, com isso faz-se possível obter o sinal PWM. Devido aos fatos citados, escolheu-se arredonda a frequência de amostragem final para 10[MHz].

Percebe-se o quanto foi reduzida a frequência de amostragem necessária para uma recuperação bem sucedida do sinal modulado utilizando o teorema da amostragem passa-faixa, se fosse utilizado o teorema de Nyquist, visando obter o mínimo de perda de informação possível, seria necessária uma frequência em torno de 200[MHz].

Fazendo novamente a construção do sinal modulado BASK, mas fazendo uso da nova frequência de amostragem. O sinal PWM será obtido com base nesse novo sinal modulado.

```

37 x = [1 1 0 0 1 0 1 0 10 1 1 0 1 0 0];
38 fs = 1000000;
39 bp = 1/(400*8);
40 ts = 1/fs;
41 t2 = 0:ts:bp-ts;
42 m= [];
43 for (i=1:1:length(x))
44     if (x(i)==1)
45         y=1*cos(2*pi*10e6*t2)
46     else
47         y=0;
48     end
49     m = [m y];
50 end

```

Com o novo sinal modulado BASK m já determinado, faz-se necessário normalizá-lo, ou seja, foram utilizados 8 bits de quantização para determinar o vetor de amostras de PCM, assim o maior valor que uma amostra pode assumir é de 255. No sinal BASK o valor máximo assumido pelo coseno é de 1, pode-se concluir que será necessário multiplicar por 255.

$$m = m * 255;$$

Aprofundando-se na construção do sinal modulado BASK, nota-se de forma clara que para cada bit, tem-se um número de ciclo de portadora igual a:

$$p_b * f_s = \frac{1}{(400 * 8)} * 10 * 10^6 = 3125$$

Como existem 16 bits na sequência binária o número total de amostras será de 50000 amostras. O valor de cada amostra depois da normalização é 255 ou 0,255. Vale ressaltar que não existem outros valores intermediários do coseno, uma vez que a frequência de amostragem é igual a frequência da portadora. De forma geral, este fato pode parecer um problema que comprometeria a onda modulada BASK m e que geraria uma demodulação falha, mas isto não ocorre.

Já que a onda modulada BASK já foi normalizada e definida, faz-se possível obter o sinal PWM. Este processo será feito na FPGA através do software Quartus, onde os valores das amostras do sinal modulado devem estar na forma binária. A conversão é feita utilizando o comando dec2bin().

O PWM é gerado fazendo a comparação do valor absoluto de cada amostra da onda modulada com uma onda dente de serra. Sendo assim, enquanto a onda dente de serra for menor que o valor da atmosfera da onda modula, a saída terá nível lógico alto, e quando não for menor, a saída terá nível lógico baixo, produzindo assim uma modulação em largura de pulso.

O PWM gerado na FPGA se dará realizando a comparação entre amostra e o valor em contador, que por sua vez, será incrementado por um clock, mas levando em conta que o contador resetará para zero quando ultrapassar o valor máximo, com isso, é visto que se deve atualizar a amostra utilizando a próxima a seguir.

A definição do clock se dará levando em conta o valor da frequência de amostragem, que corresponde à 10[MHz].

$$f_s = \frac{f_{clk}}{c_{max} + 1}$$

$c_{max}$  é o máximo valor do contador, cujo valor deve respeitar a seguinte inequação:

$$c_{max} = 2^{N+1}$$

Onde N é o número de bits de quantização utilizado. Esta inequação garante que o sinal PWM gerado não possua Duty Cicle superior a 50% da frequência de amostragem. Como foram utilizados 8 bits para quantização, logo:

$$c_{max} > 2^9 \Rightarrow c_{max} > 512$$

Para obter um valor de clock redondo, foi escolhido  $c_{max} = 549$ , satisfazendo assim a inequação. Assim, tem-se:

$$f_s = \frac{f_{clk}}{c_{max} + 1} \Rightarrow f_{clk} = f_s(c_{max} + 1) = 10 * 10^6 * (549 + 1) \Rightarrow f_{clk} = 5500[MHz]$$

Com todos os parâmetros já definidos, foi feita uma lógica em Verilog para gerar o circuito que criará o sinal PWM. Foi programado um módulo para o contador, amostragem e comparador.

Módulo das amostras:

```
module Amostras(
    input clk,
    input [15:0] A,
    output reg [11:0] amostra);
    reg [7:0] mem [0:49999];
initial begin
    $readmem("D:/Google_Drive/faculdade/2021.2/PBLE04/FPGA/C.txt",mem);
end
always @(A) begin
    amostra = mem[A];
end
endmodule
```

É criado um vetor 50000\*8 para então armazenar os valores e preencher cada uma dessas posições com as amostras do sinal modula BASK normalizado através de um arquivo txt. Vale ressaltar que a saída do módulo via ser controlado pelo valor de entrada A.

O módulo do contador é dado por:

```
module Contador(
    input clk,
    output reg[11:0] cont,
    output reg[15:0] A);
    initial cont = 12'd0;
    initial A = 16'd0;

    always @ (posedge clk) begin
        cont <= cont + 12'd1;
        if(cont >= 12'd549) begin
            cont <= 12'd0;
            A <= A + 16'd1;
            if(A == 16'd499999) begin
                A <= 16'd0;
            end
        end
    end
endmodule
```

O clock possui valor máximo de 549 e é incrementado de 1 em 1, assim que ultrapassado este valor, ele volta a ser zero, recomeçando todo o processo. O contador é utilizado também para servir de indicação de posição de memória, ou seja, qual amostra deve ser comparada para gerar o PWM. Quando o contador se reinicia, deve-se atualizar a amostra fazendo uso da próxima.

Isto é feito realizando a incrementação de 1 na variável de controle da posição da amostra toda vez que o valor do contador é extrapolado. Como existem 50000 amostras, o processo é realizado até chegar a este valor, fazendo com que o valor da posição da amostra volte a 0, reiniciando assim o ciclo.

O módulo de comparação é:

```
module comparador(
    input [11:0] amostra,
    input [11:0] cont,
    output reg pwm);

    always @* begin
        if(amostra > cont) pwm = 1;
        else pwm = 0;
    end
endmodule
```

Este módulo apenas realiza a comparação do valor das entradas referentes à amostra atual e do contador, e se o valor absoluto da amostra é maior do que o contador a saída PWM é igual a 1, se não a saída PWM é 0.

Realizando a conexão entre os 3 módulos, tem-se:

```

module pwm(
    input clk,
    output saída);

    (*keep=1*) wire[11:0] cont;
    (*keep=1*) wire[11:0] amostra;
    (*keep=1*) wire[15:0] A;

    Contador C (
        .clk(clk),
        .cont(cont),
        .A(A));

    Amostras Amos(
        .clk(clk),
        .A(A),
        .amostra(amostra));
    | 
    comparador comp(
        .amostra(amostra),
        .cont(cont),
        .pwm(saída));
    | 
endmodule

```

O módulo é conectado da seguinte forma: a saída do módulo contador A vai para a entrada do módulo amostras para indicar qual posição da memória e consequentemente qual amostra deverá ser comparada. Então essa amostra e a saída cont do módulo contador vão para a entrada do módulo comparador onde são comparadas a fim de gerar o sinal PWM.

É implementado um clock com período de 0,182[ns], o que gera uma frequência aproxima de 5500[MHz]:

```

module pwm_TB;
reg clock;
reg [11:0] cont;
reg [11:0] amostra;
reg [15:0] A;
wire saida;

pwm DUT(
    .clock(clock),
    .saida(saida) );

initial begin
    clock = 0;
end

always begin
    #0.091 clock =~clock;
end

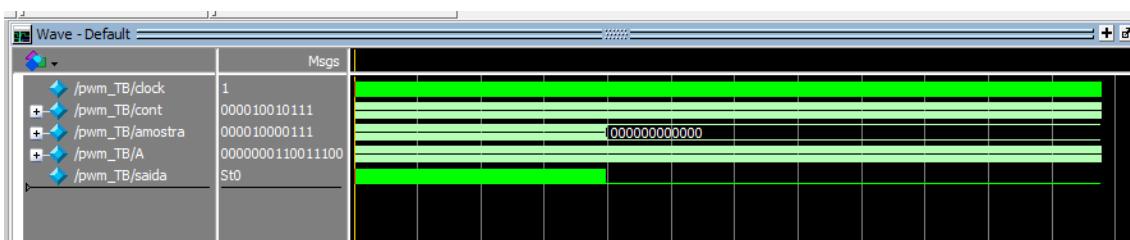
initial begin
$init_signal_spy("/pwm_TB/DUT/cont","cont",1);
$init_signal_spy("/pwm_TB/DUT/amostra","amostra",1);
$init_signal_spy("/pwm_TB/DUT/A","A",1);
end

initial
    #550000 $stop;

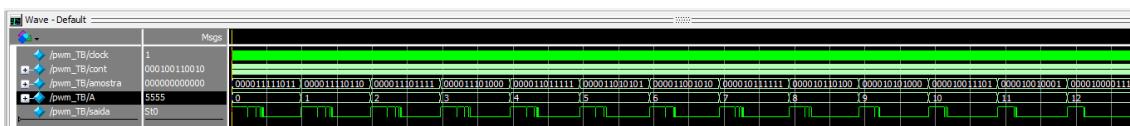
endmodule

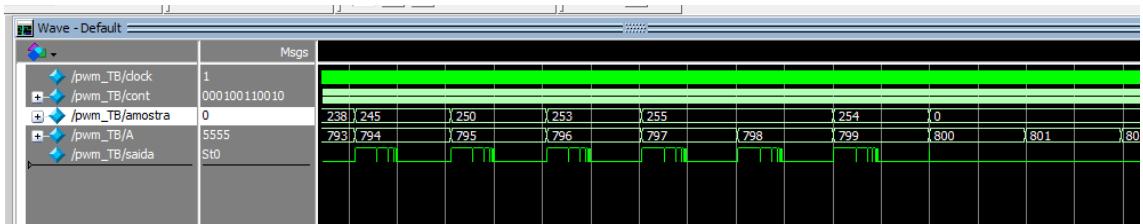
```

Tem-se como resultado de simulação:



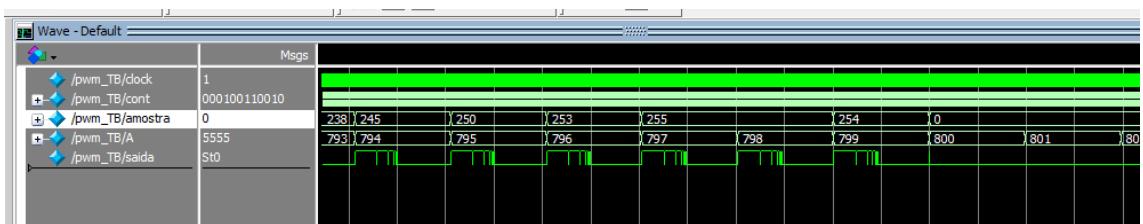
É visualizada as 12 primeiras amostras do sinal modulado BASK onde era representado o bit 1, e em seguida as primeiras amostras do primeiro bit 0 da amostra PCM escolhida, nota-se que:



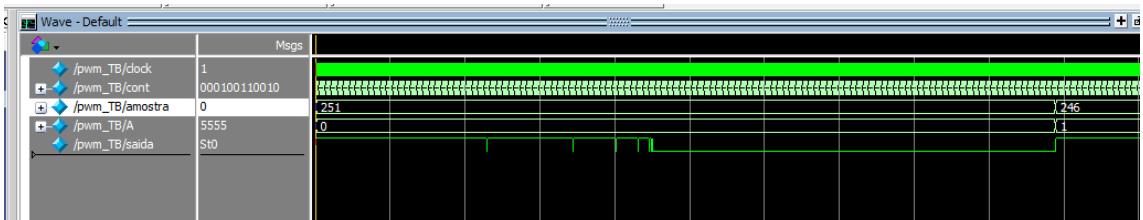


Pode-se ver então que quando a amostra é igual a 0, o sinal PWM resultante também é igual a 0 e quando a amostra é igual a 255, o sinal PWM terá um período em estado alto até o contador passar esse valor e então volta para o estado baixo.

O funcionamento deste módulo se dá analisando o comportamento da onda de PWM em uma única amostra através da comparação entre a proporção do tempo em que a saída fica em 1 e o tempo total do período da onda PWM para essa amostra em específico e a proporção do valor absoluto da amostra com o valor máximo do contador igual a 549. Teoricamente, eles deveriam ser iguais ou muito próximos. Realizando a escolha da primeira amostra para fazer tal teste, é visto:



O sinal PWM muda de nível lógico em 45806[ps]. O contador extrapola no seguinte instante:



O contador extrapola em 98910[ps]. O valor ideal seria em 100000[ps], correspondendo assim ao período de amostragem. No entanto, foi realizada a simulação em Gate Level, levou-se em conta desta forma os atrasos inerentes aos componentes utilizados para realizar a simulação além também da própria resolução utilizada.

As proporções ficam da seguinte forma:

$$\frac{\text{Valor da amostra}}{\text{Valor máximo do contador}} = \frac{255}{549} = 0,46448$$

$$\frac{\text{Tempo em nível lógico alto}}{\text{Período da onda para a amostra}} = \frac{45806[\text{ps}]}{98910[\text{ps}]} = 0,46311$$

A eficácia do módulo foi comprovada, pois a diferença entre as proporções foi mínima.

Agora realizando a implementação no Scilab. O vetor PWM de saída terá no total:

$$(c_{\max} + 1) * \text{Número de amostras} = 550 * 50000 = 27500000[\text{amostras}]$$

Onde cada amostra terá uma comparação com um valor do contador que vai de 0 até 549.

É feito o algoritmo para realizar a comparação entre o valor de cada uma das amostras da onda modulada BASK e o contador. É utilizado um FOR de 1 a 50000 para varrer cada uma das amostras e outro for de 1 a 550 para varrer todos os valores possíveis do contador.

No Scilab o contador fez o incremento de 1 até 550, equivalente ao que seria o incremento de 0 até 549 no Quartus, uma vez que o Scilab possui índice 1 para indicar a primeira posição do vetor, já no Quartus, é representado por zero.

Foi feita a comparação entre o valor do contador e o valor da amostra no índice indicado pelo FOR através da função IF. Falta ainda concatenar cada operação de comparação com os valores abstraídos do PWM, afinal, quando extrapolado o valor do contador passa-se para a seguinte posição do vetor de amostras e assim começa um novo ciclo de obtenção dos valores do PWM. Para realizar a solução de tal problema, utiliza-se de uma variável auxiliar a fim de corrigir a posição do PWM, ou seja, conforme o índice da posição do vetor de amostras aumenta em 1, o índice de posição do vetor de PWM deverá ser incrementado em 550, que corresponde justamente ao valor de extração do contador. Assim, tem-se:

```
56 for i=1:50000
57     for cont = 1:550
58         j = i-1;
59         aux = cont + 550*j;
60         teste = aux;
61         if(cont<m(i))
62             PWM(aux)=1;
63         else
64             PWM(aux)=0;
65         end
66     end
67 end
```

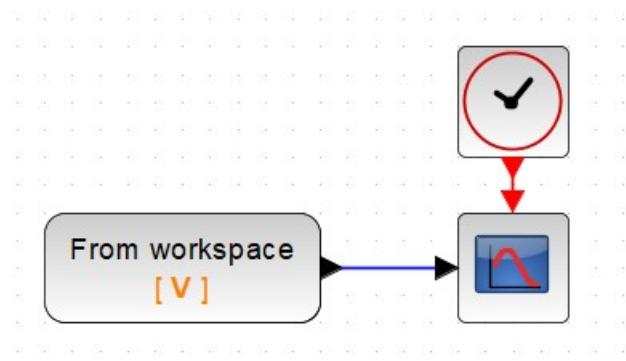
É plotado um gráfico de t em função do valor do vetor PWM, onde t é o vetor temporal que vai de 0 a 0,005[s], sendo incrementado pelo período do clock igual ao inverso de 5500 MHz. Sendo assim:

```
69 fclk = fs*550;
70 tclk = 1/fclk;
71 t = 0:tclk:(bp*length(x)-tclk);
```

A plotagem do PWM em função do tempo foi feita pelo Xcos, onde os valores do console foram passados a ele através do comando struct().

```
73 V =struct('time','t','values','PWM');
```

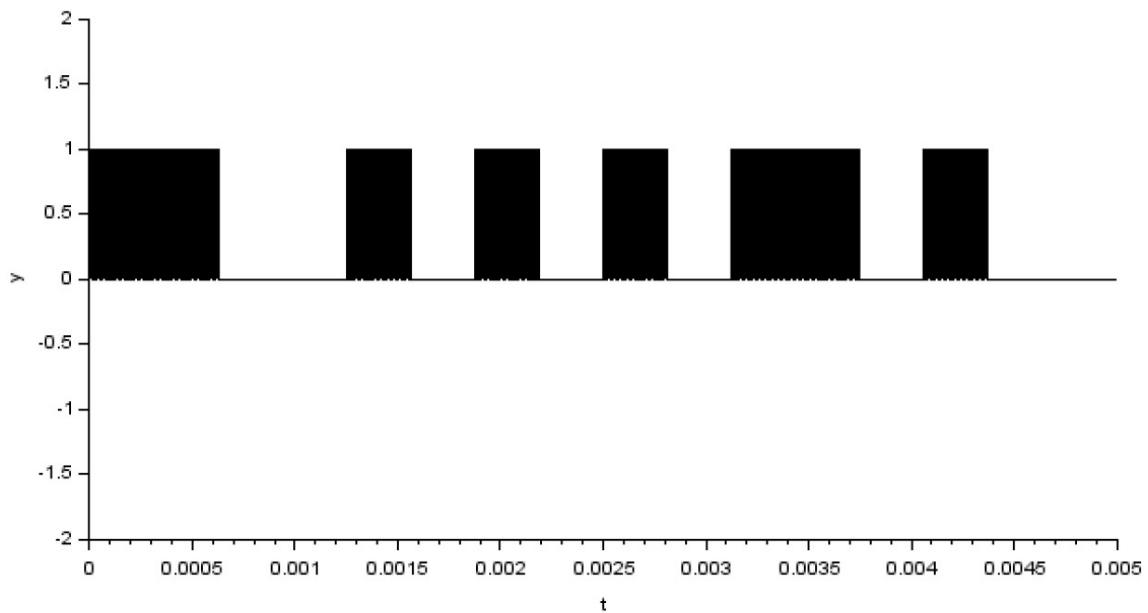
Obteve-se o seguinte diagrama de blocos:



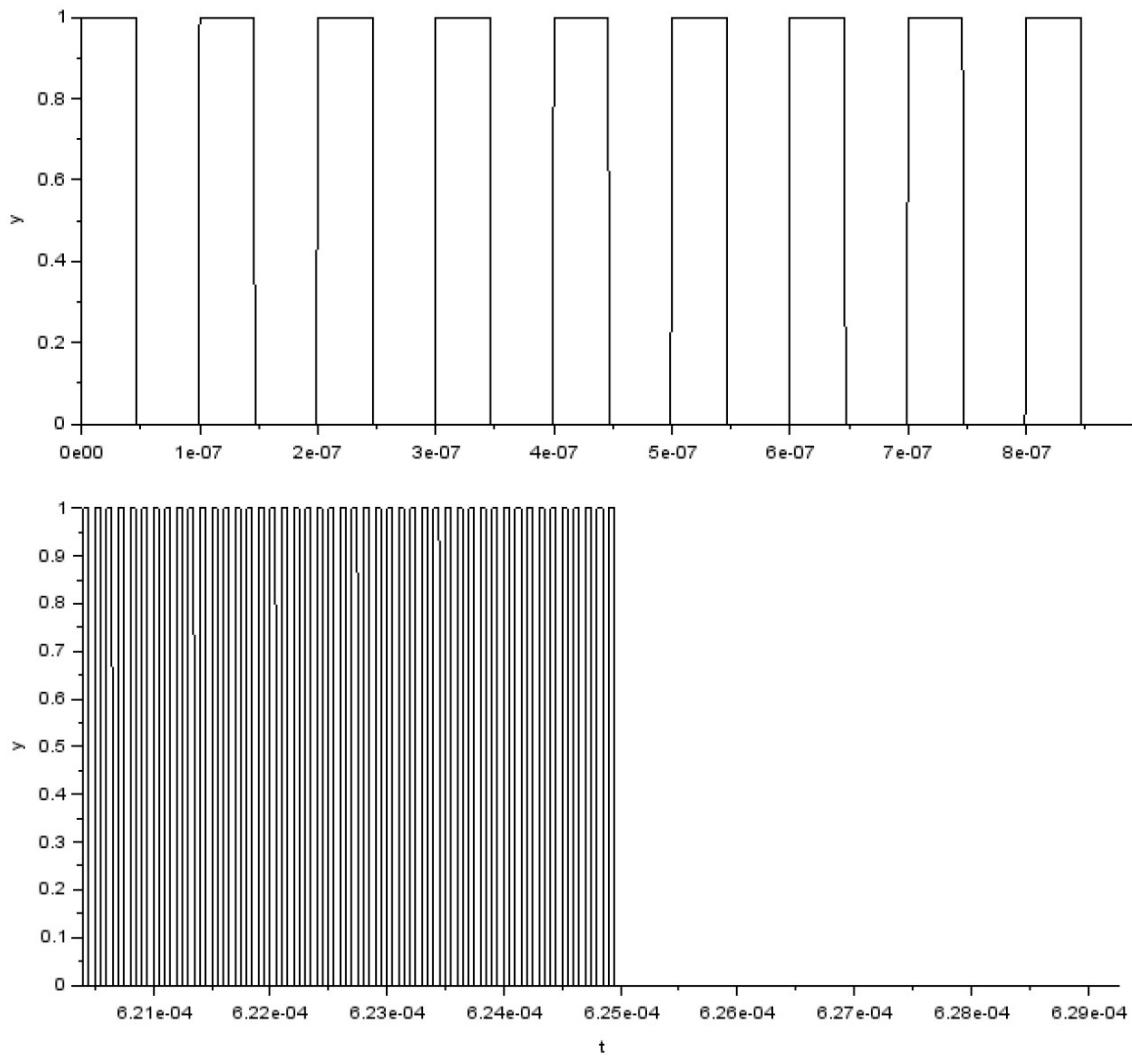
O período de clock foi determinado sendo igual a  $1 * 10^{-10}$ , valor este inferior ao período do PWM, sendo igual a:

$$\frac{1}{f_{clk}} = \frac{1}{10 * 10^6 * 550} = 1,82 * 10^{-10}$$

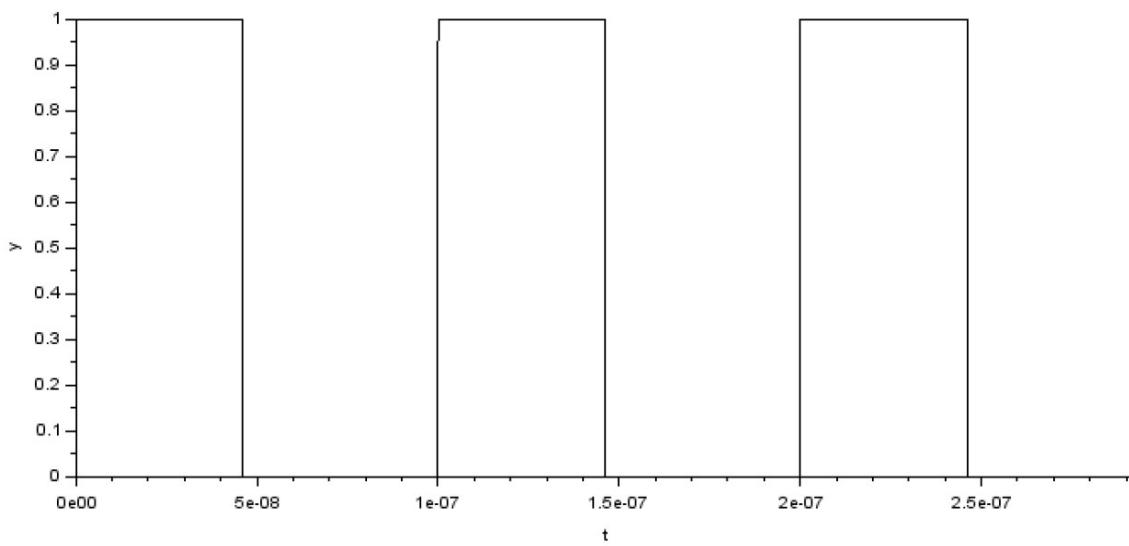
Assim, é obtida uma representação fiel da onda resultante, cujo resultado da simulação no osciloscópio, com tempo total de 0,005[s], é mostrado logo abaixo:

