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.

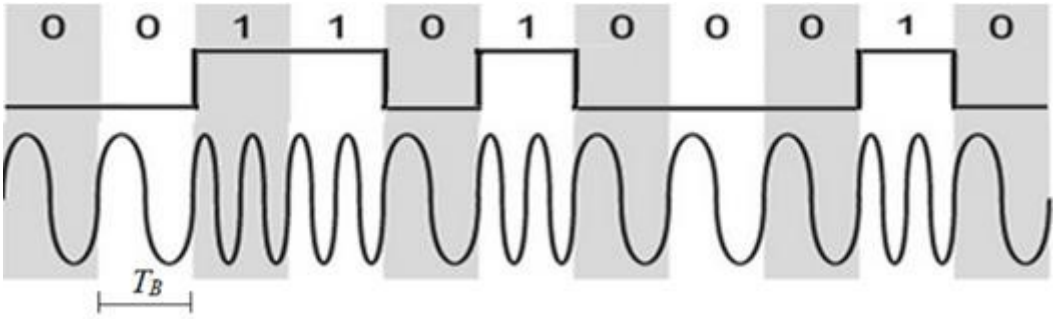
A técnica de modulação BFSK terá o seguinte funcionamento: quando o bit for 1 a resposta modulada será:

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

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

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

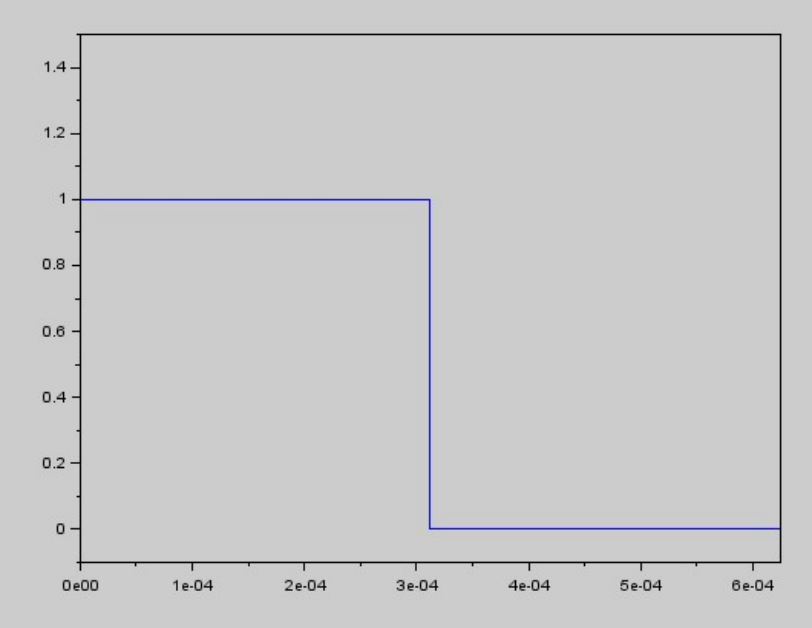
Onde f_{c2} = 11MHz. Um exemplo desta modulação é mostrado na figura abaixo:



A sequência binária foi construída no Scilab:

```
x = [1 0];
bp = 1/(400*8);
bit = [];

for (n=1:1:length(x))
    if(x(n)==1)
        se = ones(1,1000);
    else x(n)==0
        se = zeros(1,1000);
    end
    bit = [bit se];
end
```



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

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

$$= 0{,}0003125s.$$

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.

```
t2 = 0:bp/100000:bp - bp/100000;
m = [];

for (i=1:1:length(x))
    if(x(i)==1)
        y = 1*cos(2*%pi*9e6*t2);
    else
        y = 0*cos(2*%pi*11e6*t2);
    end
    m = [m y];
end

t3 = 0:bp/(100000):(bp*length(x))-bp/100000;
plot(t3,m)
```