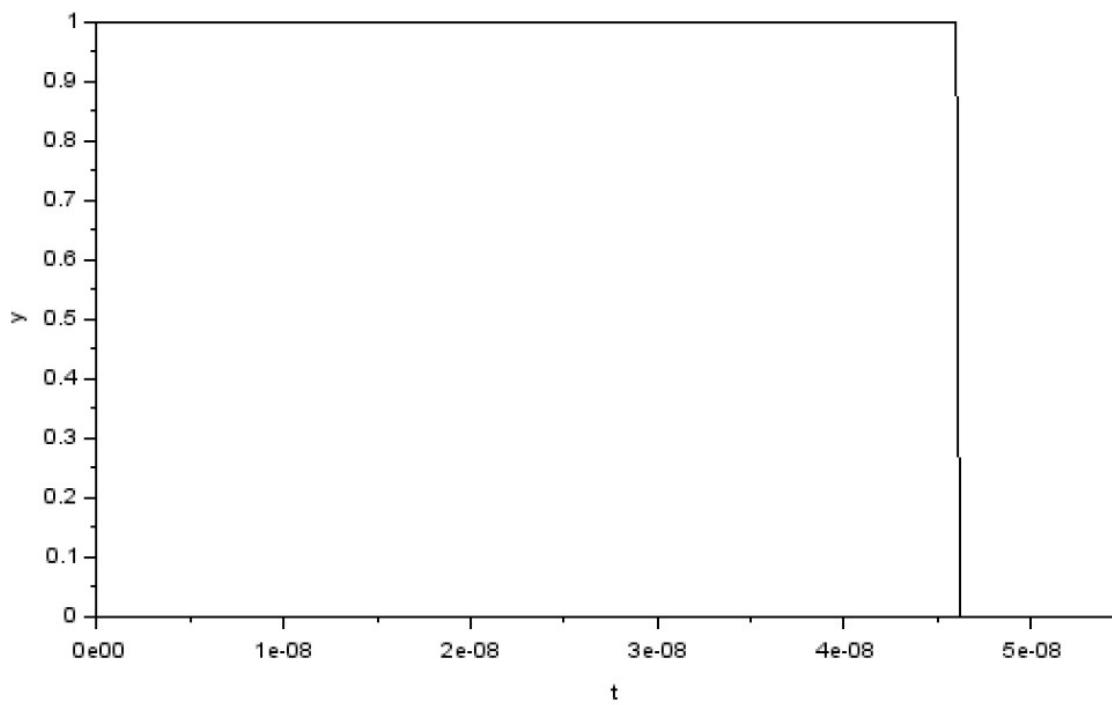


Pode-se ver o comportamento geral da onda PWM, em que há a variação constante de estado lógico alto e baixo quando o bit é 1 e completamento nulo quando o bit é 0.



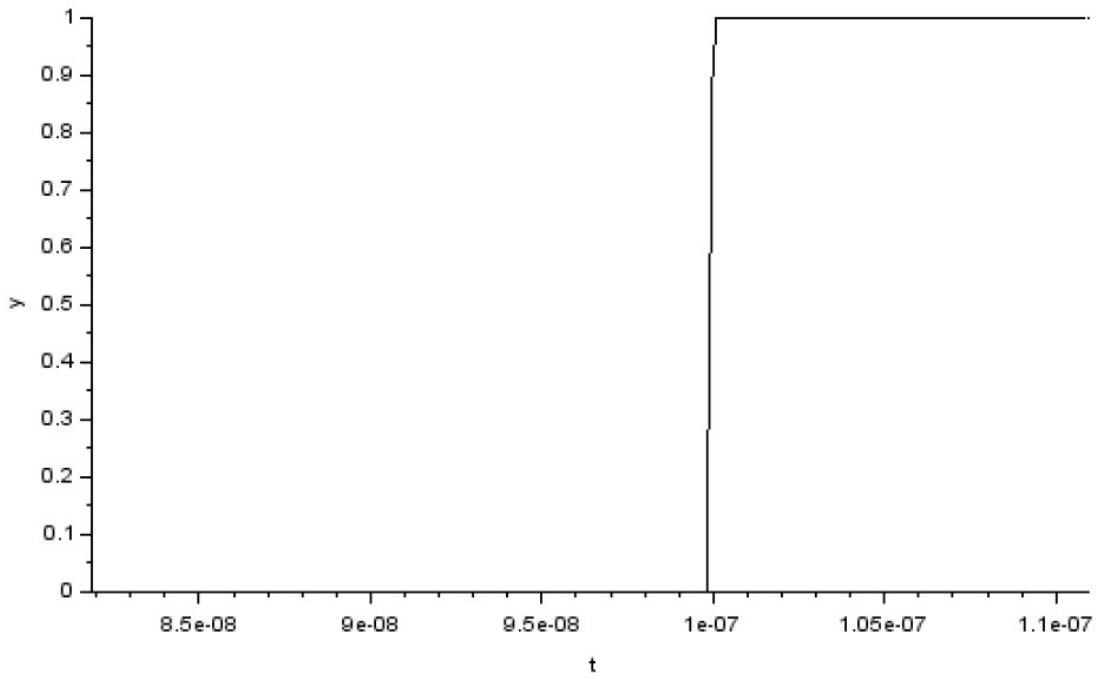
Aplicando novamente a comparação entre as proporções de tempo e valores já explicada anteriormente. Escolhendo mais uma vez como exemplo a primeira amostra, tem-se que o momento em que é mudado seu nível lógico alto para baixo no PWM, sendo este:





O instante corresponde à  $4,62 \times 10^{-8} [s]$ .

O instante de extração do contador é:



O instante é de aproximadamente  $9,98 \times 10^{-8} [s]$ .

Assim, fazendo os cálculos das proporções, tem-se:

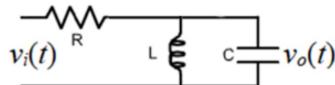
$$\frac{\text{Valor da amostra}}{\text{Valor máximo do contador}} = \frac{255}{550} = 0,46364$$

$$\frac{\text{Tempo em nível lógico alto}}{\text{Período da onda para a amostra}} = \frac{4,62 * 10^{-8}}{9,98 * 10^{-8}} = 0,46293$$

Houve uma diferença simbólica entre os valores das proporções, comprovando assim a eficácia do algoritmo. Vale ressaltar a implementação do sinal PWM feita no Quartus e no Scilab, foram praticamente idênticas.

Agora com o PWM construído, onde este foi obtido a partir do sinal modulado BASK, faz-se necessário filtrar este sinal, e o responsável por tal ação será um filtro passa-faixa, tendo como objetivo principal a conversão das amostras do sinal PWM para um sinal analógico, sendo este correspondente ao sinal modulado BASK original, com isso ao aplicar o filtro passa-baixa sobre o sinal PWM retorna-se ao sinal modulado. Tal recuperação do sinal analógico se baseia no teorema da amostragem passa-faixa, que foi utilizada para determinar a frequência de amostragem do sinal modulado, e devido a este fato, faz-se o uso do filtro passa-faixa para recuperação do sinal. Vale ressaltar que tal ação foi feita pois estava se tratando um sinal modulado, que por sua vez possui uma banda de passagem, não partindo de zero, mas sim centrado na frequência da portadora.

Logo abaixo é visto o circuito do filtro passa-faixa:



A função de transferência é dada por:

$$H(j\omega) = \frac{\frac{(j\omega)}{RC}}{(j\omega)^2 + \frac{1}{RC}(j\omega) + \frac{1}{LC}}$$

Onde  $\frac{1}{RC}$  corresponde à banda de passagem do filtro passa-faixa,  $\Delta\omega$  e  $\frac{1}{LC}$  diz respeito à frequência central ao quadrado, isto é,  $\omega^2$ .

Os 2 parâmetros foram determinados na etapa anterior, no momento em que se calculou a frequência de amostragem para o sinal modulado, sendo utilizado o teorema de amostragem passa-faixa, sendo feita através da análise do espectro de magnitude do sinal modulado. Os valores encontrados foram: banda de passagem de 200[kHz], frequência central igual a 10[MHz].

Logo, tem-se:

$$\frac{1}{RC} = B = 2\pi * 200 * 10^3$$

$$\frac{1}{LC} = \omega^2 = (2\pi * 10 * 10^6)^2$$

Houve a conversão dos parâmetros de Hz para rad/s. Faz-se necessário determinar uma componente e então calcular os valores das demais.

Escolheu-se C igual a 2,2[nF], então:

$$\frac{1}{RC} = 2\pi * 200 * 10^3 \Rightarrow R = \frac{1}{2\pi * 200 * 10^3 * 2,2 * 10^{-9}} = 361,7[\Omega]$$

O valor comercial escolhido do resistor foi de  $R = 360[\Omega]$ .

$$\frac{1}{LC} = (2\pi * 10 * 10^6)^2 \Rightarrow L = \frac{1}{(2\pi * 10 * 10^6)^2 * 2,2 * 10^{-9}} = 115,14[nH]$$

O valor comercial do indutor escolhido foi de  $L = 120[nH]$ .

Os valores comerciais foram extraídos das seguintes tabelas:

### Resistores Comerciais

1.0ohm	1.1ohm	1.2ohm	1.3ohm
1.5ohm	1.6ohm	1.8ohm	2.0ohm
2.2ohm	2.4ohm	2.7ohm	3.0ohm
3.3ohm	3.6ohm	3.9ohm	4.3ohm
4.7ohm	5.1ohm	5.6ohm	6.2ohm
6.8ohm	7.5ohm	8.2ohm	9.1ohm

### Capacitores Comerciais

1.0F	1.1F	1.2F	1.3F
1.5F	1.6F	1.8F	2.0F
2.2F	2.4F	2.7F	3.0F
3.3F	3.6F	3.9F	4.3F
4.7F	5.1F	5.6F	6.2F
6.8F	7.5F	8.2F	9.1F

### Indutores Comerciais

1.0H	1.1H	1.2H	1.3H
1.5H	1.6H	1.8H	2.0H
2.2H	2.4H	2.7H	3.0H
3.3H	3.6H	3.9H	4.3H
4.7H	5.1H	5.6H	6.2H
6.8H	7.5H	8.2H	9.1H

Fonte: <http://www3.eletronica.org/dicas-e-hacks/valores-comerciais-de-resistores-capacitores-indutores-e-fusiveis>

Desta forma, tem-se:

$$\frac{1}{RC} = \frac{1}{360 * 2,2 * 10^{-9}} = 1262626,26$$

$$\frac{1}{LC} = \frac{1}{120 * 10^{-9} * 2,2 * 10^{-9}} = 3787878787878787,89$$

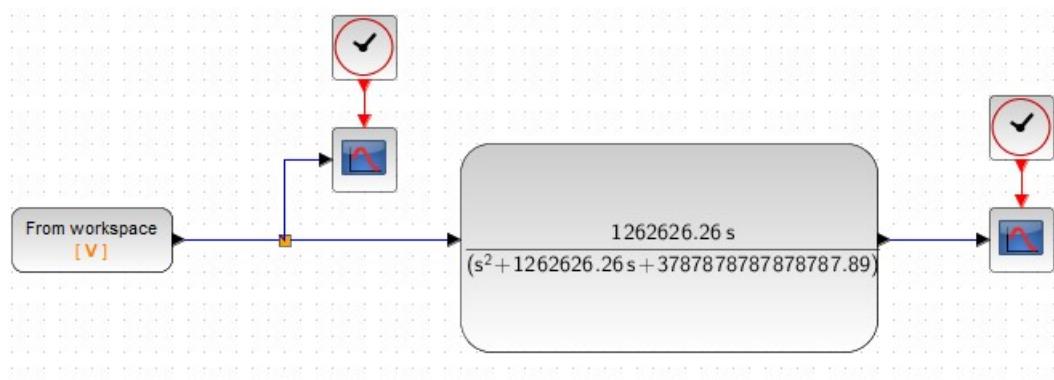
A função do filtro passa-faixa ficará:

$$H(s) = \frac{\frac{1}{RC} * s}{s^2 + \frac{1}{RC} * s + \frac{1}{LC}} = \frac{1262626,26 * s}{s^2 + 1262626,26 * s + 3787878787878787,89}$$

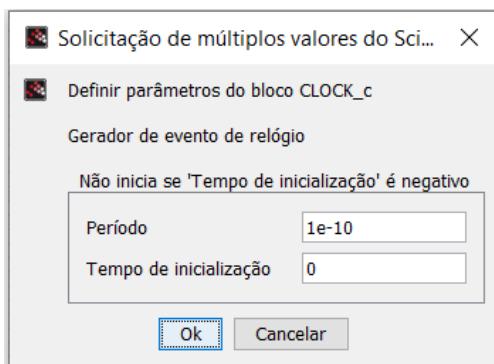
Com a função já projetada, é possível realizar a simulação em diagrama de blocos no Xcos das amostras PWM e do tempo das amostras utilizando o comando struct:

```
V = struct('time',t,'values',PWM');
```

Diagrama:



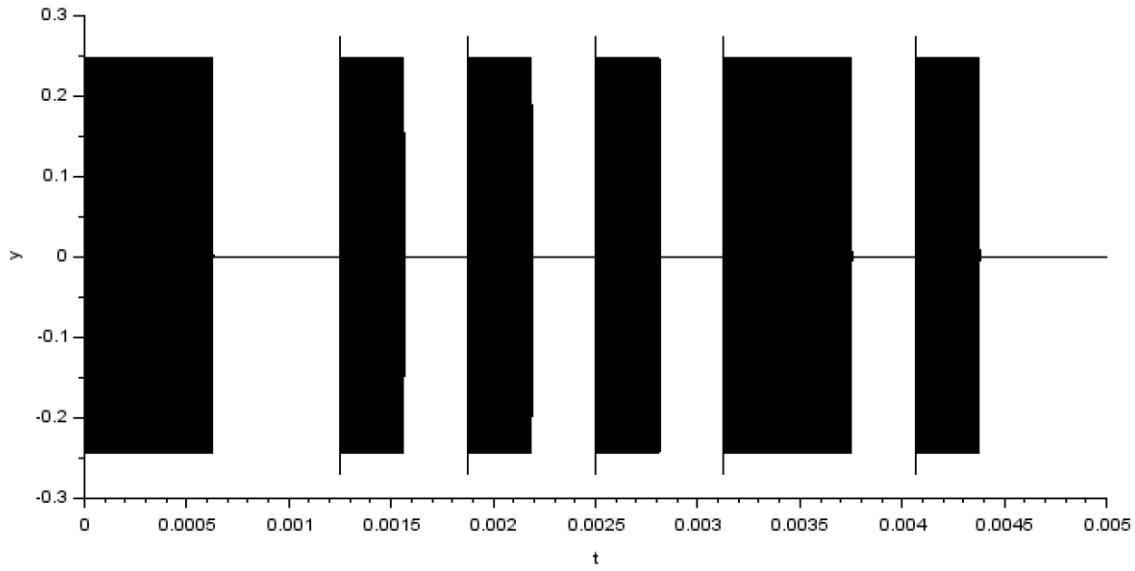
Onde o período do clock foi determinado como mostra a figura abaixo:



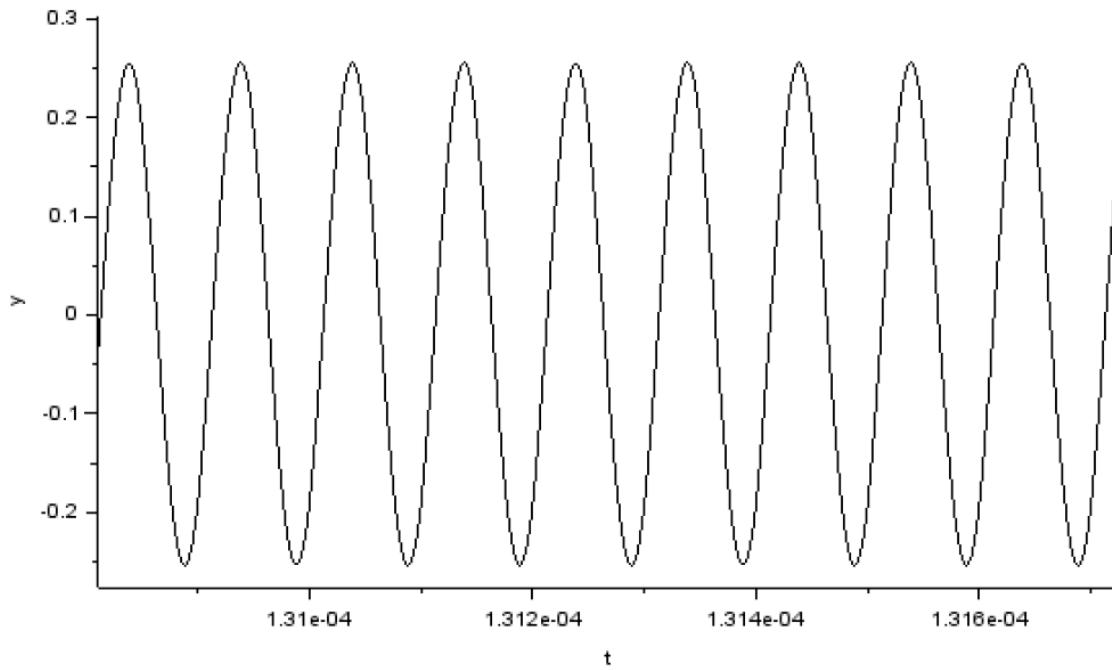
O período do clock adotado é inferior ao do PWM que é dado por:

$$\frac{1}{f_{clk}} = \frac{1}{(10 * 10^6 * 550)} = 1,82 * 10^{-10}$$

O que implica em uma frequência de simulação maior melhorando a representação da onda na simulação. O tempo total de simulação será de 0,005 [s] relativo ao tempo de toda a sequência binária. Na figura abaixo pode-se observar o resultado da onda obtida depois da aplicação do filtro passa faixa, projetado com base nas amostras do PWM.



Aplicando um zoom em uma das partes temos:

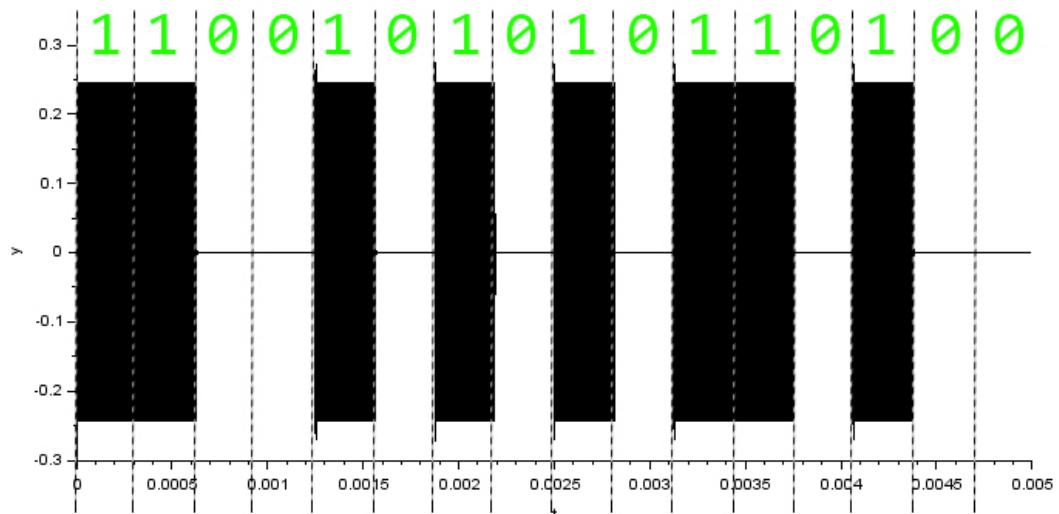


Analisando a frequência da onda, pode-se afirmar que a portadora foi recuperada com a mesma forma de onda e a mesma frequência, além de sua continuidade ser mantida. Pode-se constatar também que houve uma atenuação em sua amplitude devido ao processo realizado pelo PWM. Logo, conclui-se que foi possível recuperar o sinal modulado BASK.

Pelo processo de modulação BASK realizado, pode-se observar no sinal que os bits são representados com clareza ou seja, para o bit 1 no sinal há a presença da portadora e quando o bit é 0 o sinal é nulo. Observa-se então a sequencia da etapa anterior dada por

$$[1 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0]$$

Como é observado na imagem a seguir, sabendo também que período de cada bit é de 0,003125[s]. Cada bit é demarcado detalhadamente na imagem.



Pode-se afirmar que a sequência binária obtida através do uso do filtro passa faixa corresponde à sequencia binária escolhida originalmente. Constatando assim a funcionalidade do filtro projetado bem como a frequênciade amostragem escolhida da onda modulada.