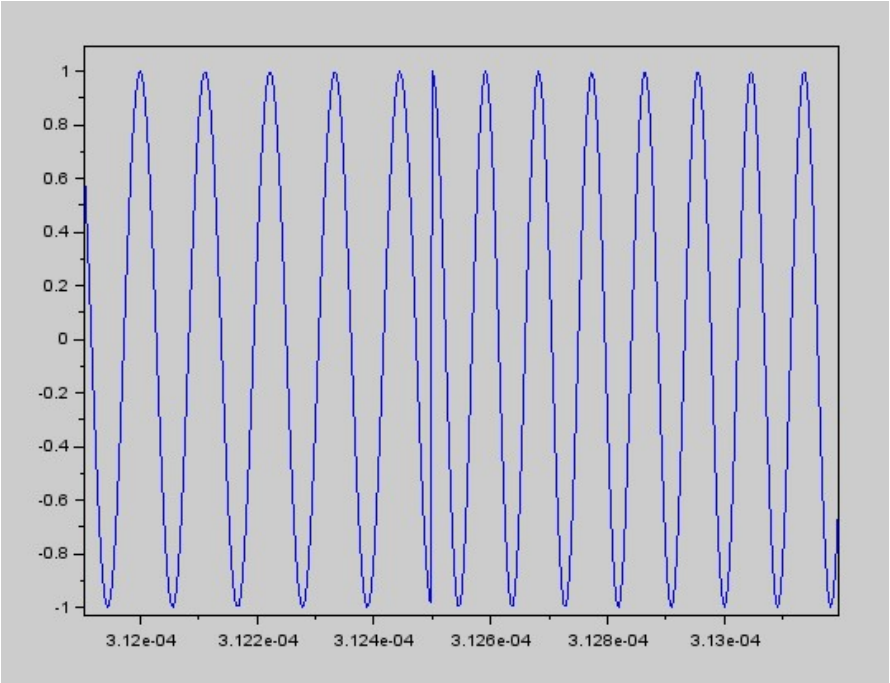
Como período do bit é de:

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

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:

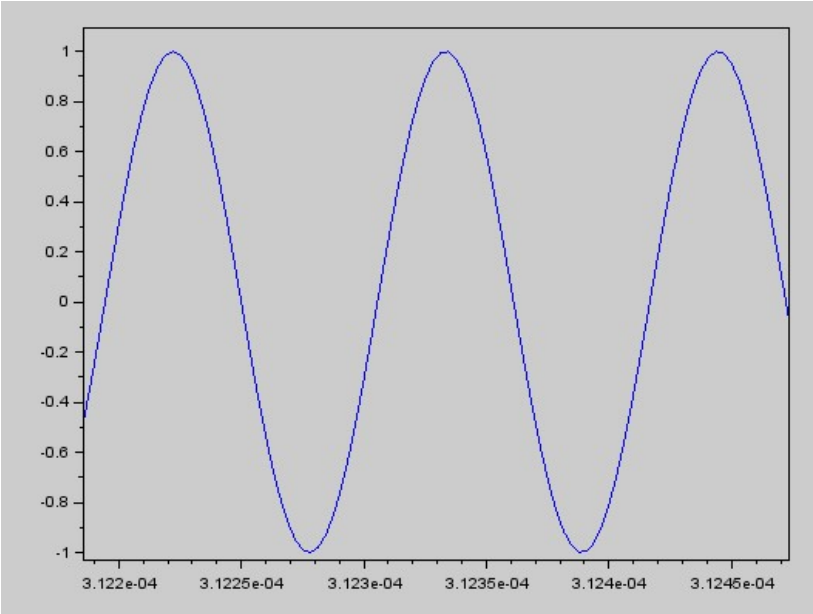
$$1 * período_{bit} = 1 * 0{,}0003125 = 0{,}0003125[\text{s}]$$

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:



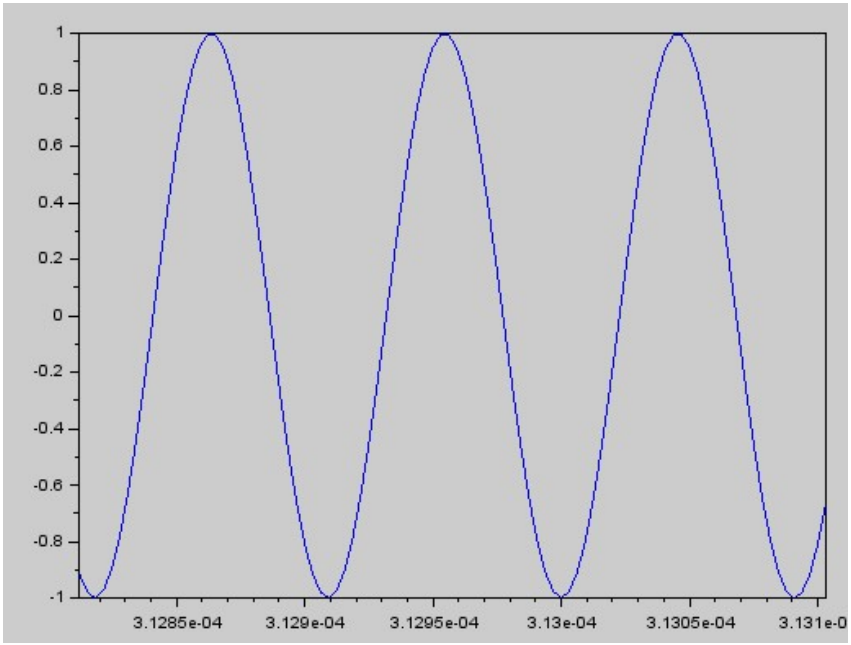
Para fazer o cálculo, se pegará as menores amplitudes do sinal em volta do instante $3{,}1235 * 10^{-4}$ [s], que acontecem aproximadamente em $3{,}1228 * 10^{-4}$[s] e $3{,}1239 * 10^{-4}$[s].

Logo a frequência do cosseno é:

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

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[s], tem-se:



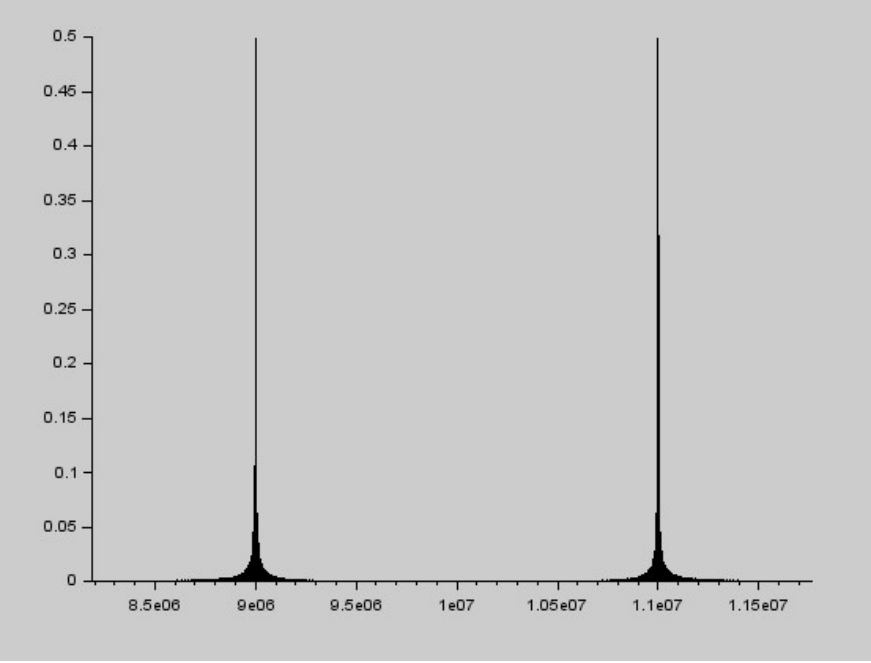
Para fazer o cálculo, se pegará as menores amplitudes do sinal em volta do instante

$3{,}1295 * 10^{-4}$[s], que acontecem aproximadamente em $3{,}1291 * 10^{-4}$[s] e $3{,}13 * 10^{-4}$[s].
Logo a frequência do cosseno é:

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

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.

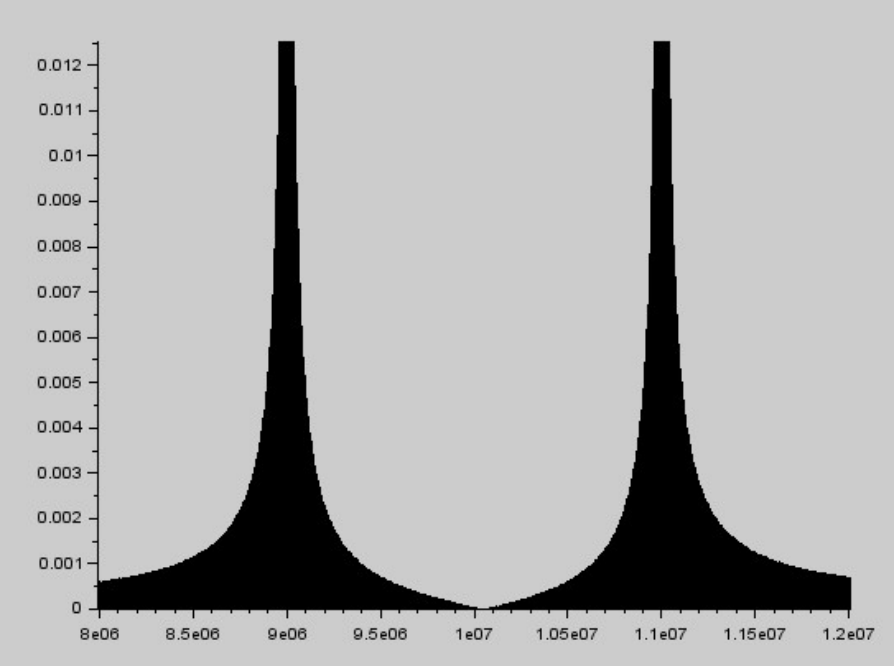
Sabe-se do teorema de amostragem passa-faixa, que é possível recuperar um sinal m(t), se ele for amostrado com f_s:

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

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 $11,6MHz/8,4[MHz]$. Com isso, a frequência máxima do sinal é igual a $11,6[MHz]$ 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[MHz] e terminando em 11,6[MHz], resultando num valor de $(11,6 - 8,4) * 10^6 = 3,2$[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.

Analisando 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.

```
fs = 13e7;
ts = 1/fs;
t2 = 0:ts:bp-ts;
m = [];
for(i=1:1:length(x))
    if(x(i)==1)
        y = 1*cos(2*%pi*9e6*t2);
    else
        y = 1*cos(2*%pi*11e6*t2);
    end
m = [m y];
end
```