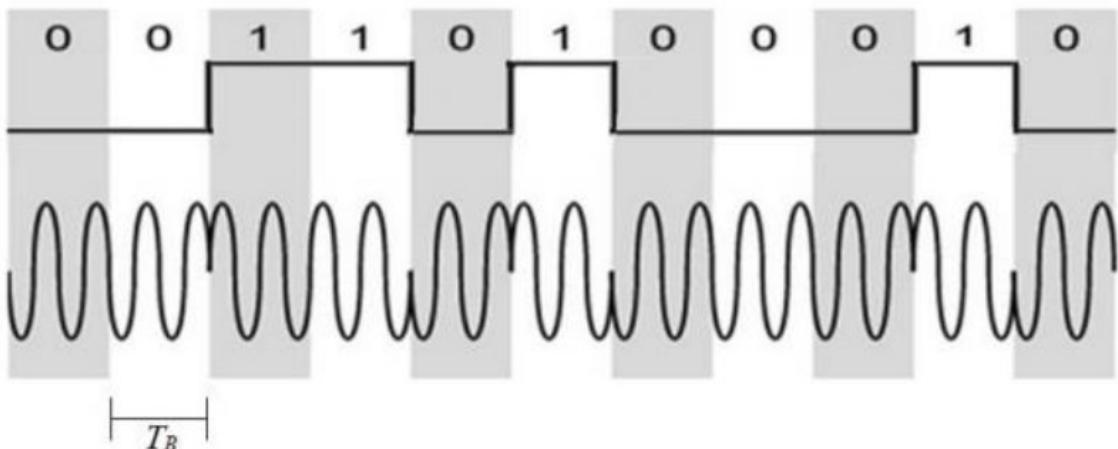
Assim como na etapa 4, para aplicar a modulação digital BPSK escolheu-se uma amostra do sinal PCM criado na etapa 1. A técnica de modulação BPSK terá o seguinte funcionamento:

- Quando o bit for 1 a resposta modulada será:

$$m(t) = 1 * \cos(2\pi f_c t), \text{ onde } f_c = 10[\text{MHz}]$$

- E quando o bit for 0 a resposta modulada será:

$$m(t) = -1 * \cos(2\pi f_c t)$$



Para melhorar a visualização do efeito da modulação BPSK, escolheu-se as amostras 84 e 85 do sinal PCM, como mostrado abaixo:

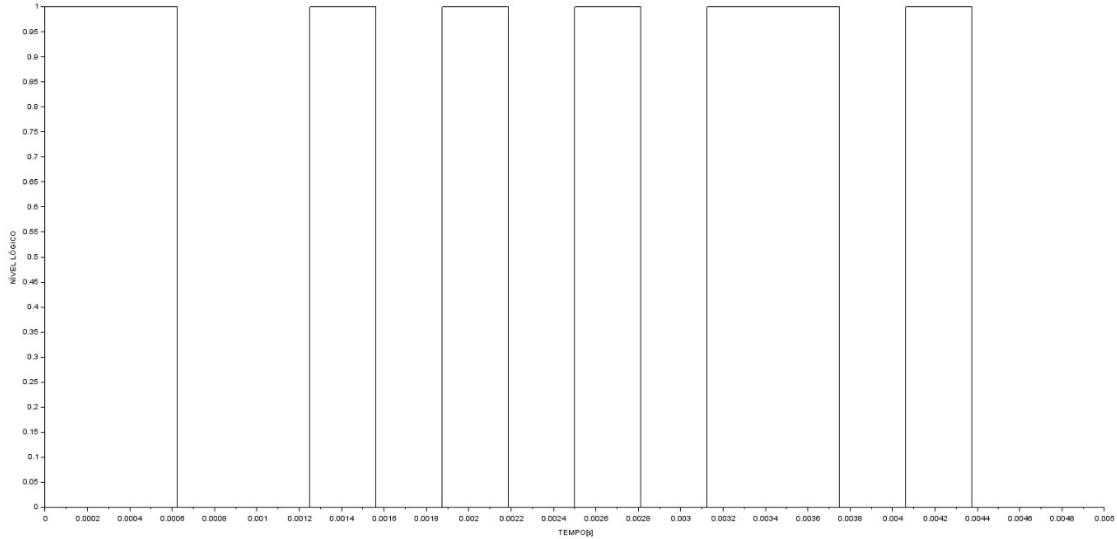
<b>84</b>	1	1	0	0	1	0	1	0
85	1	0	1	1	0	1	0	0

Construção da sequência no Scilab:

```

1 x = [1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0];
2 bp = 1/(400*8);
3 bit = [];
4
5 for n=1:1:length(x)
6   if x(n) == 1;
7     se = ones(1,1000);
8   else
9     se = zeros(1,1000);
10  end
11  bit = [bit se];
12 end
13 t1 = 0:bp/1000:(bp*length(x))-bp/1000;
14 plot2d(t1,bit), xlabel('TEMPO[s]'), ylabel('NÍVEL LÓGICO')

```



A sequência binária pode ser vista claramente. Cada bit vai possuir um período igual a:

$$Período_{bit} = \frac{1}{f_{bit}} = \frac{1}{Número_{amostras_{PCM}} * Número_{bits}} = \frac{1}{400 * 8} = 0,0003125[s]$$

Existem 16 bits na sequência binária, logo o tempo de duração total será de 0,005[s].

Para realizar a obtenção do sinal PWM na modulação BPSK, é necessário primeiramente fazer a amostragem deste sinal modulado. Para descobrir qual é o valor da frequência de amostragem, faz-se necessário saber o valor da banda ocupada pelo sinal modulado com a sequência binária utilizada, com isso será possível encontrar o valor da frequência de amostragem resultante.

Para determinar a banda ocupada na modulação BPSK, faz-se necessário encontrar o valor do espectro de magnitude do sinal. Para realizar tal ação, é utilizada a lógica da modulação BPSK, feita para cada bit da sequência binária. Sendo assim, o resultado da concatenação da aplicação da modulação para cada bit será o sinal modulado total m.

Foi escolhida uma frequência de amostragem significativamente grande o suficiente para realizar a amostragem de maneira bem sucedida, levando em conta que a frequência da portadora é de 10[MHz]. Tal escolha se faz necessária pois se deseja visualizar o espectro de magnitude da forma mais limpa possível, com isso é evitado perdas de informações acarretadas por uma frequência de amostragem ruim. Aplicando o Teorema de Nyquist, é necessária escolher uma frequência de no mínimo 20[MHz]. Portanto, levando em conta que o período do bit é igual a 0,0003125; foi escolhida uma frequência de amostragem seguindo a seguinte fórmula:

$$f_s = \frac{100000}{p_b} = \frac{100000}{0,0003125} = 320[\text{MHz}]$$

Com isso, pode-se notar claramente que se obteve um valor 32 vezes maior que o da portadora.

Vale ressaltar que esta não é a frequência de amostragem final, e sim uma frequência provisória para gerar uma boa visualização de como é o sinal modulado e o seu espectro de amplitude. A

frequência de amostragem final será determinada aplicando-se o teorema de amostragem passa-faixa.

Agora já se faz possível visualizar o sinal modulado BPSK:

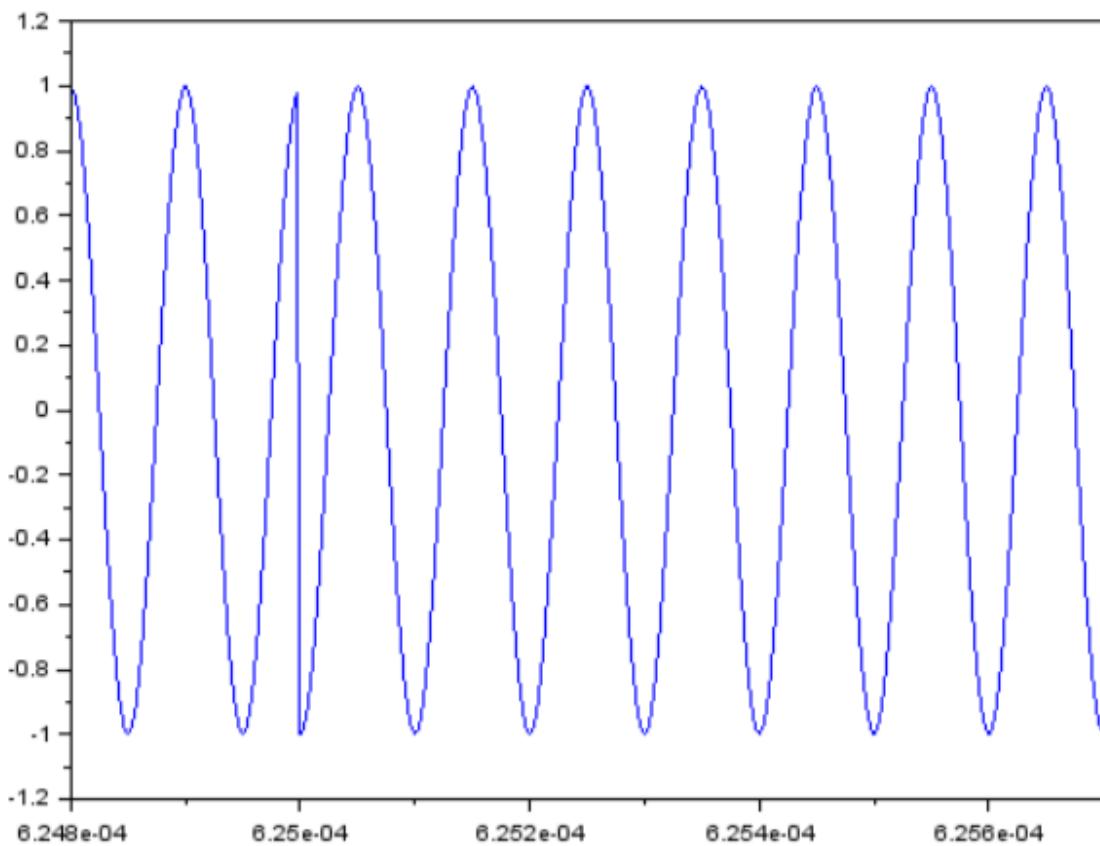
```
18 t2 = 0:bp/100000:bp - bp/100000;
19 m = [];
20 for (i=1:1:length(x))
21   if(x(i)==1)
22     y=1*cos(2*pi*10e6*t2);
23   else
24     y=0;
25 end
26 m = [m y];
27 end
28 t3 = 0:bp/(100000):(bp*length(x))-bp/100000;
29 plot2d(t3,m);
```

Como período do bit é de:

$$Período_{bit} = 0,0003125[s]$$

Pode-se observar uma transição clara de 1 para 0 do segundo para o terceiro bit, havendo uma transição nos cossenos no tempo:

$$2 * Período_{bit} = 2 * 0,0003125 = 0,000625[s]$$



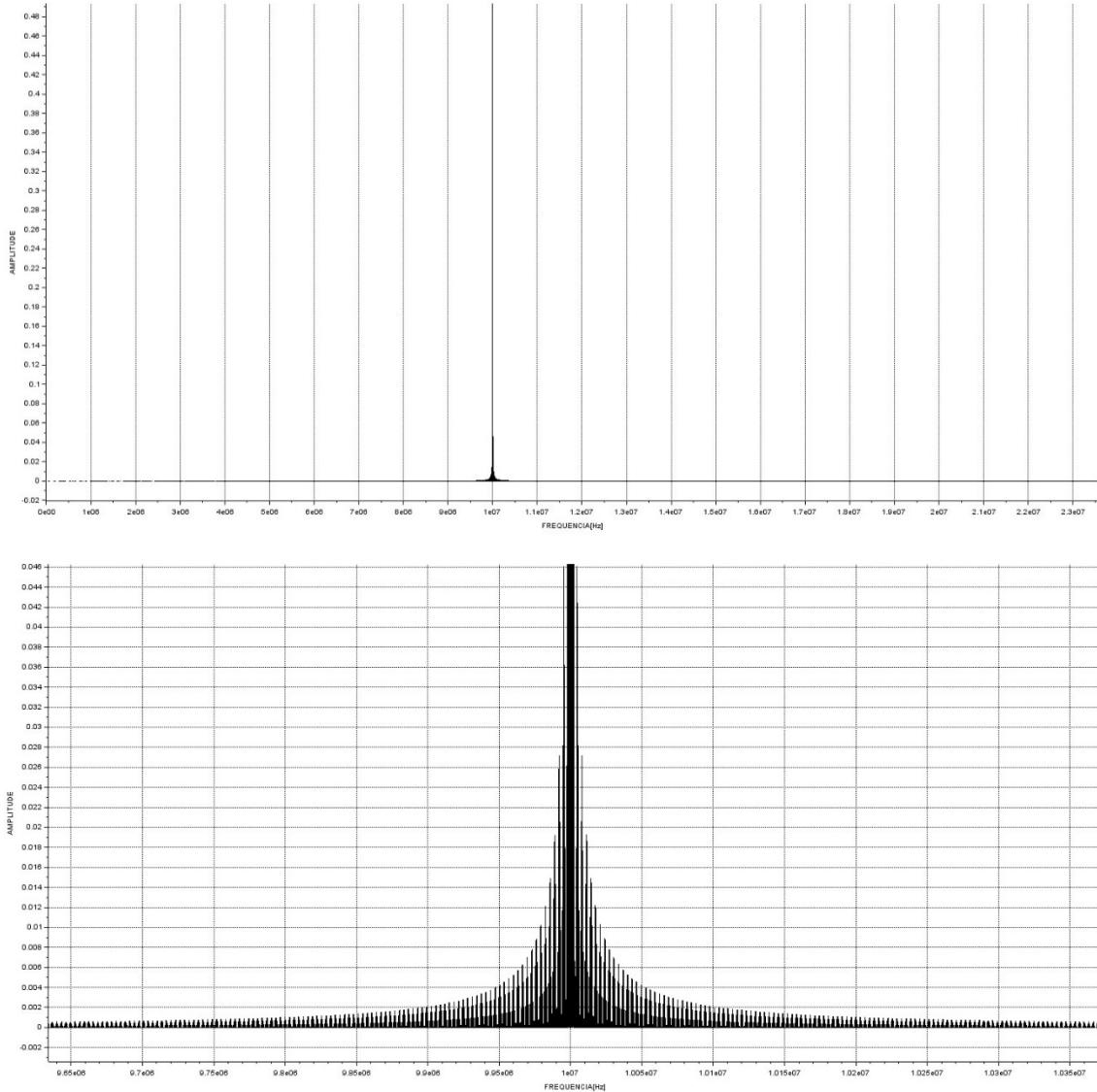
Nota-se claramente os cossenos, bem como sua transição.

O espectro de amplitude é obtido da seguinte forma: tendo as magnitudes do sinal, sendo elas em função das frequências, tem-se que o vetor de amplitude é dado pelo módulo da aplicação da FFT sobre o sinal modulado  $m$  completo multiplicado por 2 e dividido pelo número de amostras do próprio sinal, no entanto, o vetor de frequência é construído com base nos múltiplos da frequência fundamental, que por sua vez é definida como sendo o inverso do período de observação do sinal, indo até o número de amostras do sinal -1. O resultado obtido fica:

```

31 N = length(m);
32 Amp = (2*abs(fft(m))/N);
33 f = 0:1/(bp*length(x)): (N-1)*1/(bp*length(x));
34 plot2d3(f,Amp), xlabel('FREQUENCIA[Hz]'), ylabel('AMPLITUDE')

```



Fazendo a análise do espectro de magnitude, nota-se de forma clara o espectro modulado do sinal, ou seja, a mínima frequência não parte de zero, mas fica centrada a frequência da portadora, que por sua vez, corresponde à 10[MHz], ocupando uma certa banda. Para determinar a frequência de amostragem final do sinal modulado, não é necessário aplicar o teorema de Nyquist, mas sim o teorema de amostragem passa faixa, onde com ele será permitida uma escolha de frequência de amostragem bem menor que a da outra alternativa, além de não gerar sobreposição no espectro e distorção na forma de onda.

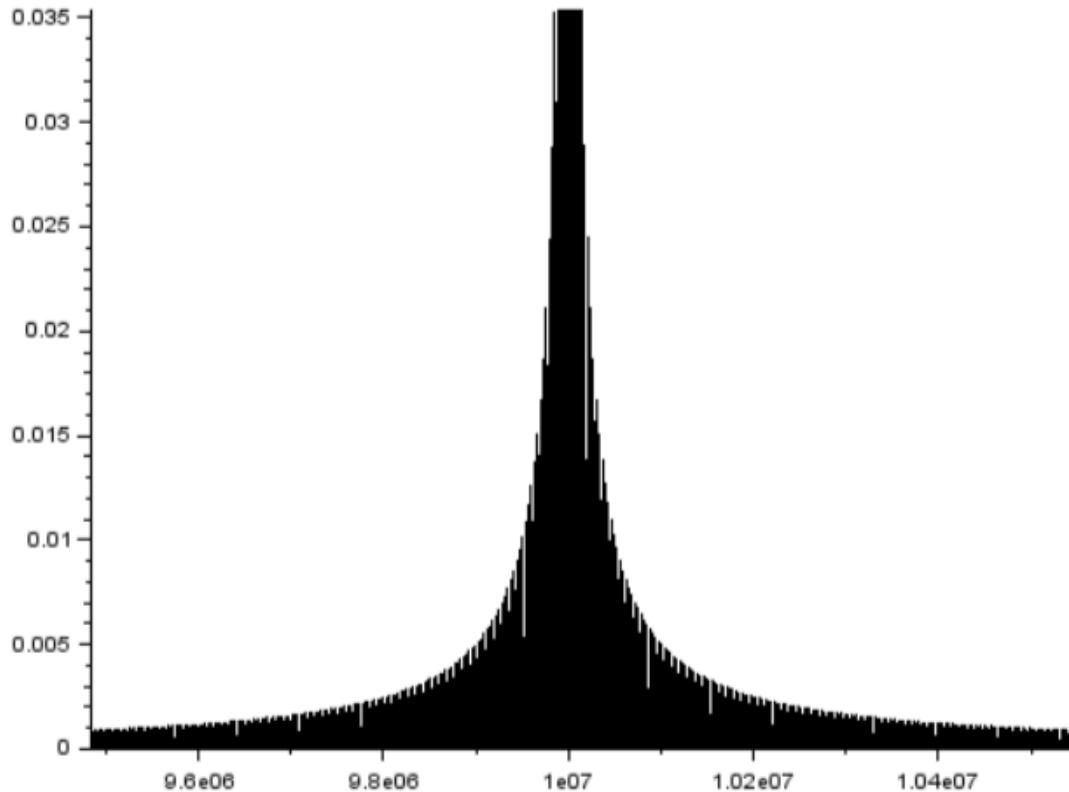
Com o teorema da amostragem passa-faixa faz-se possível recuperar o sinal  $m(t)$ , levando em conta que este foi amostrado com uma frequência  $f_s$  dada por:

$$f_s = \frac{2 * f_{max}}{k}$$

$k$  é dado pelo valor arredondado para baixo, seguindo a divisão abaixo:

$$k = \frac{f_{max}}{B}$$

Usou-se como referência a amplitude média de 0,1 do diagrama de amplitude para se determinar o valor de  $f_{max}$  como sendo a frequência cuja amplitude é 100 vezes menor do que 0,1, isto é, a frequência máxima onde se tem uma amplitude não desprezível. Esse valor é obtido dando um zoom mais aprofundado no espectro de magnitude:



Portanto tem-se uma amplitude de:

$$\frac{0,1}{100} = 0,001$$

Em aproximadamente 10,5[MHz]/9,5[MHz].

A banda ocupada pelo sinal, contendo as amplitudes não desprezíveis, pode ser definida começando a partir de 9,5[MHz] e terminando em 10,5[MHz].

$$10,5[\text{MHz}] - 9,5[\text{MHz}] = 1[\text{MHz}]$$

É importante estipular um valor de banda ocupada igual a 20 vezes a banda que o sinal modulado de fato ocupa, com o objetivo de acomodar com sucesso a transição do filtro e também reduzir a distorção gerada no PWM. Tem-se  $k$  igual a:

$$k = \frac{f_{max}}{20 * B_{real}} = \frac{10,5 * 10^6}{20 * 1 * 10^6} = 0,525$$

A frequência de amostragem final é:

$$f_s = \frac{2 * f_{max}}{k} = \frac{2 * 10,5 * 10^6}{0,525} = 40[\text{MHz}]$$

Como esse valor é um múltiplo da frequência da portadora não haverá a necessidade de arredondar o valor calculado. Logo tem-se que:

$$f_s(\text{ESCOLHIDO}) = 40[\text{MHz}]$$

Percebe-se o quanto foi reduzida a frequência de amostragem necessária para uma recuperação bem sucedida do sinal modulado utilizando o teorema da amostragem passa-faixa, se fosse utilizado o teorema de Nyquist, visando obter o mínimo de perda de informação possível, seria necessária uma frequência em torno de 200[MHz].