Diferente do sinal BASK da etapa 4, o sinal BFSK não tem valores entra 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 multiplica-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);
```

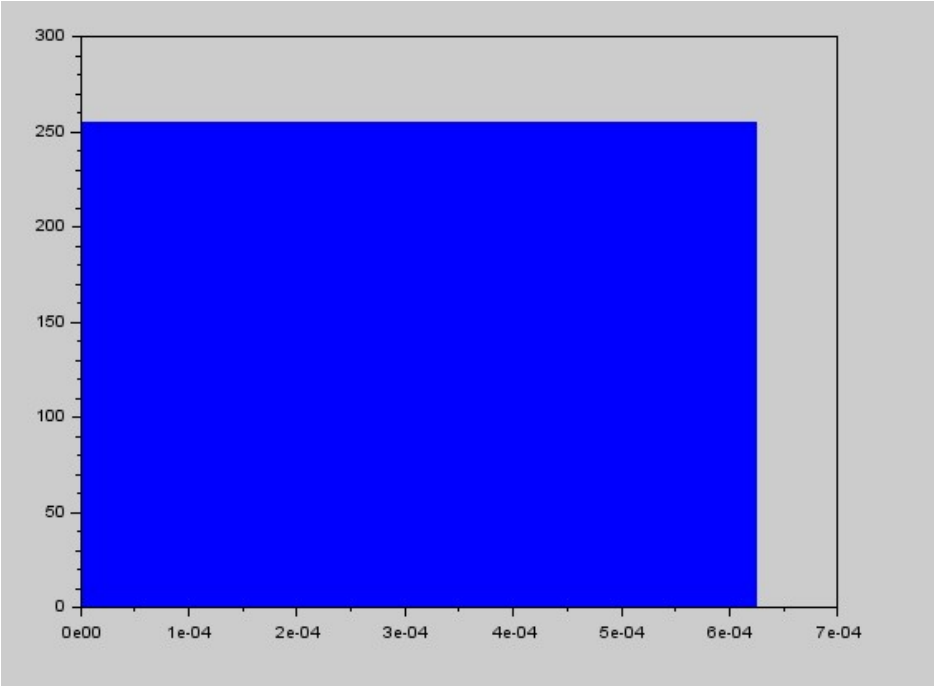
Fazendo um estudo mais aprofundado sobre como foi construído o sinal modulado BFSK percebe-se que para cada bit, tem-se um número de ciclo de portadora igual a:

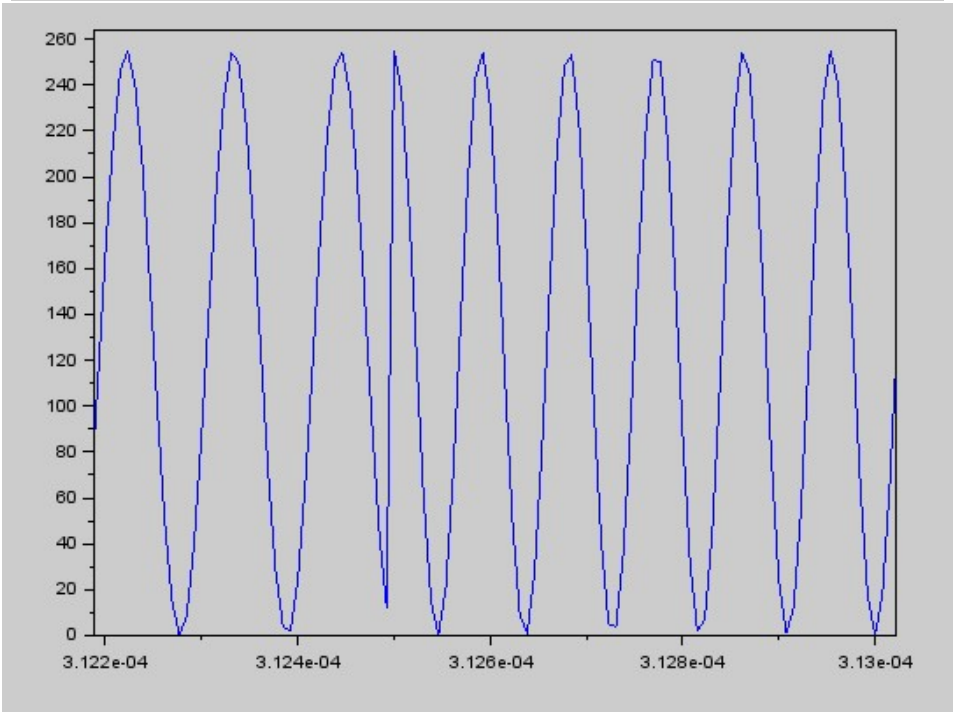
$$p_b * f_s = \frac{1}{(400 * 8)} * 130 * 10^6 = 40625$$

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:

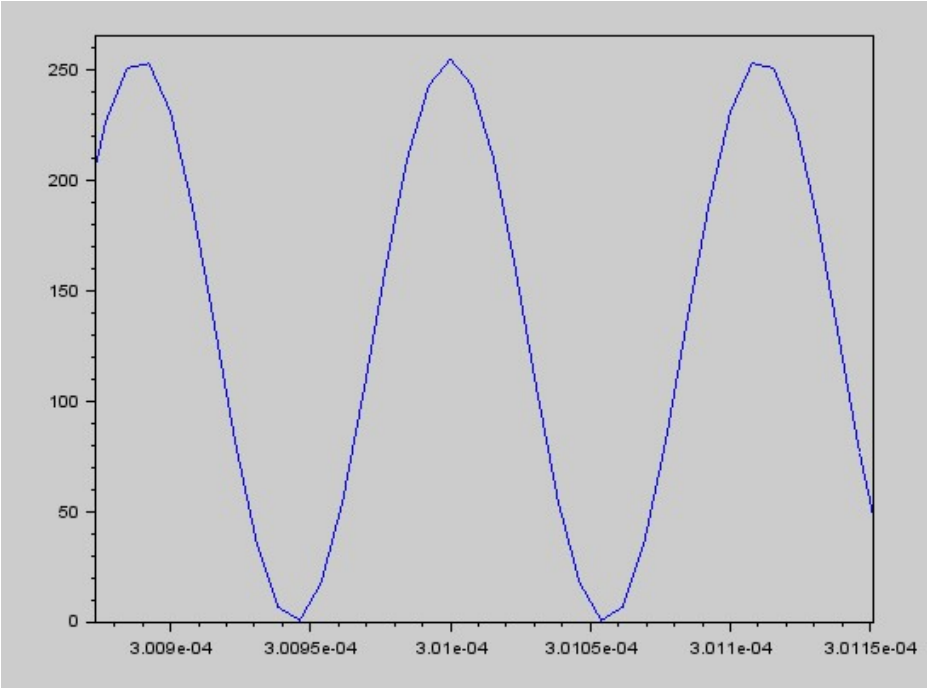
```
t3 = 0:ts:(bp*length(x))
plot(t3,m)
```





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:

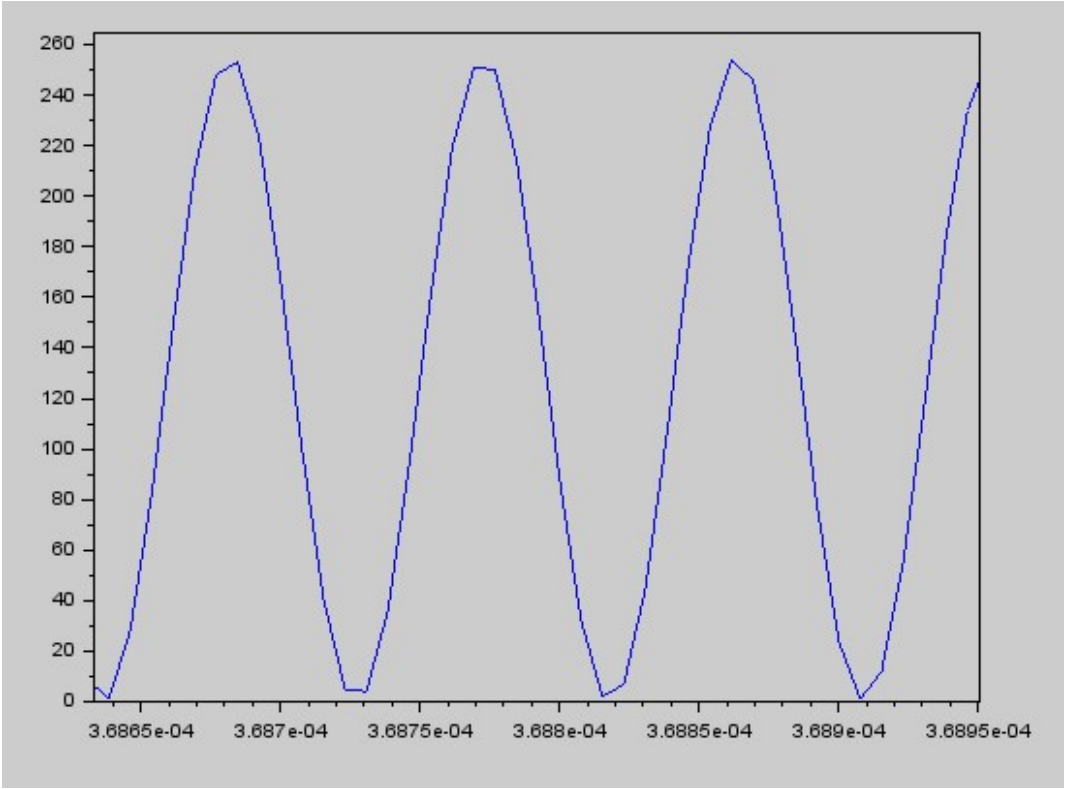


Para fazer o cálculo, se pegará as menores amplitudes do sinal em volta do instante 3,01 $*10^{-4}$[s], que acontecem aproximadamente em $3{,}00945 * 10^{-4}$[s] e $3{,}01055 * 10^{-4}$[s]. Logo a frequência do cosseno é:

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

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[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 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 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 cicle 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:

```
module amostras(input clk, input [16:0] A, output reg [11:0] amostra);

   reg [7:0] mem [0:81249];

   initial begin
      $readmemb("../m.txt", mem);
   end

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

```
module contador(input clk, output reg [11:0] cont, output reg [16:0] A);

initial cont = 12'd0;
initial A = 17'd0;

always @ (posedge clk) begin
   cont <= cont + 12'd1;
   if(cont >= 12'd549) begin
      cont <= 12'd0;
      A <= A + 17'd1;
      if(A == 17'd81249) A <= 17'd0;
      end
   end
endmodule
```

Como foi esclarecido anteriormente, o contador é incrementado com o clock, de 1 digito 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 saida);

    (*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, saida);

endmodule
```

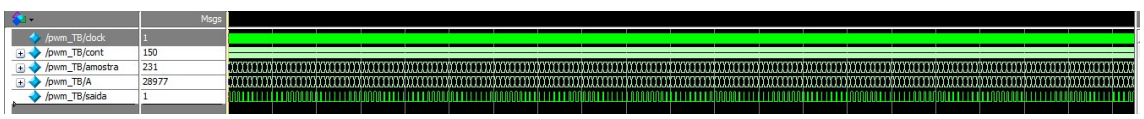
Onde a expressão (*keep=1*) é feita meramente para a visualização dessas variáveis na simulação. O módulo é conectado da seguinte forma: a saída do módulo contador A vai para a entrada do módulo amostras para indicar qual posição da memória e consequentemente qual amostra deverá ser comparada. Então 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 saida;
9
10  pwm DUT(
11  .clock(clock),
12  .saida(saida)
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;
```

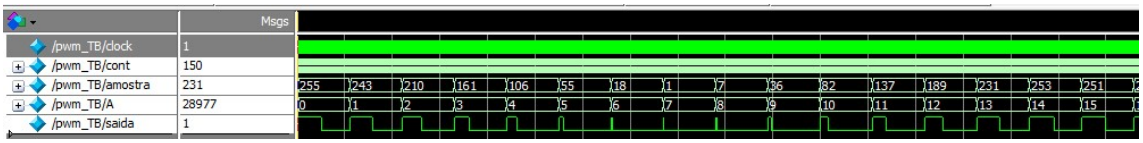
O resultado da simulação é mostrado a seguir:

O resultado da simulação é mostrado a seguir:



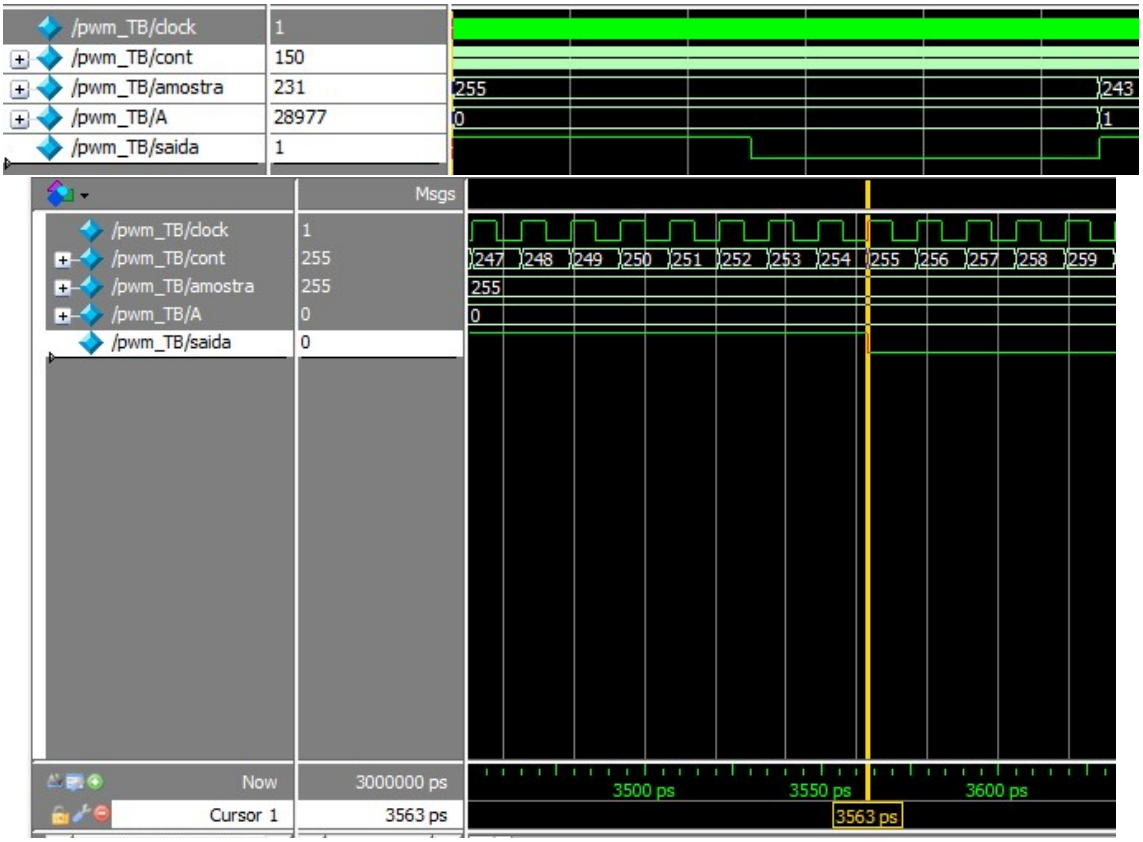
Como a visão original não dá para analisar praticamente nada, graças ao tamanho relativamente grande do vetor de saída PWM, será necessário dar um zoom para poder fazer uma análise mais qualificada.