Fazendo novamente a construção do sinal modulado BPSK, mas fazendo uso da nova frequência de amostragem. O sinal PWM será obtido com base nesse novo sinal modulado.

```
37 x = [1 1 0 0 1 0 1 0 10 1 1 0 1 0 0];
38 fs = 10000000;
39 bp = 1/(400*8);
40 ts = 1/fs;
41 t2 = 0:ts:bp-ts;
42 m= [];
43 for (i=1:l:length(x))
44     if(x(i)==1)
45         y=1*cos(2*pi*10e6*t2)
46     else
47         y=0;
48    end
49    m = [m y];
50 end
```

Diferente do sinal BASK da etapa 4, o sinal BPSK não tem valores entre 0 e 1, mas sim entre -1 e 1, logo é necessário dar um Offset no sinal modulado. Além disso, com o novo sinal modulado BPSK já determinado é necessário normalizá-lo, isto é, como foram usados 8 bits de quantização para determinar o vetor de amostras de PCM, o maior valor que uma amostra pode assumir é de 255. Como no sinal modulado BPSK o valor máximo assumido pelo cosseno é de 2, devido ao offset aplicado no sinal, será necessário multiplicá-lo por 255/2. Por fim, para não obter valores quebrados será necessário arredondá-los também através do comando round().

```
--> m = m + 1;
--> m = m*(255/2);
--> m = round(m);
```

Analizando a construção do sinal modulado BPSK percebe-se que para cada bit, tem-se um número de ciclo de portadora igual a:

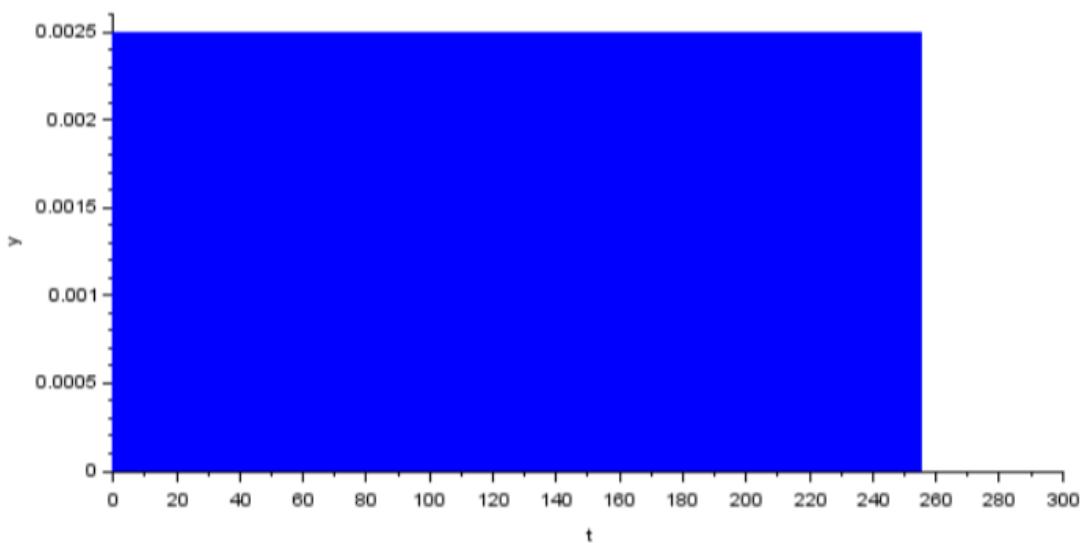
$$p_b * f_s = \frac{1}{(400 * 8)} * 40 * 10^6 = 12500$$

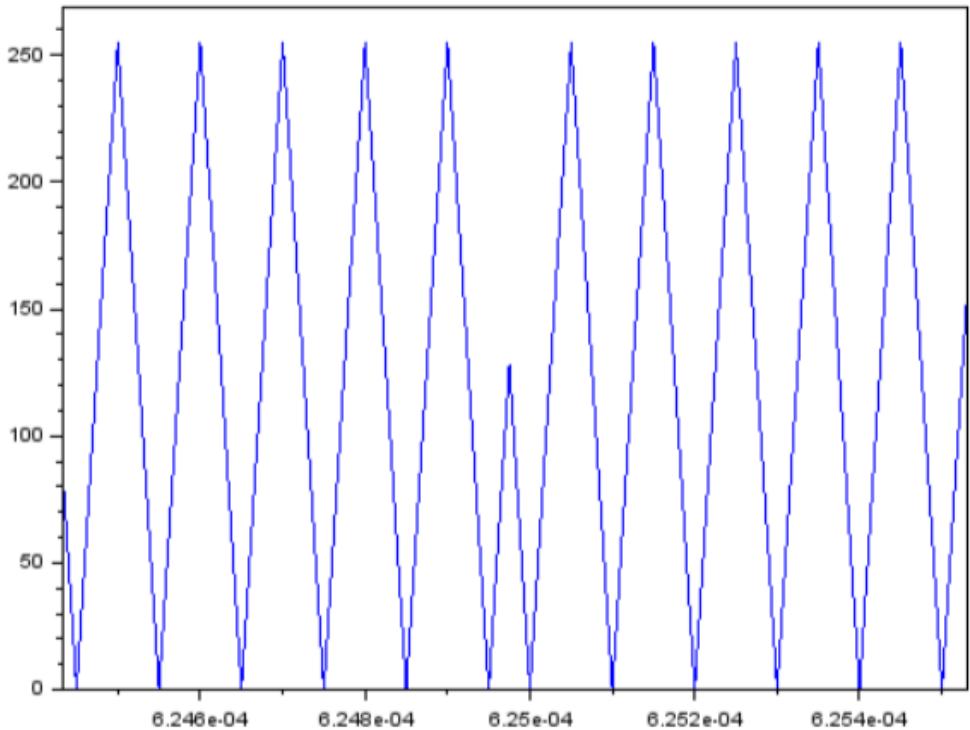
Há 8 bits na sequência binária, então o número total de amostras do sinal modulado é igual a:

$$12500 * 8 = 100000[\text{amostras}]$$

Pode-se plotar o sinal modulado normalizado resultante, especialmente na região de transição do bit 1 para 0, já identificada anteriormente no momento de 0,000625s, da seguinte forma:

```
--> t3 = 0:ts:(bp*length(x)) - ts;  
--> plot(t3,m)
```





Mesmo que a onda tenha um aspecto triangular ao invés de cossenoidal, pode-se perceber a mesma característica da transição de bits com a modulação BPSK, que não afetará no resultado final do PWM nem na recuperação da onda modulada a partir do sinal PWM, pois o teorema de amostragem passa-faixa foi respeitado. Com a onda modulada BPSK já normalizada e definida, faz-se possível obter o PWM da mesma. Primeiramente, esse processo será realizado na FPGA através do software Quartus, onde os valores das amostras do sinal modulado passados a ele devem estar na forma binária. Essa conversão de decimal para binário é feita utilizando o comando `dec2bin()`.

```
|--> mbin = dec2bin(m);
```

O PWM é gerado fazendo a comparação do valor absoluto de cada amostra da onda modulada com uma onda dente de serra. Sendo assim, enquanto a onda dente de serra for menor que o valor da atmosfera da onda modula, a saída terá nível lógico alto, e quando não for menor, a saída terá nível lógico baixo, produzindo assim uma modulação em largura de pulso.

O PWM gerado na FPGA se dará realizando a comparação entre amostra e o valor em contador, que por sua vez, será incrementado por um clock, mas levando em conta que o contador resetará para zero quando ultrapassar o valor máximo, com isso, é visto que se deve atualizar a amostra utilizando a próxima a seguir.

A definição do clock se dará levando em conta o valor da frequência de amostragem, que corresponde à 40[MHz].

$$f_s = \frac{f_{clk}}{c_{max} + 1}$$

$c_{max}$  é o máximo valor do contador, cujo valor deve respeitar a seguinte inequação:

$$c_{max} = 2^{N+1}$$

Onde N é o número de bits de quantização utilizado. Esta inequação garante que o sinal PWM gerado não possua Duty Cicle superior a 50% da frequência de amostragem. Como foram utilizados 8 bits para quantização, logo:

$$c_{max} > 2^9 \Rightarrow c_{max} > 512$$

Para obter um valor de clock redondo, foi escolhido  $c_{max} = 549$ , satisfazendo assim a inequação. Assim, tem-se:

$$f_s = \frac{f_{clk}}{c_{max} + 1} \Rightarrow f_{clk} = f_s(c_{max} + 1) = 40 * 10^6 * (549 + 1) \Rightarrow f_{clk} = 22000[\text{MHz}]$$

Com todos os parâmetros já definidos, foi feita uma lógica em Verilog para gerar o circuito que criará o sinal PWM. Foi programado um módulo para o contador, amostragem e comparador.

Módulo das amostras:

```
module Amostras(
    input clk,
    input [15:0] A,
    output reg [11:0] amostra);

    reg [7:0] mem [0:99999];
    initial begin
        $readmemb("D:/Google_Drive/faculdade/2021.2/PBLE04/FPGA/C.txt",mem);
    end

    always @(A) begin
        amostra = mem[A];
    end
endmodule
```

É criado um vetor 100000x8 para então armazenar os valores e preencher cada uma dessas posições com as amostras do sinal modula BPSK normalizado através de um arquivo txt. Vale ressaltar que a saída do módulo via ser controlado pelo valor de entrada A.

O módulo do contador é dado por:

```

module Contador(
    input clk,
    output reg[11:0] cont ,
    output reg[15:0] A);

    initial cont =12'd0;
    initial A = 16'd0;

    always @ (posedge clk)begin
        cont <= cont + 12'd1;
        if(cont >= 12'd549)begin
            cont <=12'd0;
            A <= A + 16'd1;
            if(A==16'd99999)begin
                A<= 16'd0;
            end
        end
    end
endmodule

```

O clock possui valor máximo de 549 e é incrementado de 1 em 1, assim que ultrapassado este valor, ele volta a ser zero, recomeçando todo o processo. O contador é utilizado também para servir de indicação de posição de memória, ou seja, qual amostra deve ser comparada para gerar o PWM. Quando o contador se reinicia, deve-se atualizar a amostra fazendo uso da próxima. Isto é feito realizando a incrementação de 1 na variável de controle da posição da amostra toda vez que o valor do contador é extrapolado. Como existem 100000 amostras, o processo é realizado até chegar a este valor, fazendo com que o valor da posição da amostra volte a 0, reiniciando assim o ciclo.

O módulo de comparação é:

```

module comparador(
    input[11:0] amostra,
    input[11:0] cont,
    output reg pwm);

    always @* begin
        if(amostra > cont) pwm = 1;
        else pwm =0;
    end
endmodule

```

Este módulo apenas realiza a comparação do valor das entradas referentes à amostra atual e do contador, e se o valor absoluto da amostra é maior do que o contador a saída PWM é igual a 1, se não a saída PWM é 0.

Realizando a conexão entre os 3 módulos, tem-se:

```

module pwm(
    input clk,
    output saida);

    (*keep=1*) wire[11:0] cont;
    (*keep=1*) wire[11:0] amostra;
    (*keep=1*) wire[15:0] A;

    Contador C (
        .clk(clk),
        .cont(cont),
        .A(A));

    Amostras Amos(
        .clk(clk),
        .A(A),
        .amostra(amostra));
    | 
    comparador comp(
        .amostra(amostra),
        .cont(cont),
        .pwm(saida));

endmodule

```

O módulo é conectado da seguinte forma: a saída do módulo contador A vai para a entrada do módulo amostras para indicar qual posição da memória e consequentemente qual amostra deverá ser comparada. Então essa amostra e a saída cont do módulo contador vão para a entrada do módulo comparador onde são comparadas a fim de gerar o sinal PWM.

É implementado um clock com período de 0,04544[ns], o que gera uma frequência aproximada de 22000[MHz]:

```

module pwm_TB;
reg clock;
reg [11:0] cont;
reg [11:0] amostra;
reg [15:0] A;
wire saida;

pwm DUT(
    .clock(clock),
    .saida(saida) );

initial begin
    clock = 0;
end

always begin
    #0.091 clock =~clock;
end

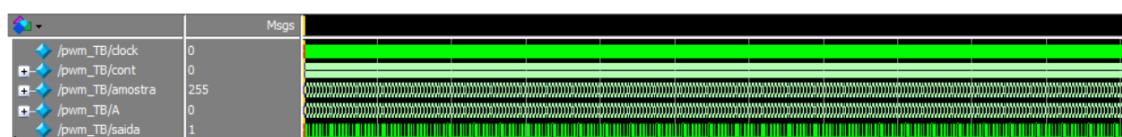
initial begin
$init_signal_spy("/pwm_TB/DUT/cont","cont",1);
$init_signal_spy("/pwm_TB/DUT/amostra","amostra",1);
$init_signal_spy("/pwm_TB/DUT/A","A",1);
end

initial
    #550000 $stop;
|

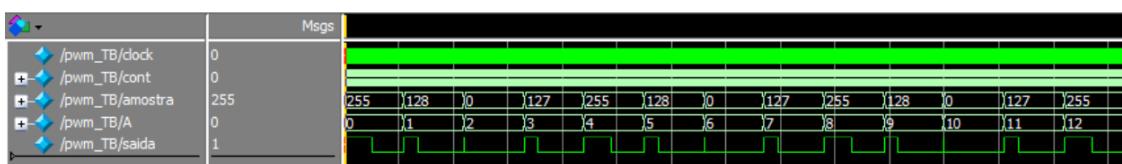

endmodule

```

Tem-se como resultado de simulação:



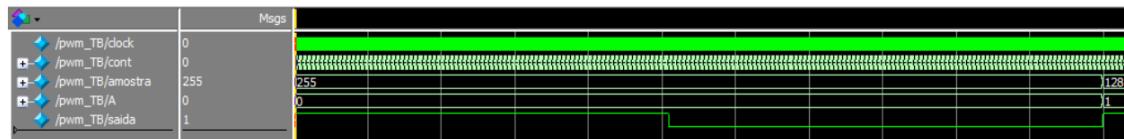
É visualizada as 12 primeiras amostras do sinal modulado BPSK onde era representado o bit 1, e em seguida as primeiras amostras do primeiro bit 0 da amostra PCM escolhida, nota-se que:



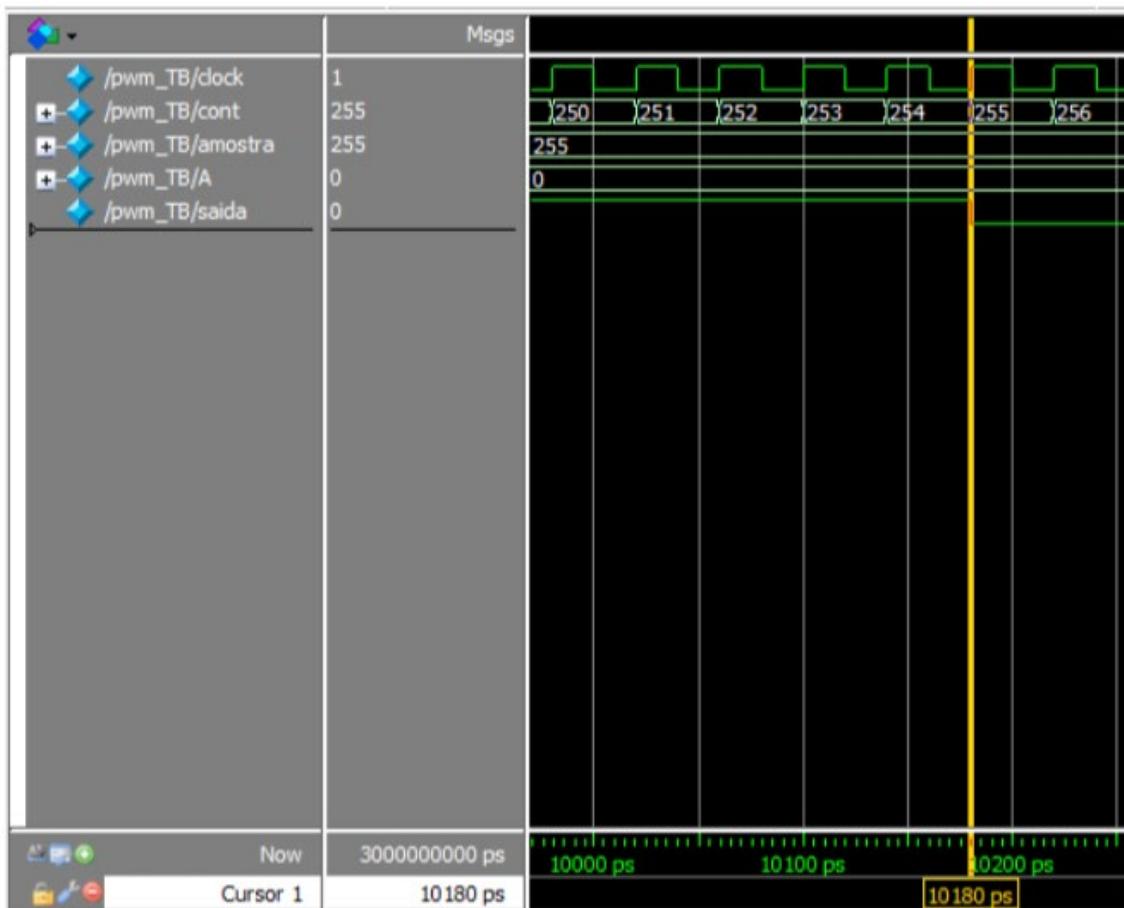
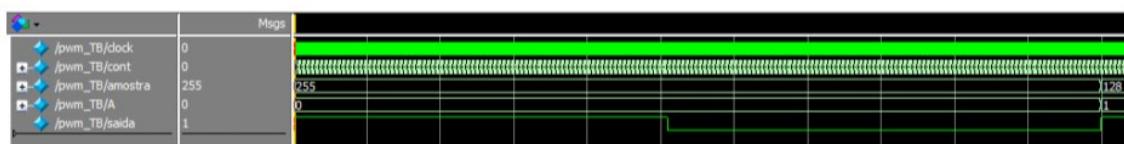
Pode-se ver então que quando a amostra é igual a 0, o sinal PWM resultante também é igual a 0 e quando a amostra é igual a 255, o sinal PWM terá um período em estado alto até o contador passar esse valor e então volta para o estado baixo.

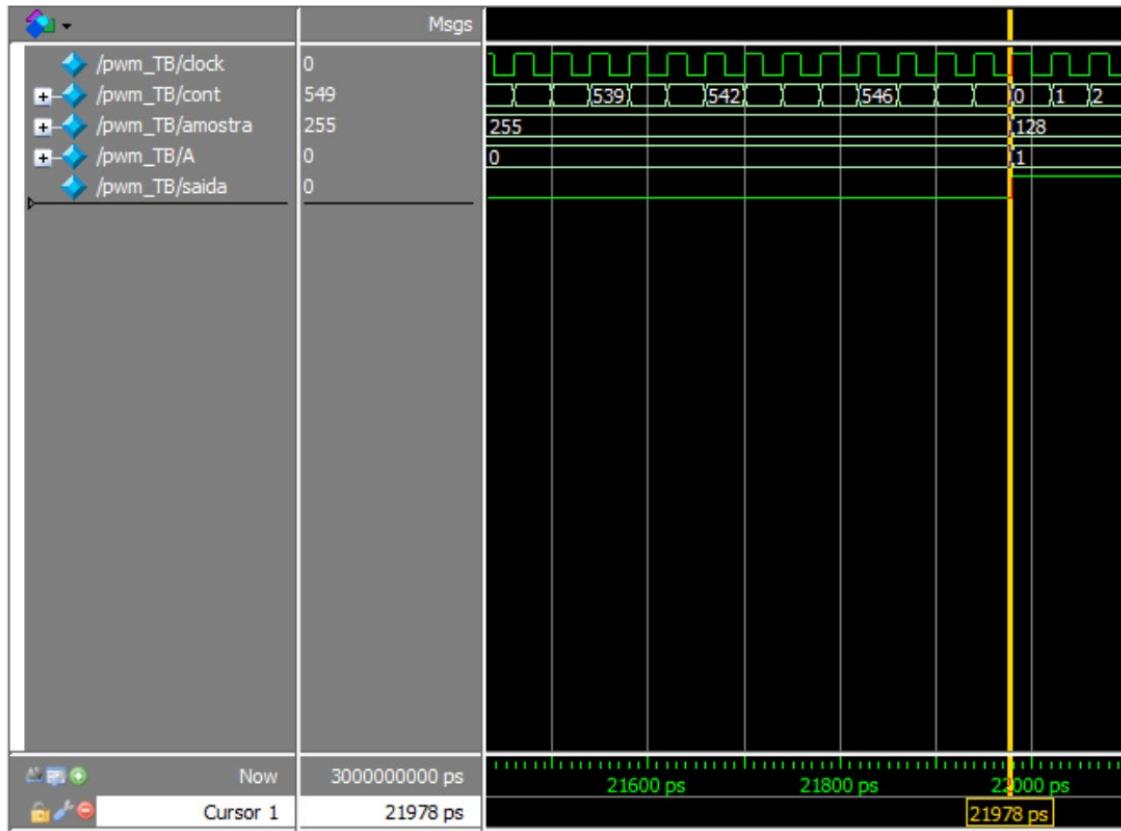
O funcionamento deste módulo se dá analisando o comportamento da onda de PWM em uma única amostra através da comparação entre a proporção do tempo em que a saída fica em 1 e

o tempo total do período da onda PWM para essa amostra em específico e a proporção do valor absoluto da amostra com o valor máximo do contador igual a 549. Teoricamente, eles deveriam ser iguais ou muito próximos. Realizando a escolha da primeira amostra para fazer tal teste, é visto:



O sinal PWM muda de nível lógico em 10180[ps]. O contador extrapola no seguinte instante:





Ou seja, em 21978[ps]. Em teoria, o valor ideal seria de:

$$\frac{1}{f_s} = \frac{1}{40M} = 25000[\text{ps}]$$

A simulação foi realizada em Gate Level, sendo assim, os tempos de atrasos inerentes aos componentes foram levados em conta.

As proporções ficam da seguinte forma:

$$\frac{\text{Valordo amostra}}{\text{Valormáximo do contador}} = \frac{255}{549} = 0,46448$$

$$\frac{\text{Tempo em nível lógico alto}}{\text{Período da onda para amostra}} = \frac{10180[\text{ps}]}{21978[\text{ps}]} = 0,46319$$

Comprova-se a eficácia do módulo coma base na constatação da mínima diferença entre as proporções.

Após realizada a implementação no Quartus, será executado algo similar no Scilab, empregando-se as mesmas variáveis.

No Scilab, o vetor PWM de saída será:

$$(c_{\max} + 1) * \text{Número de amostras} = 550 * 100000 = 55000000[\text{amostras}]$$

É feito o algoritmo para realizar a comparação entre o valor de cada uma das amostras da onda modulada BPSK e o contador. É utilizado um **for** de 1 a 100000 para varrer cada uma das amostras e outro **for** de 1 a 550 para varrer todos os valores possíveis do contador.

Foi feita a comparação entre o valor do contador e o valor da amostra no índice indicado pelo **for** através do bloco **if**. Restando apenas concatenar cada operação de comparação com os valores abstraídos do PWM, afinal, quando extrapolado o valor do contador passa-se para a seguinte posição do vetor de amostras e assim começa um novo ciclo de obtenção dos valores do PWM. Para realizar a solução de tal problema, utiliza-se de uma variável auxiliar a fim de corrigir a posição do PWM, ou seja, conforme o índice da posição do vetor de amostras aumenta em 1, o índice de posição do vetor de PWM deverá ser incrementado em 550, que corresponde justamente ao valor de extração do contador. Assim, tem-se:

```

for i=1:100000
    for cont = 1:550
        j = i-1;
        aux = cont + 550*j;
        teste = aux;
        if(cont<m(i))
            PWM(aux)=1;
        else
            PWM(aux)=0;
        end
    end
end

```

Para averiguar o código, plota-se o gráfico de **t** em função do valor do vetor PWM, onde **t** é o vetor temporal que vai de 0 a 0,0025s (tempo total da sequência binária) incrementado pelo período do clock igual ao inverso de 22000 MHz. Desta forma, tem-se:

```

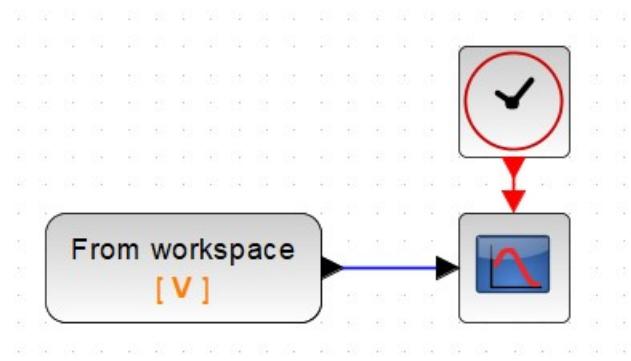
69 fclk = fs*550;
70 tclk = 1/fclk;
71 t = 0:tclk:(bp*length(x)-tclk);

```

O gráfico do PWM em função do tempo foi feito pelo Xcos, onde os valores do console foram passados a ele através do comando **struct()**.

```
V = struct('time',t,'values',PWM);
```

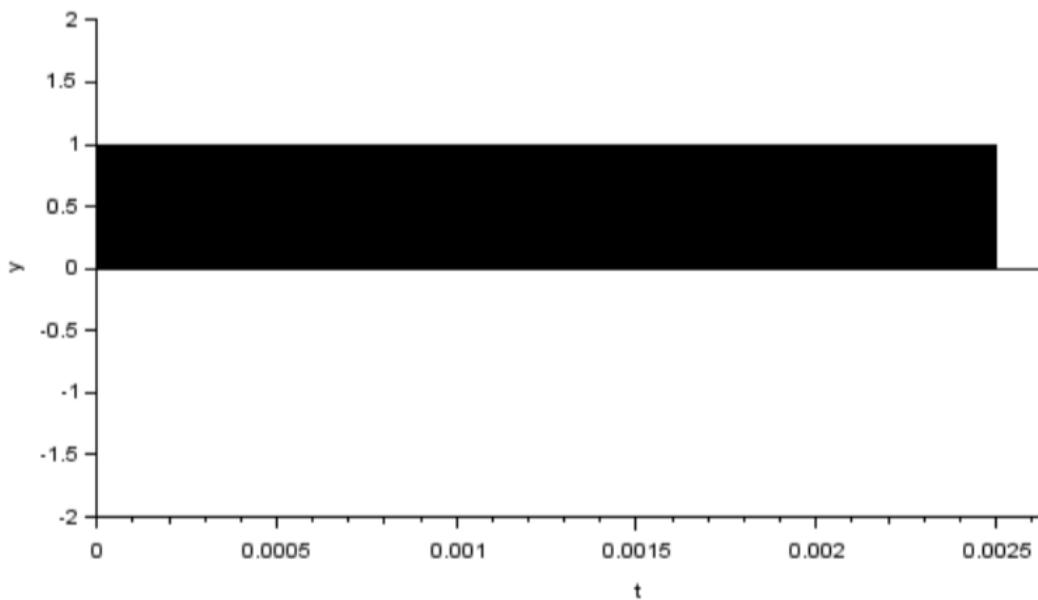
Obteve-se o seguinte diagrama de blocos:



O período de clock foi determinado sendo igual a  $f_{clk} = 1 * 10^{-11}$ , valor este inferior ao período do PWM, sendo igual a:

$$\frac{1}{f_{clk}} = \frac{1}{40 * 10^6 * 550} = 4,54 * 10^{-11}$$

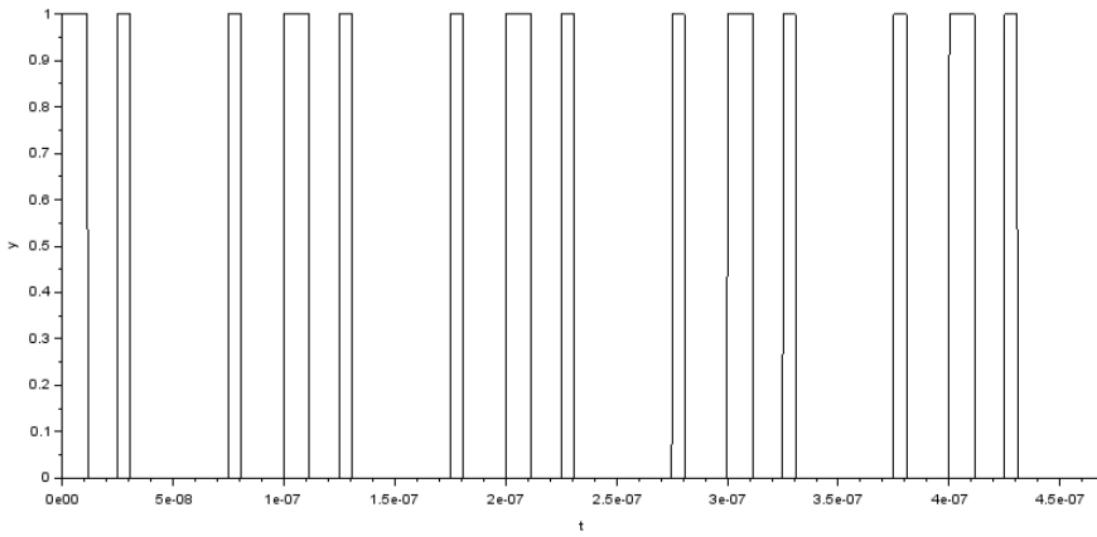
Assim, é obtida uma representação fiel da onda resultante, cujo resultado da simulação no osciloscópio, com tempo total de 0,003[s], é mostrado logo abaixo:



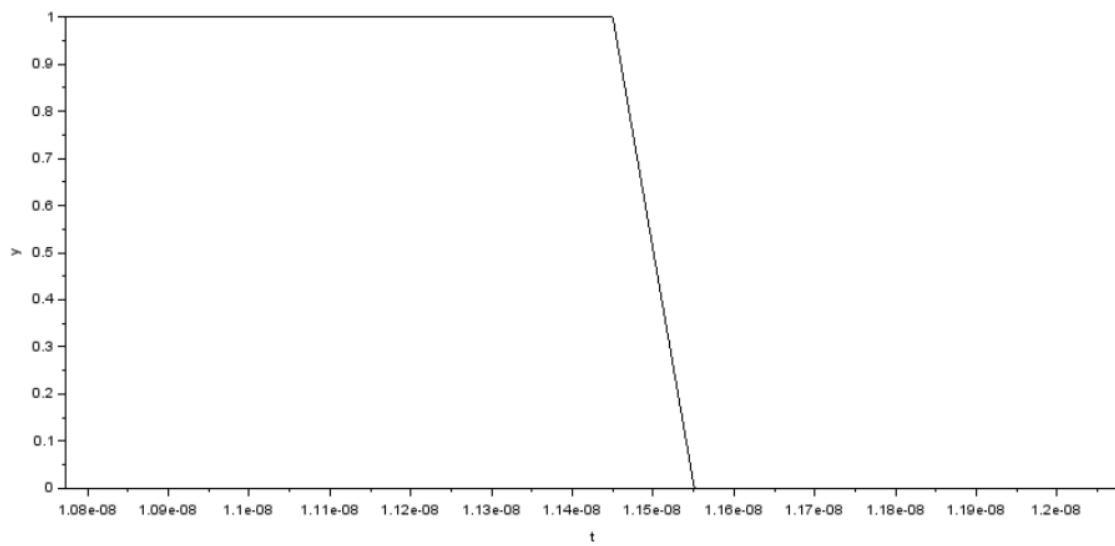
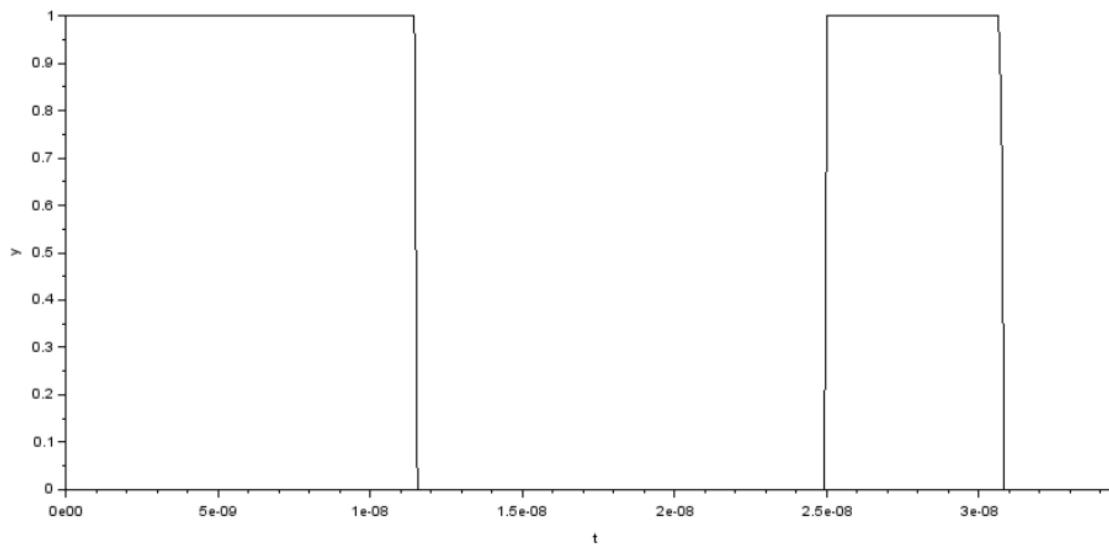


Onde pode-se ver o comportamento geral da onda PWM, em que o comprimento do estado lógico alto varia entre  $\frac{255}{550}$ ,  $\frac{125}{550}$  e 0, isto é, ele será maior de acordo com o tamanho da amostra. Pode-se dar um zoom para ter uma verificação mais clara deste comportamento.

Pode-se observar o comportamento da onda PWM, na qual o comprimento do estado lógico alto varia entre  $\frac{255}{550}$ ,  $\frac{125}{550}$  e 0, ou seja, será proporcional ao tamanho da amostra. Analisando-se com um zoom temos:

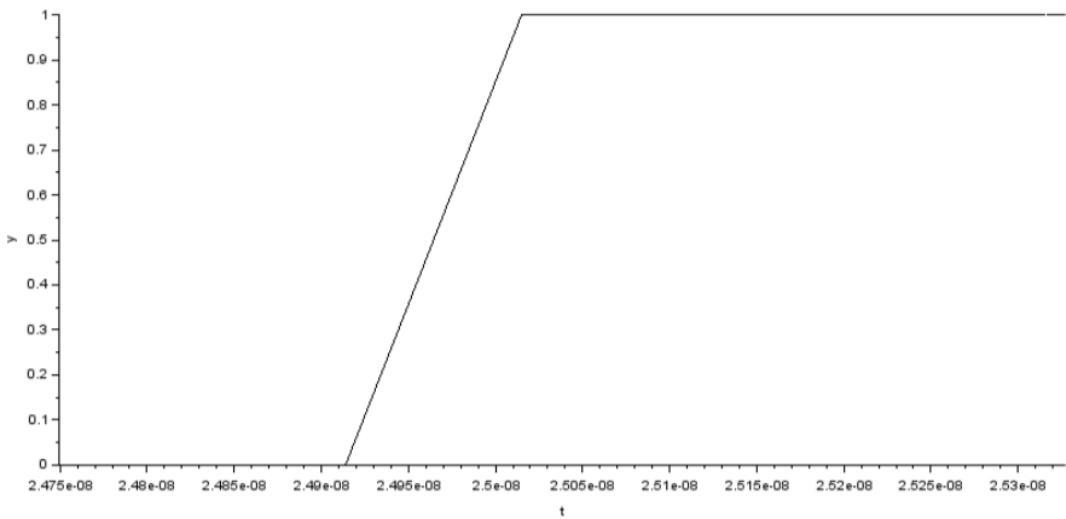


Pode-se aplicar novamente a comparação entre as proporções de tempo e valores já explicadas anteriormente. Tomando-se mais uma vez como exemplo a primeira amostra, tem-se que o momento em que é mudado seu nível lógico alto para baixo no pwm é:



O instante corresponde à  $1,155 * 10^{-8}[s]$ .

O instante de extração do contador é:



O instante é de aproximadamente  $2,491 \times 10^{-8} [s]$ .

Assim, fazendo os cálculos das proporções, tem-se:

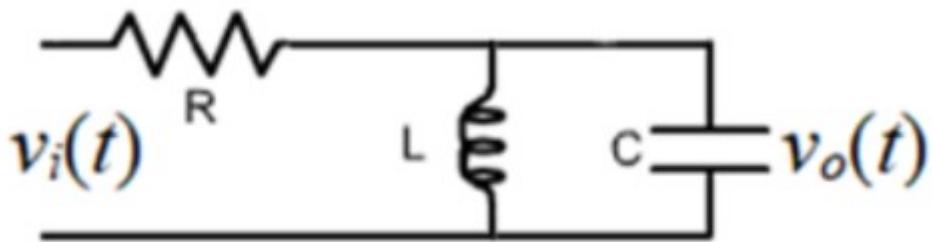
$$\frac{\text{Valordo amostra}}{\text{Valormáximo do contador}} = \frac{255}{550} = 0,46364$$

$$\frac{\text{Tempo em nível lógico alto}}{\text{Período da onda para amostra}} = \frac{1,155 \times 10^{-8}}{2,491 \times 10^{-8}} = 0,46367$$

Com base na análise entre os resultados obtidos no Quartus e Scilab pode-se constatar que houve uma pequena diferença entre as proporções de 0,00003, comprovando-se a eficácia do algoritmo implementado. Além disso, pode-se confirmar a correspondência entre o PWM implementado no Quartus com o implementado no Scilab.

Sendo obtido o sinal PWM com base no sinal modulado BPSK é necessário realizar a conversão das amostras do PWM para o sinal analógico correspondente ao sinal modulado BPSK original e para isso realiza-se a filtragem utilizando um filtro passa baixa. A aplicação do filtro passa-faixa no sinal PWM retorna ao sinal modulado. Esta recuperação do sinal analógico teve como princípio o teorema da amostragem passa faixa que foi usada para determinar a frequência de amostragem do sinal modulado. Isso foi feito pois se estava tratando de um sinal modulado, ou seja, com uma banda de passagem que não partia do zero, mas sim que estava centrada na frequência da portadora.

O filtro passa-faixa pode ser realizado com o circuito abaixo:



Cuja função de transferência é dada por:

$$H(jw) = \frac{\frac{(j\omega)}{RC}}{(j\omega)^2 + \frac{1}{RC}(j\omega) + \frac{1}{LC}}$$

Onde  $\frac{1}{RC}$  corresponde à banda de passagem do filtro passa-faixa,  $\Delta w$  e  $\frac{1}{LC}$  diz respeito à frequência central ao quadrado, isto é,  $\omega^2$ .

Através da análise do espectro de magnitude do sinal modulado, foram determinados os dois parâmetros na etapa anterior no momento de calcular a frequência de amostragem para o sinal modulado utilizando o teorema de amostragem passa-faixa, obtendo uma banda de passagem de 1MHz e frequência central de 10 MHz. Logo, tem-se que:

$$\left\{ \begin{array}{l} \frac{1}{RC} = B = 2\pi * 1 * 10^6 \\ \frac{1}{LC} = \omega^2 = (2\pi * 10 * 10^6)^2 \end{array} \right.$$

Onde converteram-se os valores dos parâmetros de Hz para rad/s, pois é nesta unidade pelo qual é projetado o filtro.

Para a determinação dos valores escolhidos para os componentes, observa-se que há no total 3 incógnitas e 2 equações no sistema. Desta forma, é necessário determinar uma componente e então calcular os valores das demais.

Escolhendo C igual a 1 nF, tem-se que:

$$\frac{1}{RC} = 2\pi * 1 * 10^6 \rightarrow \frac{1}{(2\pi * 1 * 10^6 * 1 * 10^{-9})} = 159,15\Omega$$

O valor comercial de resistor mais próximo ao calculado é de  $R = 160\Omega$ .

$$\frac{1}{LC} = (2\pi * 1 * 10^6)^2 \rightarrow \frac{1}{((2\pi * 1 * 10^6)^2 * 1 * 10^{-9})} = 253,30\text{nH}$$

O valor comercial de indutor mais próximo ao calculado é de  $L = 240\text{nH}$ .

Os valores tabelados comerciais para os resistores e comerciais foram abstraídos das seguintes tabelas:

## Resistores Comerciais

1.0ohm	1.1ohm	1.2ohm	1.3ohm
1.5ohm	1.6ohm	1.8ohm	2.0ohm
2.2ohm	2.4ohm	2.7ohm	3.0ohm
3.3ohm	3.6ohm	3.9ohm	4.3ohm
4.7ohm	5.1ohm	5.6ohm	6.2ohm
6.8ohm	7.5ohm	8.2ohm	9.1ohm

## Capacitores Comerciais

1.0F	1.1F	1.2F	1.3F
1.5F	1.6F	1.8F	2.0F
2.2F	2.4F	2.7F	3.0F
3.3F	3.6F	3.9F	4.3F
4.7F	5.1F	5.6F	6.2F
6.8F	7.5F	8.2F	9.1F

## Indutores Comerciais

1.0H	1.1H	1.2H	1.3H
1.5H	1.6H	1.8H	2.0H
2.2H	2.4H	2.7H	3.0H
3.3H	3.6H	3.9H	4.3H
4.7H	5.1H	5.6H	6.2H
6.8H	7.5H	8.2H	9.1H

Do site:

<http://www3.eletronica.org/dicas-e-hacks/valores-comerciais-de-resistores-capacitores-indutores-e-fusiveis>

Desta forma, os valores resultantes de  $\frac{1}{RC}$  e  $\frac{1}{LC}$  são de:

$$\left\{ \begin{array}{l} \frac{1}{RC} = \frac{1}{160 * 1 * 10^{-9}} = 6250000 \\ \frac{1}{LC} = \frac{1}{240 * 10^{-9} * 1 * 10^{-9}} = 416666666666666,66 \end{array} \right.$$

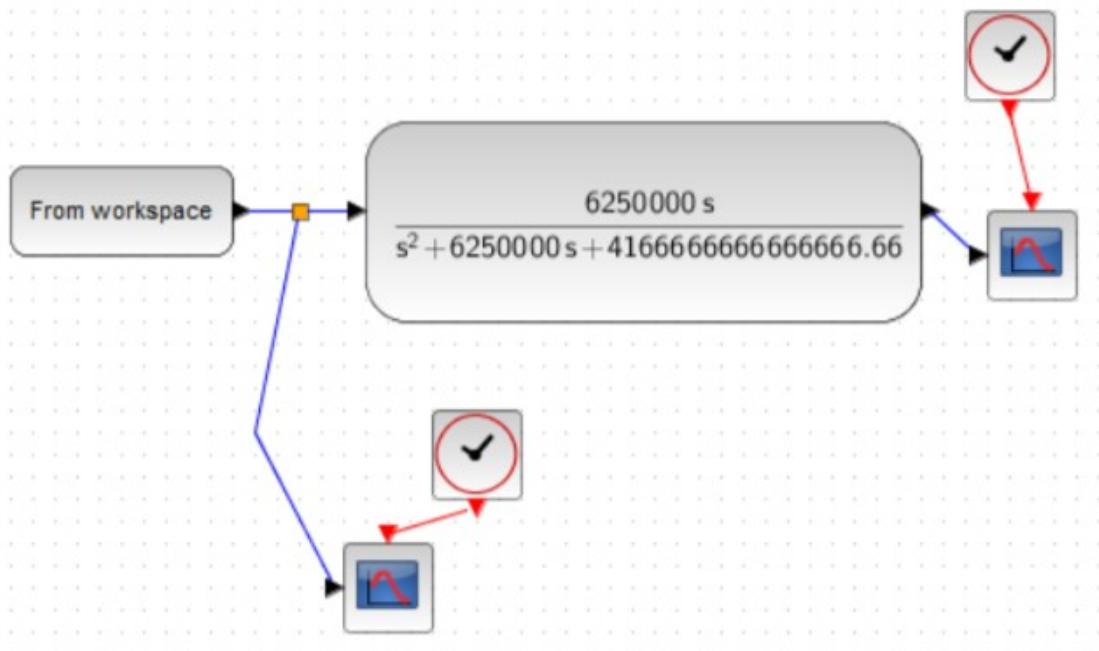
E com isso, a função de transferência do filtro passa-faixa será dada por:

$$H(s) = \frac{\frac{1}{RC}s}{s^2 + \frac{1}{RC}s + \frac{1}{LC}} = \frac{6250000s}{s^2 + 6250000s + 416666666666666,66}$$

Após o desenvolvimento da função, é possível realizar a simulação no Xcos, onde os valores das amostras PWM e do vetor de tempo do PWM, obtidos na etapa anterior, são passadas do console para o Xcos através do comando struct:

```
> V = struct('time',t,'values',PWM);
```

O diagrama de blocos no Xcos fica da seguinte forma:



No qual foi definido o período do clock como:  $1 \times 10^{-10}$ , valor inferior ao período do PWM, igual a:

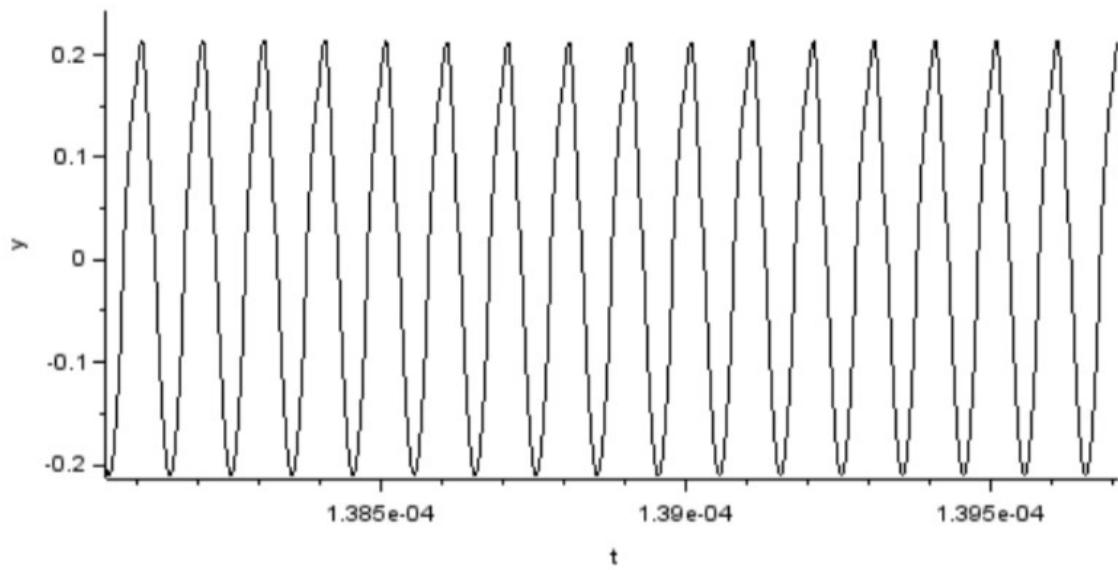
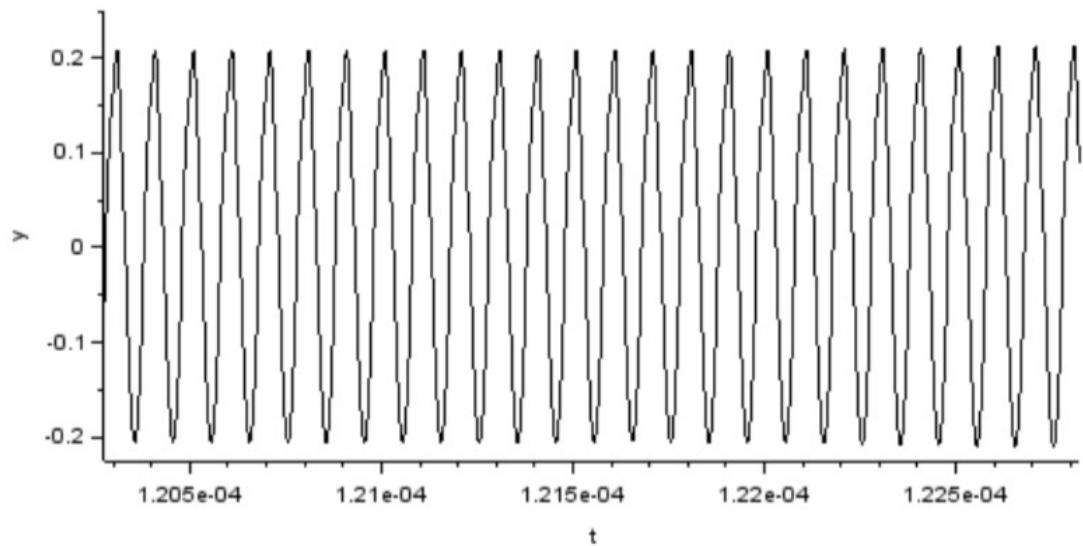
$$\frac{1}{f_{clk}} = \frac{1}{(40 * 10^6 * 550)} = 4,54 * 10^{-10}$$

Com isso, teremos uma representação mais correta da onda resultante, cuja simulação terá um tempo total de 0,0025s (tempo completo da sequência binária total que possui 8 bits, cujo período de cada um é:

$$\frac{1}{400 * 8} = 0,0003125s$$

O resultado da simulação para os valores da onda PWM já foram mostrados na etapa anterior. Já o resultado da onda obtida após a aplicação do filtro passa-faixa projetado nas amostras do PWM original em relação com o tempo total de simulação é impreciso, visto a alta frequência da onda portadora.

Pode-se ver então o comportamento geral da onda resultante, onde um zoom mais aprofundado revela o seguinte:



A recuperação da portadora, com a mesma forma de onda e frequência e de forma contínua, foi realizada com sucesso. Verificou-se que a frequência do sinal resultante é igual à da portadora pegando dois pontos com o mesmo valor consecutivo. Por exemplo, nas menores amplitudes do sinal em volta do instante  $1,385 \times 10^{-4}$ , que acontecem aproximadamente em  $1,3855 \times 10^{-4}$ s e  $1,3845 \times 10^{-4}$ .

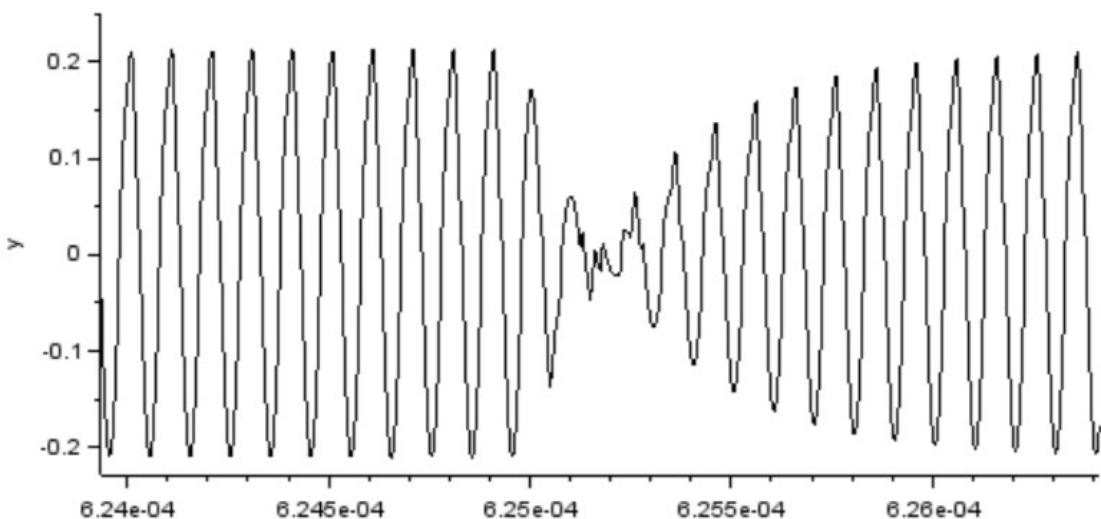
Logo a frequência da onda resultante é:

$$\frac{1}{(1,3855 * 10^{-4} - 1,3845 * 10^{-4})} = 10MHz$$

que é exatamente a frequência da onda portadora.

É possível perceber também que a onda resultante possui uma amplitude menor se comparada com a onda portadora original, decorrente do processo realizado pelo PWM. Portanto, conseguiu-se recuperar o sinal modulado BPSK com sucesso. O motivo para não se ter uma onda cossenoidal exatamente perfeita pode ser indicado como sendo o período de amostragem do clock, que foi escolhido como aproximadamente 0,25 vezes o período da onda PWM. O ideal seria pegar um valor 10 vezes menor ao menos. Entretanto, uma escolha menor para este período de amostragem resultaria em um custo computacional muito alto, que levaria a um tempo de processamento e finalização da simulação muito alto.

Pode-se observar corretamente o processo de modulação BPSK, no instante da transição entre o bit 1 e o bit 0. O primeiro instante que ocorre essa transição já foi determinado na etapa anterior e é em 0,000625s visto que a sequência binária é dada por [1 1 0 0 1 0 1 0] e o período do bit é igual a 0,0003125s. Vendo este instante na onda recuperada, percebe-se que:



Nota-se claramente que neste instante de transição de bits com estado lógico diferentes há a inversão da onda cossenoidal, exatamente o que deveria acontecer em uma onda modulada BPSK, provando assim a eficácia tanto do filtro projetado como da escolha de frequência de amostragem da onda modulada utilizando o teorema da amostragem passa-faixa.

Depois da aplicação do filtro e a obtenção da onda modulada BPSK, para se obter a sequência binária completa seria necessário aplicar por exemplo a demodulação coerente, que consiste basicamente em um modulador de produto, onda multiplica-se a onda BPSK filtrada pela portadora (uma com sinal positivo e outra com sinal negativo), seguido por um filtro passa-

baixa cuja frequência de corte deve ser ligeiramente maior que a frequência do bit e inferior a frequência da portadora.

Escolheu-se uma sequência binária simples composta por 1 e 0. O motivo para esta escolha é que nesta modulação a largura de banda do sinal modulado acaba sendo significantemente maior do que as das anteriores, visto que o sinal modulado pode assumir dois valores de frequências diferentes, ao invés de apenas um, como era anteriormente. A consequência disso é que, através do teorema de amostragem passa-faixa, a frequência de amostragem que será definida é dependente da banda de passagem do sinal modulado e por causa do que acontece ela também será maior. Uma frequência de amostragem maior acarreta em um maior número de amostras para a variável correspondente ao sinal modulado e consequentemente também para o sinal PWM resultante. Entretanto, utilizando uma amostra do sinal PCM com 8 bits, o número de amostras que ficou para o sinal PWM foi descomunal, o que causou o travamento do programa e do computador. Diante deste problema, optou por utilizar uma amostra de entrada com apenas 2 bits, 1 e 0, de forma a poder ver na prática a modulação BFSK na transição entre os bits, ou seja, poder observar que o sinal modulado muda a sua frequência quanto muda o valor do bit.

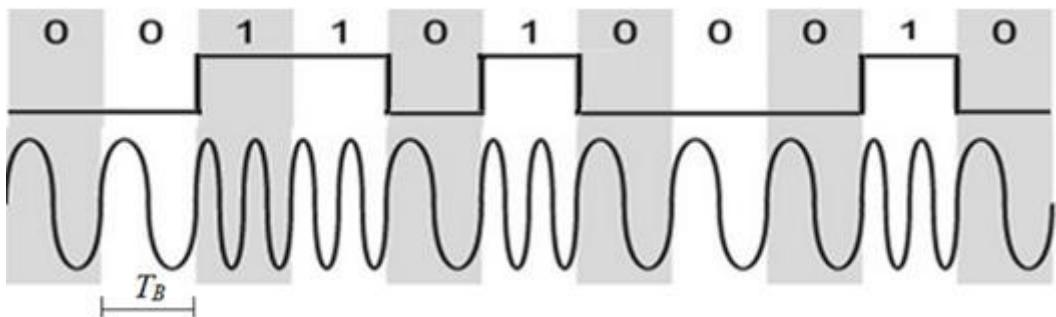
A técnica de modulação BFSK terá o seguinte funcionamento: quando o bit for 1 a resposta modulada será:

$$m(t) = 1 * \cos(2\pi f_{c1} t)$$

Onde  $f_{c1} = 9[\text{MHz}]$ . E quando o bit for 0 a resposta modulada será:

$$m(t) = 1 * \cos(2\pi f_{c2} t)$$

Onde  $f_{c2} = 11[\text{MHz}]$ . Um exemplo desta modulação é mostrado na figura abaixo:

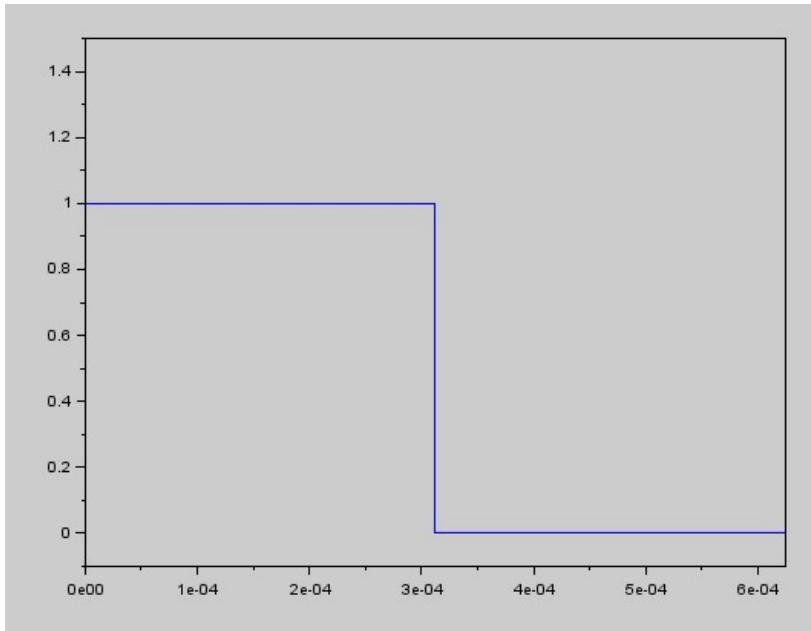


A sequência binária foi construída no Scilab:

```

1 x = [1 0];
2 bp = 1 / (400*8);
3 bit = [];
4
5 for (n=1:1:length(x))
6   if (x(n)==1)
7     se = ones(1,1000);
8   else x(n)==0
9     se = zeros(1,1000);
10  end
11  bit = [bit se];
12 end

```



Pode-se ver a sequência binária claramente. Sabe-se de informações sobre ela que cada bit vai possuir um período igual a:

$$\begin{aligned}
período_{bit} &= \frac{1}{f_{bit}} = \frac{1}{(número_{amostrasPCM} * número_{bits})} = \frac{1}{(400 * 8)} \\
&= 0,0003125s.
\end{aligned}$$

Desta forma, como há 2 bits na sequência binária, o tempo de duração total desta será de:

$$0,0003125[s] * 2 = 0,000625[s]$$

Depois de construir a sequência binária, já é possível obter o sinal modulado BFSK.

Para obter o sinal PWM do sinal BFSK, é necessário primeiramente fazer a amostragem deste sinal modulado, isto é, fazer a amostragem dos cossenos resultantes do sinal BFSK através da frequência de amostragem determinada pelo teorema de amostragem passa faixa. Para descobrir qual é esse valor de frequência de amostragem, é necessário primeiramente saber o valor da banda ocupada pelo sinal modulado com a sequência binária utilizada, pois ela é um dos fatores pelo qual é calculada a frequência de amostragem resultante.

Para determinar então a banda ocupada do sinal BFSK será necessário obter o espectro de magnitude deste sinal. Para fazer isso se aplicará a lógica da modulação BFSK normalmente para cada bit da sequência binária, onde o sinal modulado total m será resultado da concatenação da aplicação da modulação para cada bit.

Neste caso, escolheu-se uma frequência de amostragem grande o suficiente para fazer a amostragem de forma bem sucedida, sabendo que a maior frequência da portadora é de 11[MHz]. Isso é necessário pois se deseja ver o espectro de magnitude da maneira mais limpa possível, evitando perdas de informação que decorreriam de uma má escolha de frequência de amostragem. Desta forma, aplicando neste caso o teorema de Nyquist, deve-se escolher uma frequência de amostragem de ao menos 22[MHz]. Entretanto, mesmo que suficiente, ela ainda é relativamente baixa, visto que se quer uma representação fiel aos cossenos. Com isso, sabendo que o período do bit é igual a 0,0003125[s], escolheu-se uma frequência de amostragem de acordo com a seguinte fórmula:

$$f_s = \frac{100000}{p_b} = \frac{100000}{0,0003125} = 320MHz$$

Um valor 29 vezes maior que o maior da portadora, o que gera uma resposta satisfatória.

**Observação:** Essa não será a frequência de amostragem final para a modulação. Esta será determinada depois da aplicação do teorema de amostragem passa-faixa. A provisória serve apenas para se ter uma boa visualização de como é o sinal modulado e o seu espectro de amplitude.

```

19 t2 = 0:bp/100000:bp - bp/100000;
20 m = [];
21
22 for (i=1:1:length(x))
23     if (x(i)==1)
24         y = 1*cos(2*pi*9e6*t2);
25     else
26         y = 0*cos(2*pi*11e6*t2);
27     end
28     m = [m y];
29 end
30
31 t3 = 0:bp/(100000):(bp*length(x))-bp/100000;
32 plot(t3,m)

```

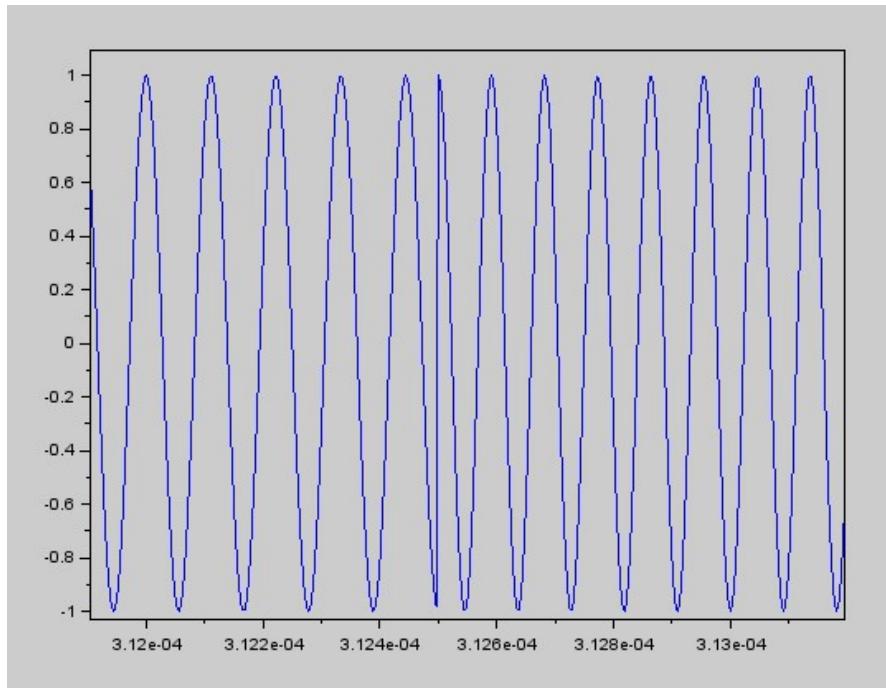
Como período do bit é de:

$$período_{bit} = \frac{1}{f_{bit}} = \frac{1}{(número_{amostrasPCM} * número_{bits})} = \frac{1}{(400 * 8)} = 0,0003125$$

E há uma transição clara de 1 para 0 do primeiro bit para o segundo, tem-se que haverá uma transição nos cossenos no tempo:

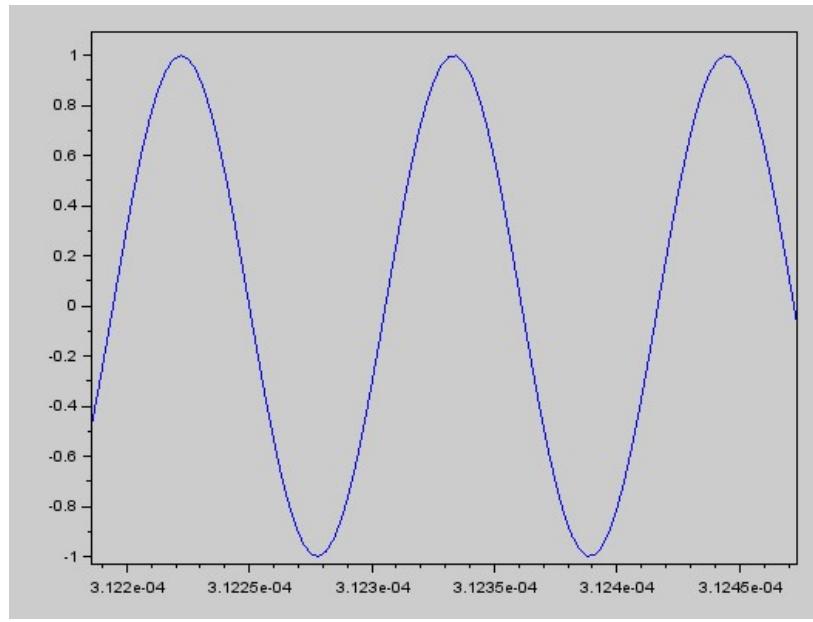
$$1 * período_{bit} = 1 * 0,0003125 = 0,0003125[\text{s}]$$

Ampliando a transição de 1 para 0, temos:



Pode-se ver claramente os cossenos, e sua transição. É notável e claro que a frequência dos cossenos muda após a transição, já que as ondas se tornam-se mais “curtas”, o que indica que a frequência se tornou maior, o que realmente acontece. Isso pode ser provado pegando um cosseno em algum instante antes da transição e um depois e medir as frequências deles.

Pegando primeiramente um cosseno antes da transição em 0,0003125[s], tem-se:



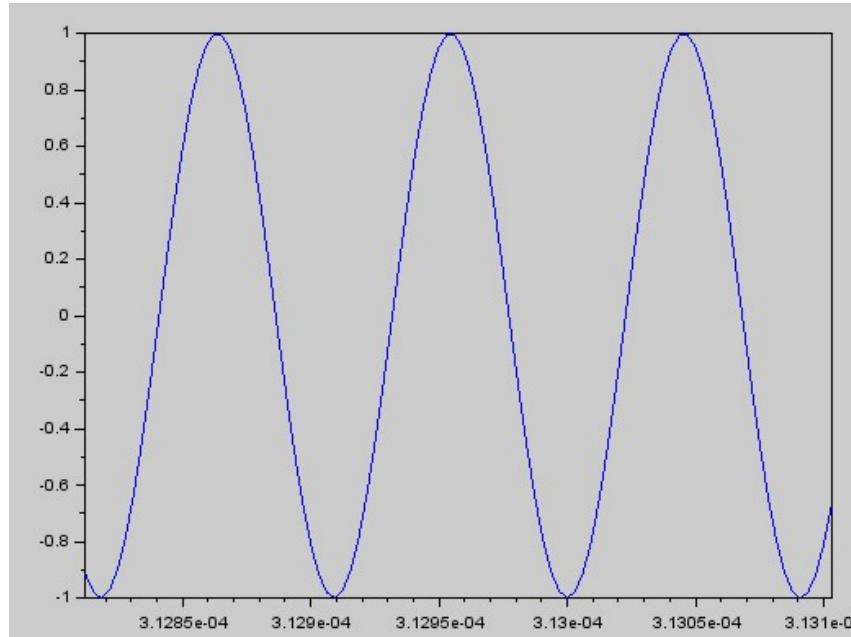
Para fazer o cálculo, se pegará as menores amplitudes do sinal em volta do instante  $3,1235 \times 10^{-4} [\text{s}]$ , que acontecem aproximadamente em  $3,1228 \times 10^{-4} [\text{s}]$  e  $3,1239 \times 10^{-4} [\text{s}]$ .

Logo a frequência do cosseno é:

$$\frac{1}{(3,1239 \times 10^{-4} - 3,1228 \times 10^{-4})} = 9,09 \text{ MHz}$$

Valor muito próximo ao real, que só não foi igual pois pegou-se valores aproximados para os pontos no cálculo, adicionando assim uma parcela de erro.

Pegando agora um cosseno depois da transição em  $0,0003125 [\text{s}]$ , tem-se:



Para fazer o cálculo, se pegará as menores amplitudes do sinal em volta do instante

$3,1295 * 10^{-4}[\text{s}]$ , que acontecem aproximadamente em  $3,1291 * 10^{-4}[\text{s}]$  e  $3,13 * 10^{-4}[\text{s}]$ .

Logo a frequência do cosseno é:

$$\frac{1}{(3,13 \cdot 10^{-4} - 3,1291 \cdot 10^{-4})} = 11,11 \text{MHz}$$

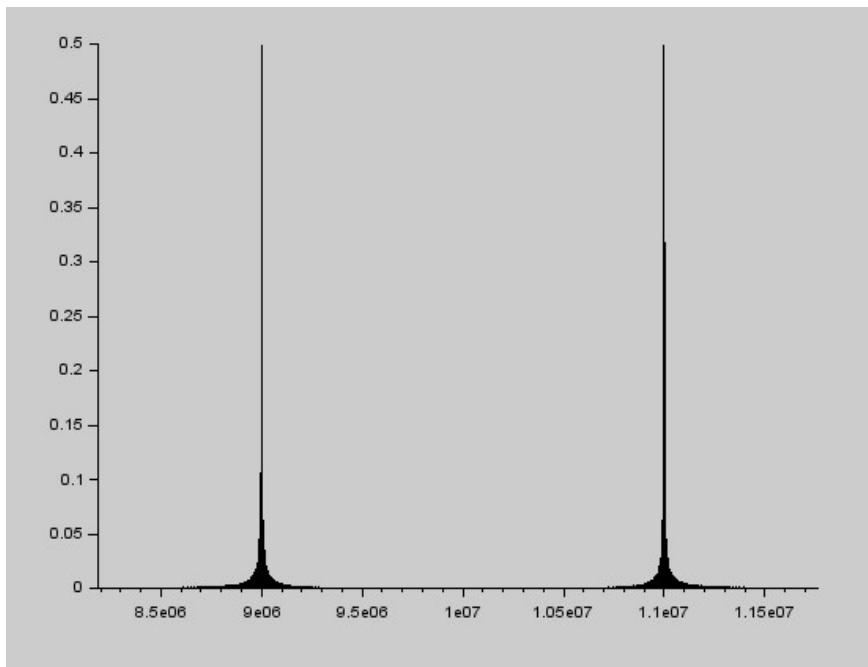
Valor muito próximo ao real, que só não foi igual novamente pois se escolheu valores aproximados para os pontos pegos no cálculo, adicionando desta forma uma parcela de erro. Ainda assim, pode-se notar agora com provas, que a frequência do sinal modulado muda de 9[MHz] para 11[MHz] à medida que o bit da amostra vai de 1 para 0, exatamente o que se desejava.

Para obter o espectro de amplitude, onde tem-se as magnitudes do sinal em função das frequências, tem-se que o vetor de amplitudes é dado pelo módulo da aplicação da fft sobre o sinal modulado m completo multiplicado por 2 e dividido pelo número de amostras do próprio sinal, enquanto que o vetor de frequência é construído com base nos múltiplos da frequência fundamental que é definida como o inverso do período de observação do sinal, e vai até o número de amostras do sinal menos um. O resultado obtido é:

```

35 N = length(m);
36 Amp = (2*abs(fft(m))/N);
37 f = 0:1/(bp*length(x)):(N-1)*1/(bp*length(x));
38 plot2d3(f,Amp);

```



Como foi explicado na etapa 4, a frequência mínima não parte do zero, mas fica centrada nas frequências de 9[MHz] e 11[MHz], frequências da portadora, e ocupa uma certa banda. Então para a escolha da frequência de amostragem, se usará o teorema de amostragem passa faixa, que permite uma escolha de frequência menor que a escolhida

pelo teorema de Nyquist, e não irá gerar sobreposição no espectro e distorção na forma de onda.

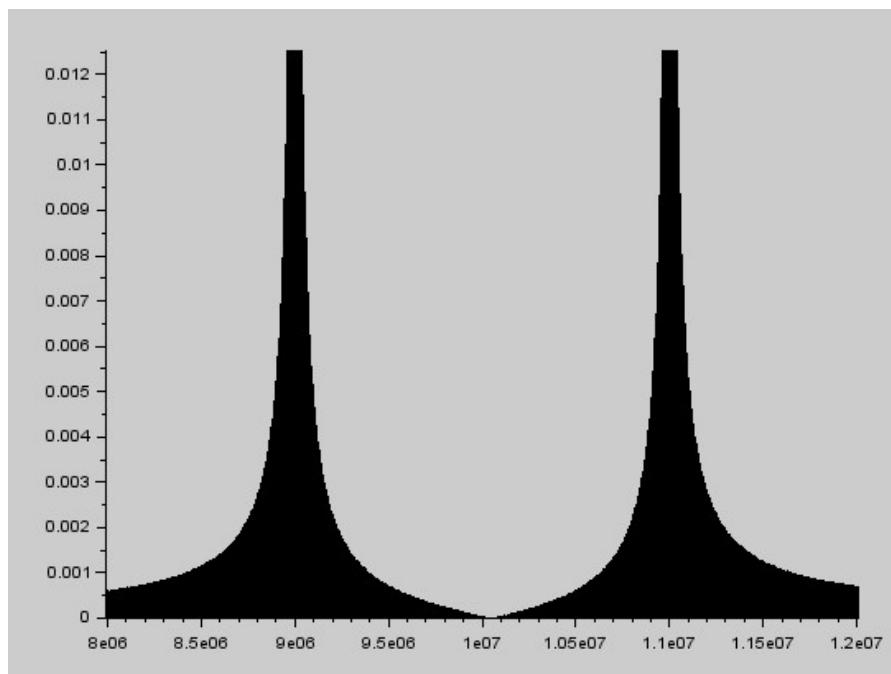
Sabe-se do teorema de amostragem passa-faixa, que é possível recuperar um sinal  $m(t)$ , se ele for amostrado com  $f_s$ :

$$f_s = \frac{2 * f_{max}}{k}$$

Onde  $k$  é:

$$k = \frac{f_{max}}{B}$$

Para se determinar o valor de  $f_{max}$ , assim como foi feito na etapa 6, usou-se como referência a amplitude média de 0,1 do diagrama de amplitude.  $f_{max}$  foi determinado então como sendo a frequência em que se tem uma amplitude 100 vezes menor que 0,1, isto é, a frequência máxima onde se tem uma amplitude não desprezível. Esse valor é obtido dando um zoom mais aprofundado no espectro de magnitude:



Ou seja, tem-se uma amplitude de:

$$\frac{0,1}{100} = 0,001$$

Em aproximadamente  $11,6\text{MHz}/8,4[\text{MHz}]$ . Com isso, a frequência máxima do sinal é igual a  $11,6[\text{MHz}]$  de acordo com o que se foi proposto. Usando a mesma lógica, a banda ocupada pelo sinal, contando as amplitudes não desprezíveis, pode ser definida como começando a partir de  $8,4[\text{MHz}]$  e terminando em  $11,6[\text{MHz}]$ , resultando num valor de  $(11,6 - 8,4) * 10^6 = 3,2[\text{MHz}]$ .

Entretanto, é recomendável estipular um valor de banda ocupada igual a 20 vezes a banda que realmente o sinal modulado ocupa, a fim de acomodar com sucesso a transição do filtro e também para reduzir a distorção gerada no PWM. Logo, tem-se que k é:

$$k = \frac{f_{max}}{20 * B_{real}} = \frac{11,6 * 10^6}{20 * 3,2 * 10^6} = 0,18125$$

E  $f_s$ :

$$f_s = \frac{2 * f_{max}}{k} = \frac{2 * 11,6 * 10^6}{0,18125} = 128MHz$$

Esse valor, entretanto, não é múltiplo das frequências da portadora, logo o escolher não seria uma boa prática. Por outro lado, a frequência de 99[MHz] que é um múltiplo tanto de 11[MHz] e 9[MHz] possui um valor demasiadamente menor que o calculado para a frequência de amostragem. Logo, nesta balança, preferiu por pegar um múltiplo do valor médio das frequências da portadora que seja próximo de 128[MHz], ou seja, 130[MHz].

$$f_{sESCOLHIDO} = 130[MHz]$$

Onde se terão 14 amostras por período da portadora no novo sinal modulado BFSK para quando o bit for 1 e onze amostras por período da portadora para quando o bit for 0.

Analizando este valor, pode-se perceber o quanto foi reduzida a frequência de amostragem necessária para uma recuperação bem sucedida do sinal modulado utilizando o teorema de amostragem passa-faixa, se comparado com o que seria necessário caso se usasse o teorema de Nyquist, o que levaria a um valor, mirando o mínimo de perda de informação, de vinte vezes a maior frequência da portadora, ou seja, algo em torno de 220[MHz].

Agora com a nova frequência de amostragem, constrói-se um novo sinal modulado BFSK.

Será com base nesse novo sinal modulado que se obterá o PWM.

```

41 fs = 13e7;
42 ts = 1/fs;
43 t2 = 0:ts:bp-ts;
44 m = [];
45 for(i=1:1:length(x))
46     if(x(i)==1)
47         y = 1*cos(2*pi*9e6*t2);
48     else
49         y = 1*cos(2*pi*11e6*t2);
50     end
51 m = [m y];
52 end

```

Diferente do sinal BASK da etapa 4, o sinal BFSK não tem valores entre 0 e 1, mas sim entre -1 e 1, logo é necessário dar um offset no sinal modulado. Além disso, com o novo sinal modulado BFSK m já determinado é necessário normalizá-lo, isto é, como foram usados 8 bits de quantização para determinar o vetor de amostras de PCM, o maior valor que uma amostra pode assumir é de 255. Como no sinal modulado BFSK o valor máximo assumido pelo cosseno é de 2, devido ao offset aplicado no sinal, será necessário multiplicá-lo por 255/2. Por fim, para não obter valores quebrados será necessário arredondá-los também através do comando round().

```

54 m = m + 1;
55 m = m*(255/2);
56 m = round(m);

```

Fazendo um estudo mais aprofundado sobre como foi construído o sinal modulado BFSK percebe-se que para cada bit, tem-se um número de ciclo de portadora igual a:

$$p_b * f_s = \frac{1}{(400 * 8)} * 130 * 10^6 = 40625$$

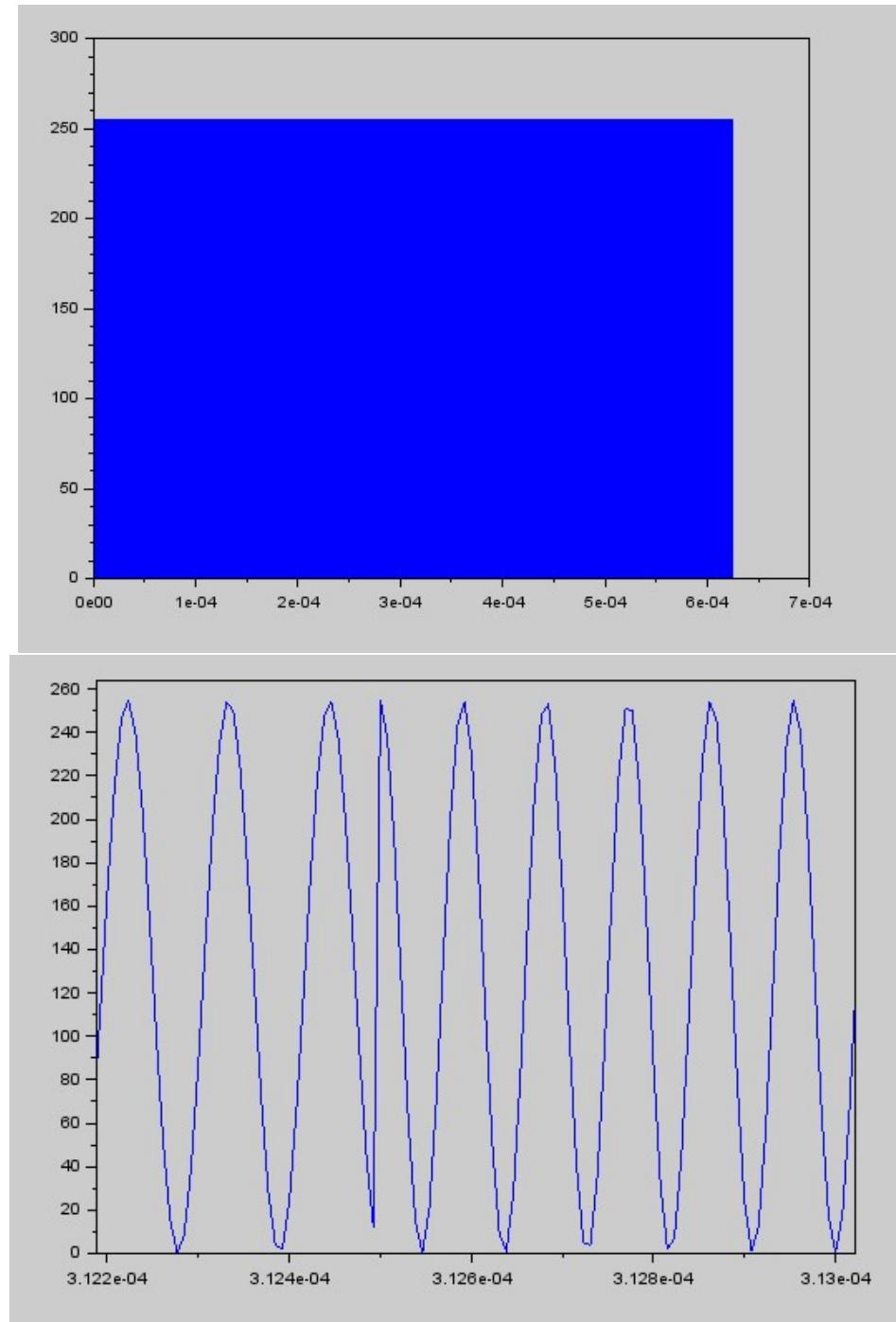
Como há 2 bits na sequência binária, o número total de amostras do sinal modulado é igual a  $40625 * 2 = 81250$  amostras.

Pode-se plotar o sinal modulado normalizado resultante, especialmente na região de transição do bit 1 para 0, já identificada anteriormente no momento de 0,0003125[s], da seguinte forma:

```

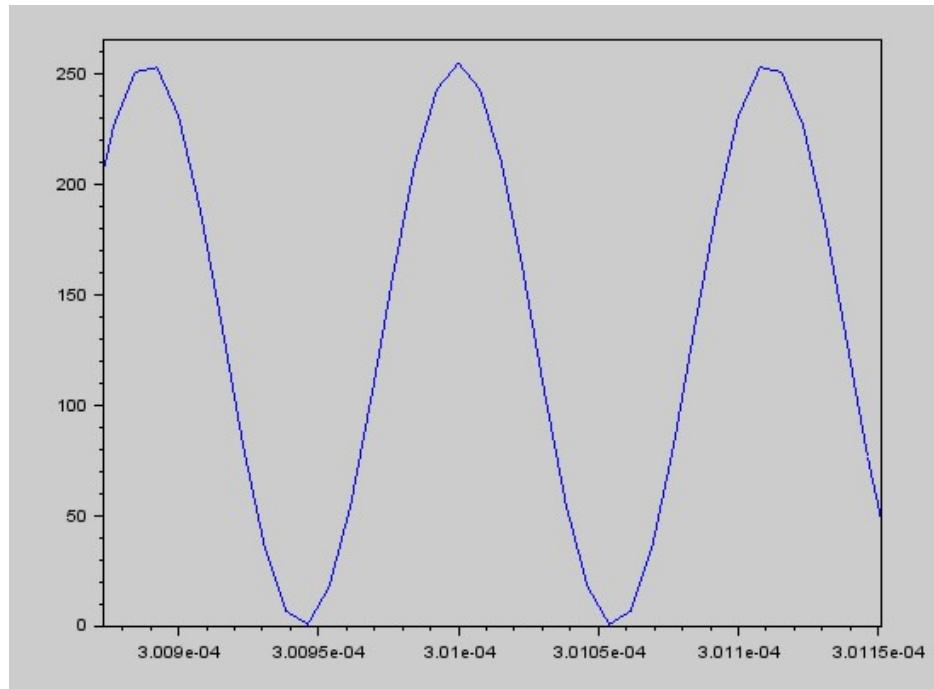
58 t3 = 0:ts:(bp*length(x))
59 plot(t3,m)

```



Onde percebe-se que a onda modulada manteve seu aspecto cossenoidal, assim como a mesma característica de mudança de frequência conforme a transição de bits com a modulação BFSK. Pode-se replicar o que foi feito anteriormente a fim de se calcular a frequência das ondas cossenoidais antes e depois do momento de transição dos bits.

Pegando primeiramente um cosseno antes da transição em 0,0003125[s], tem-se:

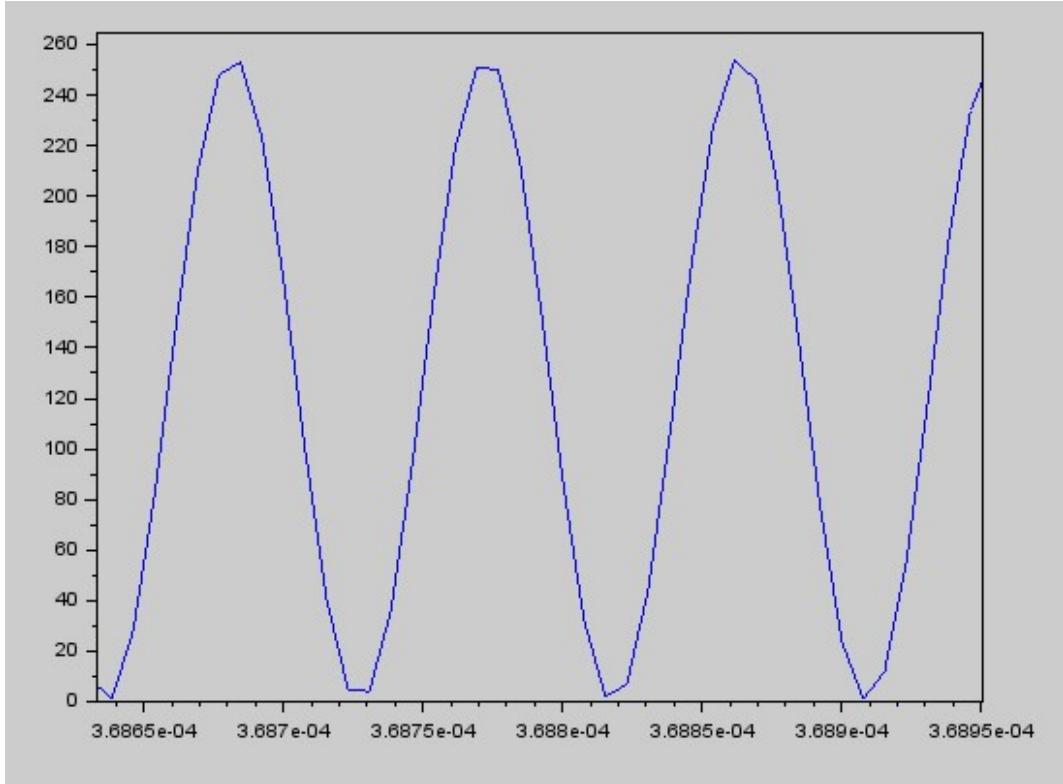


Para fazer o cálculo, se pegará as menores amplitudes do sinal em volta do instante  $3,01 \times 10^{-4}[\text{s}]$ , que acontecem aproximadamente em  $3,00945 \times 10^{-4}[\text{s}]$  e  $3,01055 \times 10^{-4}[\text{s}]$ . Logo a frequência do cosseno é:

$$\frac{1}{(3,01055 \times 10^{-4} - 3,00945 \times 10^{-4})} = 9,09 \text{ MHz}$$

Valor muito próximo ao real, que só não foi igual pois pegou-se valores aproximados para os pontos no cálculo, adicionando assim uma parcela de erro.

Pegando agora um cosseno depois da transição em  $0,0003125[\text{s}]$ , tem-se:



Para fazer o cálculo, se pegará as menores amplitudes do sinal em volta do instante

$3,6885 * 10^{-4}s$ , que acontecem aproximadamente em  $3,68815 * 10^{-4}s$  e  $3,68905 * 10^{-4}s$ . Logo a frequência do cosseno é:  $\frac{1}{(3,68905*10^{-4}-3,68815*10^{-4})} = 11,11MHz$ , valor muito próximo ao real, que só não foi igual novamente pois se escolheu valores aproximados para os pontos pegas no cálculo, adicionando desta forma uma parcela de erro. Ainda assim, pode-se notar agora com provas, que a frequência do sinal modulado muda de 9MHz para 11MHz à medida que o bit da amostra vai de 1 para 0, exatamente o que se desejava.

Com a onda modulada BFSK já normalizada e definida, já é possível obter o PWM desta. Primeiramente, fará esse processo na FPGA através do Quartus, onde os valores das amostras do sinal modulado passados a ele devem estar na forma binária. Essa conversão de decimal para binário é feita utilizando o comando dec2bin () .

```
|--> mbin = dec2bin(m);
```

A geração do PWM é feita através da comparação do valor absoluto de cada amostra da onda modulada com uma onda dente de serra. A saída funcionará da seguinte forma: enquanto a onda dente de serra for menor que o valor da amostra da onda modulada a saída terá nível lógico alto e quando não for menor a saída terá nível lógico baixo, produzindo assim uma modulação em largura de pulso.

A geração do PWM na FPGA será realizada comparando a amostra e o valor em um contador incrementado por um clock, levando em conta que o contador deverá resetar para zero quando ultrapassar seu valor máximo, significando que se deve atualizar a amostra utilizando a próxima em seguida.

Para definir o valor da frequência do clock que servirá para incrementar o contador se usará da seguinte relação desta variável com a frequência de amostragem do sinal modulado BFSK, que por sua vez é igual a 130 MHz.

$$f_s = \frac{f_{clk}}{c_{max} + 1}$$

Onde  $c_{max}$  é o valor máximo do contador, cujo valor deve respeitar a seguinte inequação:  $c_{max} > 2^{N+1}$ , em que N é o número de bits de quantização utilizado. Esta inequação existe para garantir que o sinal PWM gerado não possua nunca um duty cycle superior a 50% da frequência de amostragem. Como neste trabalho utilizou-se 8 bits de quantização,

tem-se que:  $c_{max} > 2^{8+1} \Rightarrow c_{max} > 2^9 \Rightarrow c_{max} > 512$ . A fim de obter um valor da frequência do clock redondo, escolheu-se um valor de  $c_{max}$  igual a 549 que satisfaz a inequação. Desta forma, tem-se que:

$$f_s = \frac{f_{clk}}{c_{max} + 1} \Rightarrow f_{clk} = f_s(c_{max} + 1) = 130 * 10^6(549 + 1) \Rightarrow f_{clk} = 71500MHz$$

Com todos os parâmetros definidos, já é possível começar a desenvolver a lógica em Verilog para replicar o circuito desejado para gerar o PWM. A fim de evitar falhas e comprovar o funcionamento do módulo geral decidiu-se por programar um módulo para cada “componente” do circuito completo, isto é, programou-se um módulo separado para

o contador, a amostragem e o comparador. Por fim, juntou-se ambos para o módulo geral do PWM.

Primeiramente, foi criado o módulo das amostras, dado por:

```
1 module amostras(input clk, input [16:0] A, output reg [11:0] amostra);
2
3     reg [7:0] mem [0:81249];
4
5     initial begin
6         $readmemb("../m.txt", mem);
7     end
8
9     always @ (A) begin
10        amostra = mem[A];
11    end
12 endmodule
```

Ele basicamente cria um vetor 81250x8 para armazenar os valores e então preenche cada uma dessas posições com as amostras do sinal modulado BFSK normalizado através de um arquivo txt. Além disso, o valor de saída deste módulo vai ser controlado pelo valor de entrada A, onde a saída será dada pelo valor da posição A da memória. Por exemplo, se o valor de A é 1, a saída amostra será o valor da memória na posição 1.

Já o módulo do contador é dado por:

```
1 module contador(input clk, output reg [11:0] cont, output reg [16:0] A);
2
3     initial cont = 12'd0;
4     initial A = 17'd0;
5
6     always @ (posedge clk) begin
7         cont <= cont + 12'd1;
8         if(cont >= 12'd549) begin
9             cont <= 12'd0;
10            A <= A + 17'd1;
11            if(A == 17'd81249) A <= 17'd0;
12        end
13    end
14 endmodule
```

Como foi esclarecido anteriormente, o contador é incrementado com o clock, de 1 dígito e possui valor máximo de 549 e assim que ele passa esse valor ele volta a valer 0 e recomeça todo o processo. Além disso, o contador também deve servir para indicar qual posição da memória, isto é, qual amostra, deve ser comparada para gerar o PWM. Ou seja, quando o contador reinicia deve-se atualizar a amostra utilizando a próxima. Isso é feito incrementando 1 na variável de controle da posição da amostra toda vez que o valor do contador é extrapolado. Como há no total 81250 amostras, esse é processo é realizado até chegar neste valor, que então faz com que o valor da posição da amostra volte a 0, reiniciando assim o ciclo.

O módulo de comparação é dado por:

```
module comparador(input [11:0] amostra, input [11:0] cont, output reg pwm);
    always @* begin
        if(amostra > cont)    pwm = 1;
        else pwm = 0;
    end
endmodule
```

Este módulo é bem simples. Ele apenas pega o valor das entradas referentes à amostra atual e do contador, e se o valor absoluto da amostra é maior do que o contador a saída pwm é igual a 1(nível lógico alto), se não, a saída pwm é 0 (nível lógico baixo).

A interconexão desses três módulos é realizada no módulo principal, dado por:

```
module pwm(input clock, output saída);
    (*keep=1*) wire [11:0] cont;
    (*keep=1*) wire [11:0] amostra;
    (*keep=1*) wire [16:0] A;
    contador C(clock, cont, A);
    amostras Amos(clock, A, amostra);
    comparador comp(amostra, cont, saída);
endmodule
```

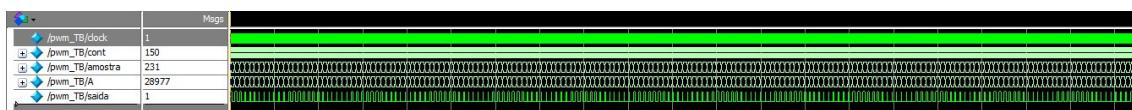
Onde a expressão (\*keep=1\*) é feita meramente para a visualização dessas variáveis na simulação. O módulo é conectado da seguinte forma: a saída do módulo contador A vai para a entrada do módulo amostras para indicar qual posição da memória e consequentemente qual amostra deverá ser comparada. Então essa amostra e a saída cont do módulo contador vão para a entrada do módulo comparador onde são comparadas a fim de gerar o sinal PWM.

O código de simulação é mostrado abaixo, onde se implementa um clock com período de 13,986ps, o que dá uma frequência do clock de aproximadamente 71500MHz:

```
1 `timescale 1ps/10fs
2
3 module pwm_TB;
4     reg clock;
5     reg [11:0] cont;
6     reg [11:0] amostra;
7     reg [16:0] A;
8     wire saída;
9
10    DUT(
11        .clock(clock),
12        .saída(saída)
13    );
14
15    initial begin
16        clock = 0;
17    end
18
19    always begin
20        #6.993 clock = ~clock;
21    end
22
23    initial begin
24        $init_signal_spy("/pwm_TB/DUT/cont","cont",1);
25        $init_signal_spy("/pwm_TB/DUT/amostra","amostra",1);
26        $init_signal_spy("/pwm_TB/DUT/A","A",1);
27    end
28
29    initial
30        #3000000 $stop;
31
```

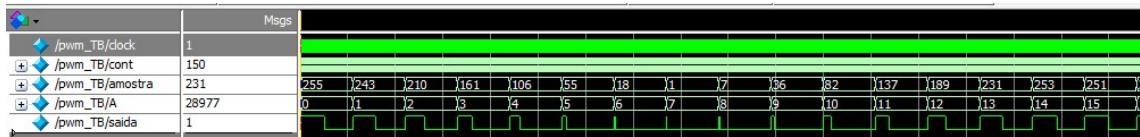
O resultado da simulação é mostrado a seguir:

O resultado da simulação é mostrado a seguir:



Como a visão original não dá para analisar praticamente nada, graças ao tamanho relativamente grande do vetor de saída PWM, será necessário dar um zoom para poder fazer uma análise mais qualificada.

Desta forma, dando um zoom a fim de visualizar as primeiras 15 amostras do sinal modulado BFSK, percebe-se que:

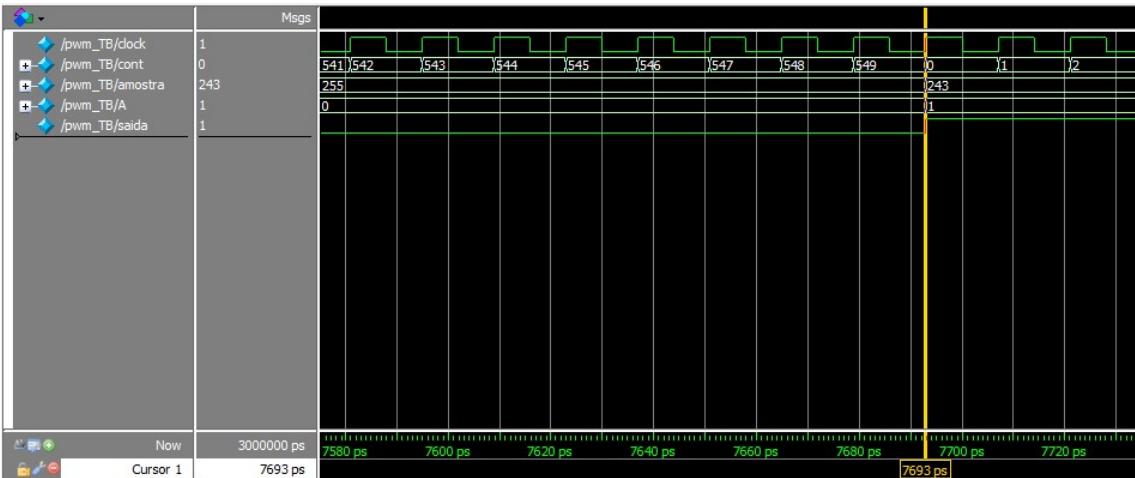


Pode-se ver então que quanto maior o valor da amostra, maior será o comprimento do estado lógico alto do PWM, justamente como era esperado.

A validade do funcionamento deste módulo pode ser verificada analisando o comportamento da onda de pwm em uma única amostra através da comparação entre a proporção do tempo em que a saída fica em nível lógico alto e o tempo total do período da onda pwm para essa amostra em específico e a proporção do valor absoluto da amostra com o valor máximo do contador igual a 549. Em teoria, eles deveriam ser iguais ou ao menos muito próximos. Escolhendo a primeira amostra para fazer esse teste, tem-se que:



Ou seja, o sinal pwm muda de nível lógico em 3563ps, logo após o momento em que o valor da amostra não é mais maior que o valor do contador. Por fim, o contador extrapola no seguinte instante:



Ou seja, em 7693ps. O valor ideal seria em  $\frac{1}{f_s} = \frac{1}{130 \text{ M}} = 7692,31$  ps, correspondente ao período de amostragem. No entanto, como se utilizou da simulação em gate level, levou-se em conta desta forma os atrasos inerentes aos componentes utilizados para realizar a simulação além também da própria resolução utilizada.

De qualquer forma, as proporções ficam da seguinte forma:

$$\frac{\text{valor da amostra}}{\text{valor máximo do contador}} = \frac{255}{549} = 0,46448$$

$$\frac{\text{tempo em nível lógica alto}}{\text{período da onda para a amostra}} = \frac{3563\text{ps}}{7693\text{ps}} = 0,46315$$

Ou seja, houve uma diferença entre as proporções de apenas 0,00133, um valor relativamente bem baixo, comprovando assim a eficácia do módulo proposto.

Agora que a implementação no Quartus já foi realizada, agora será feito algo similar no Scilab. Neste caso, se empregarão os mesmos valores das variáveis já definidos anteriormente, ainda que não se dará um valor “físico” em relação às operações realizadas para chegar ao resultado.

Sabe-se que o vetor PWM de saída terá no total ( $c_{max} + 1$ ) \* Número de amostras =  $550 * 81250 = 44687500$ ) amostras, já que cada amostra terá uma comparação com um valor do contador que vai de 0 até 549.

Agora se aplicará o algoritmo para fazer a comparação entre o valor de cada uma das amostras da onda modulada BFSK e o contador. Desta forma se usará um for de 1 a 81250 para varrer cada uma das amostras e outro for de 1 a 550 para varrer todos os valores possíveis do contador.

O motivo para não realizar o for começando de 0 para ambos, como parecia mais plausível, é porque não pode ser realizado no Scilab, pois este software não aceita empregar o índice de 0 para uma variável, já que em ambas a primeira posição do vetor é dada por 1 e não zero. Desta forma para o contador fez o incremento de 1 até 550, equivalente ao que seria o incremento de 0 até 549.

Além disso é feito a comparação entre o valor do contador e o valor da amostra no índice indicado pelo for através de um if. A única coisa que falta é poder concatenar cada operação de comparação com os valores abstraídos do pwm, afinal, quando extrapolado o valor do contador passa-se para a seguinte posição do vetor de amostras e assim começa um novo ciclo de obtenção dos valores do pwm. Para resolver isso, usa-se de uma variável auxiliar a fim de corrigir a posição do pwm, isto é, conforme o índice da posição do vetor de amostras aumenta em 1 o índice de posição do vetor de pwm deverá ser incrementado em 550, justamente o valor de extração do contador. Este algoritmo é apresentado a seguir:

```

61 for i=1:81250
62   for cont = 1:550
63     j = i-1;
64     aux = cont + 550*j;
65     if(cont < m(i))
66       PWM(aux)=1;
67     else
68       PWM(aux)=0;
69     end
70   end
71 end

```

Para verificar a funcionalidade deste código, plota-se o gráfico de t em função do valor do vetor PWM, onde t é o vetor temporal que vai de 0 a 0,000625s (tempo total da sequência binária) incrementado pelo período do clock igual ao inverso de 71500 MHz. Desta forma, tem-se:

```

73 fclk = fs*550;
74 tclk = 1/fclk;
75 t = 0:tclk:(bp*length(x))-tclk;

```

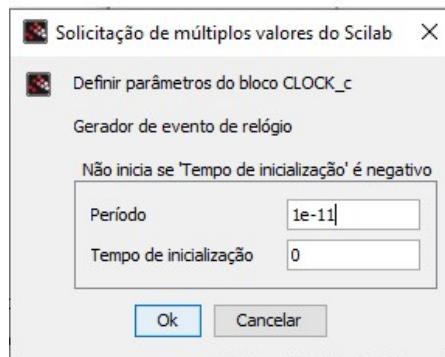
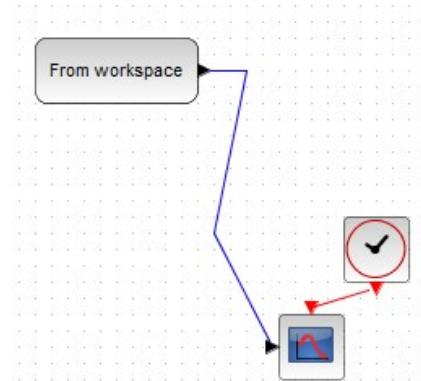
Para fazer a plotagem do PWM em função do tempo usou-se o Xcos, onde os valores do console foram passados a ele através do comando struct().

```

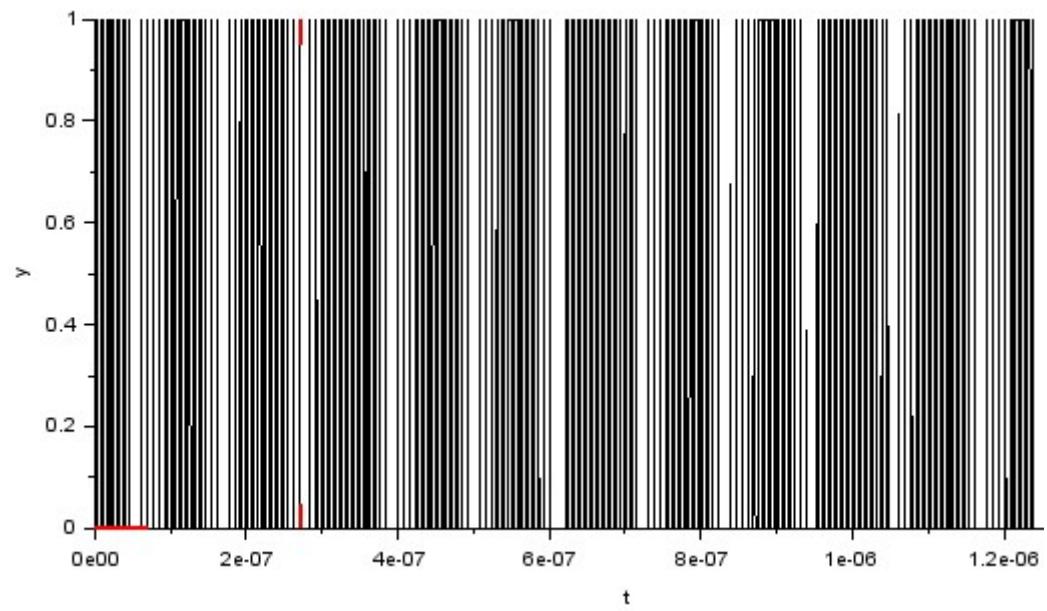
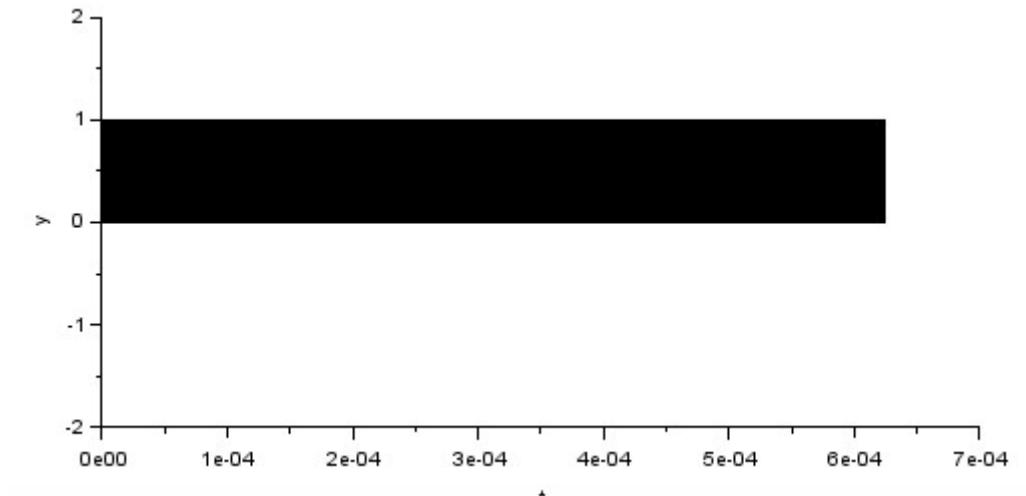
77 PWM = PWM';
78 v = struct('time',t,'values',PWM')

```

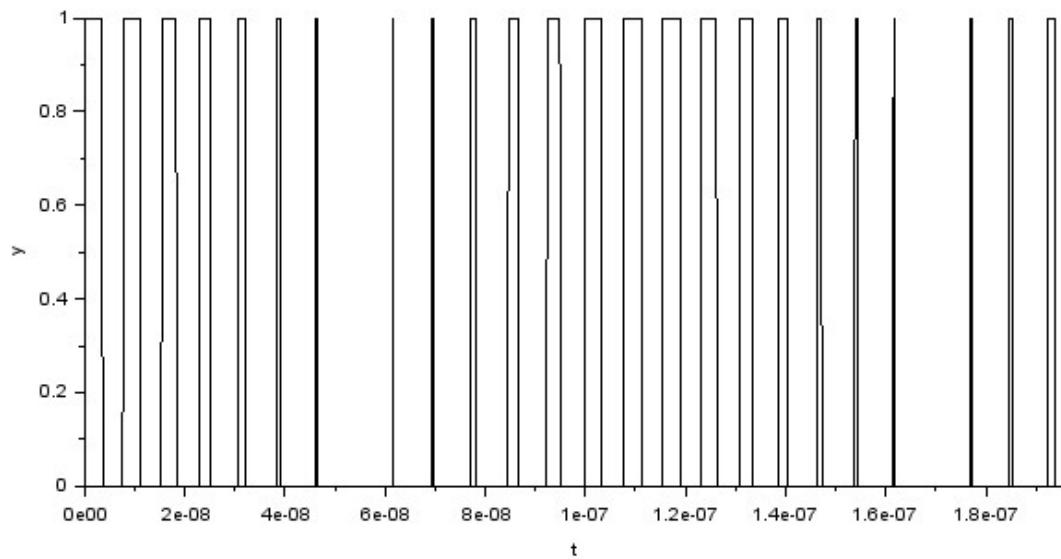
O diagrama de blocos no Xcos fica da seguinte forma:



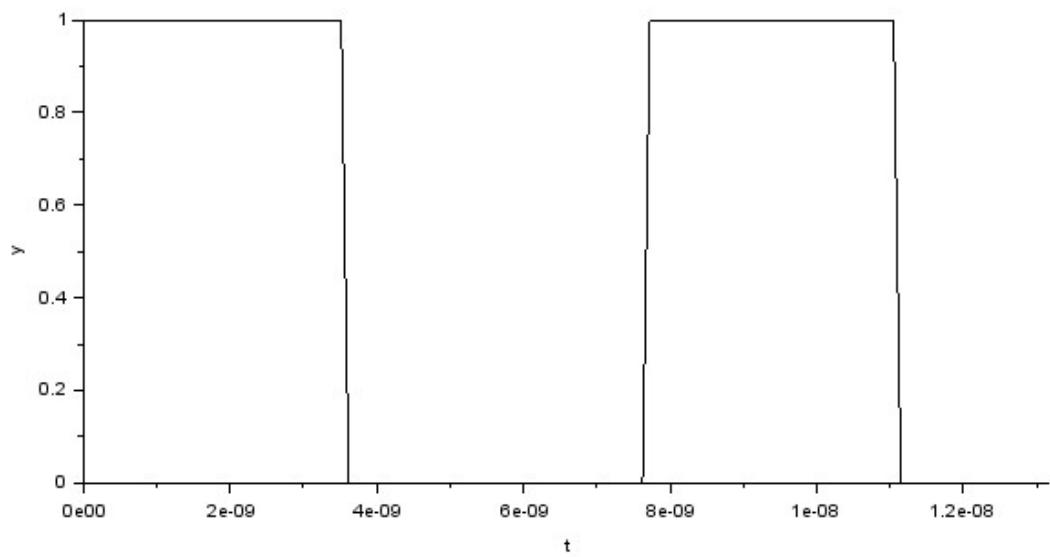
Onde o período do clock foi determinado como sendo igual a  $1 * 10^{-11}$ , valor inferior ao período do PWM, igual a:  $\frac{1}{f_{clk}} = \frac{1}{(130*10^6*550)} = 1,40 * 10^{-11}$ . Com isso, se poderá ter uma representação fiel da onda resultante, cujo resultado da simulação no osciloscópio, com tempo total de 0,000625s, é mostrado abaixo:

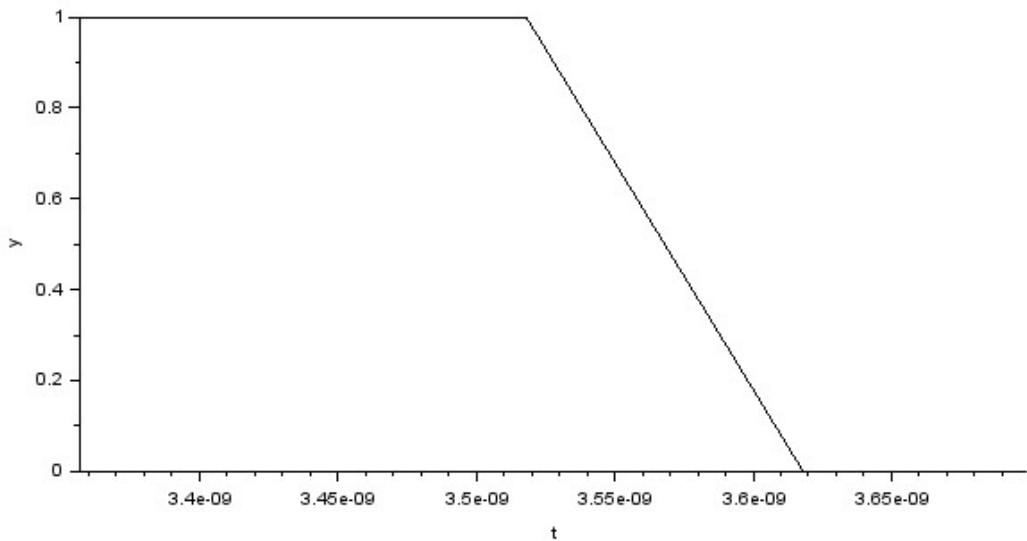


Percebe-se o comportamento geral da onda PWM, em que a sua largura será maior de acordo com o tamanho do valor da amostra. Pode-se dar um zoom para ter uma verificação mais clara deste comportamento.



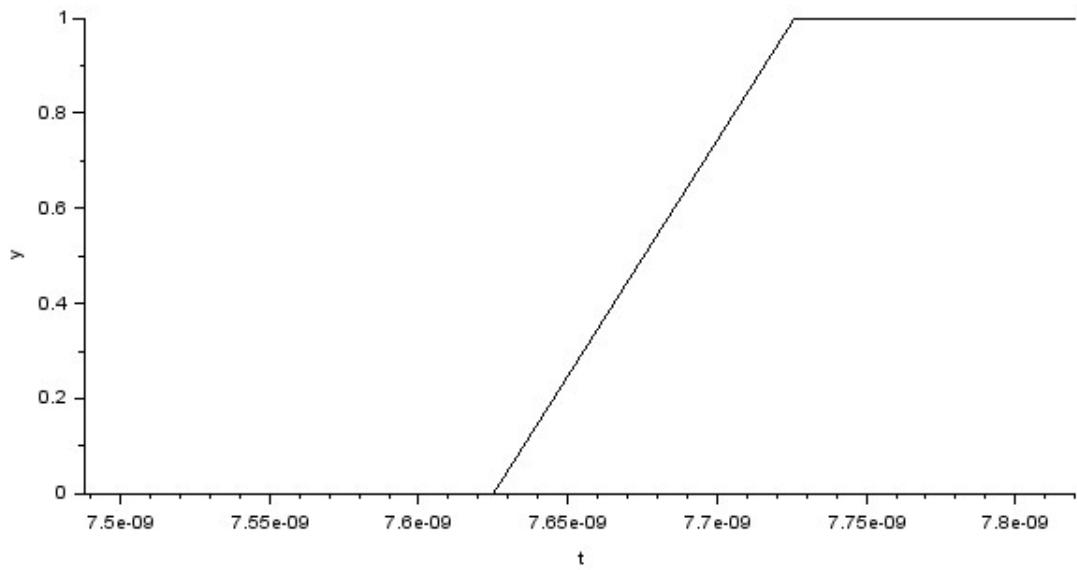
Pode-se aplicar novamente a comparação entre as proporções de tempo e valores já explicada anteriormente. Escolhendo mais uma vez como exemplo a primeira amostra, tem-se que o momento em que é mudado seu nível lógico alto para baixo no pwm é:





Aproximadamente  $3,618 \times 10^{-9}$ s.

Já o contador extrapola no seguinte estante:



Aproximadamente,  $7,625 \times 10^{-9}$ s.

Assim, as proporções ficam da seguinte forma:

$$\frac{\text{valor da amostra}}{\text{valor máximo do contador}} = \frac{255}{550} = 0,46364$$

$$\frac{\text{tempo em nível lógica alta}}{\text{período da onda para a amostra}} = \frac{3,618 \times 10^{-9}}{7,625 \times 10^{-9}} = 0,47449$$

Logo, pode-se constatar uma diferença entre as proporções de 0,0108, um valor pequeno comprovando também assim a eficácia do algoritmo. Nota-se também que o pwm implementado no Scilab é compatível com o implementado no Quartus.