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

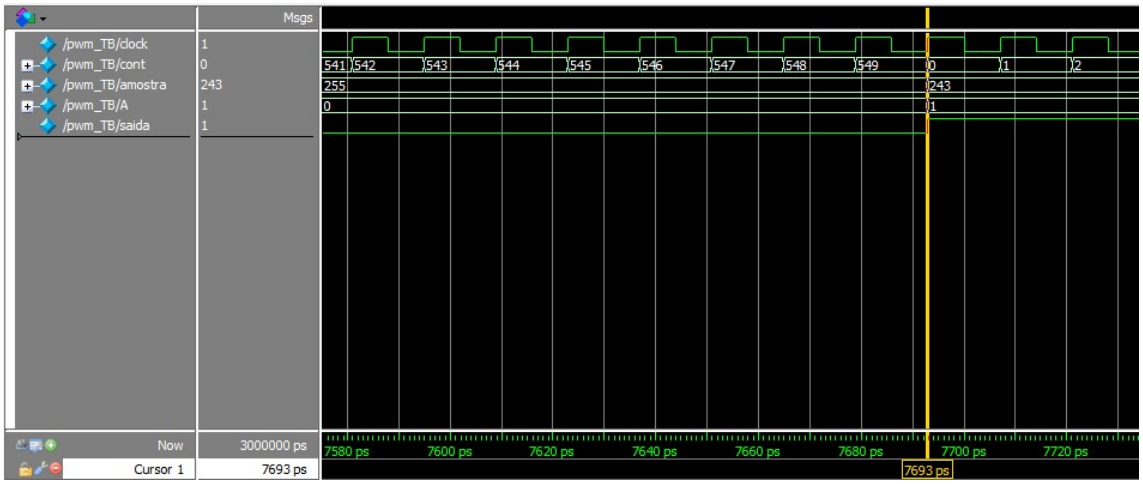


Pode-se ver então que quanto maior o valor da amostra, maior será o comprimento do estado lógico alto do PWM, justamente como era esperado.

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 $\frac{1}{f_s} = \frac{1}{130\ M} = 7692{,}31$ ps, correspondente ao período de amostragem. No entanto, como se utilizou da simulação em gate level, levouse 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:

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

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

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 extrapolação 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
```

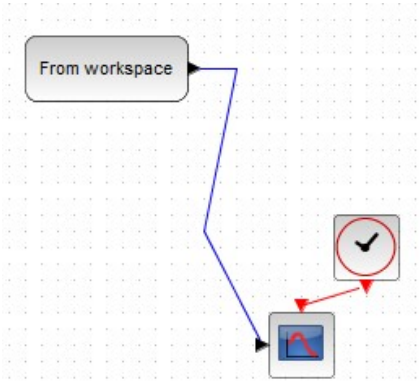
Para verificar a funcionalidade deste código, plota-se o gráfico de t em função do valor do vetor PWM, onde t é o vetor temporal que vai de 0 a 0,000625s (tempo total da sequência binária) incrementado pelo período do clock igual ao inverso de 71500 MHz. Desta forma, tem-se:

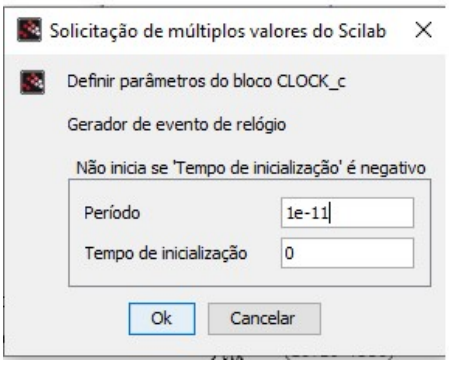
```
73 fclk = fs*550;
74 tclk = 1/fclk;
75 t = 0:tclk:(bp*length(x))-tclk;
```

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().

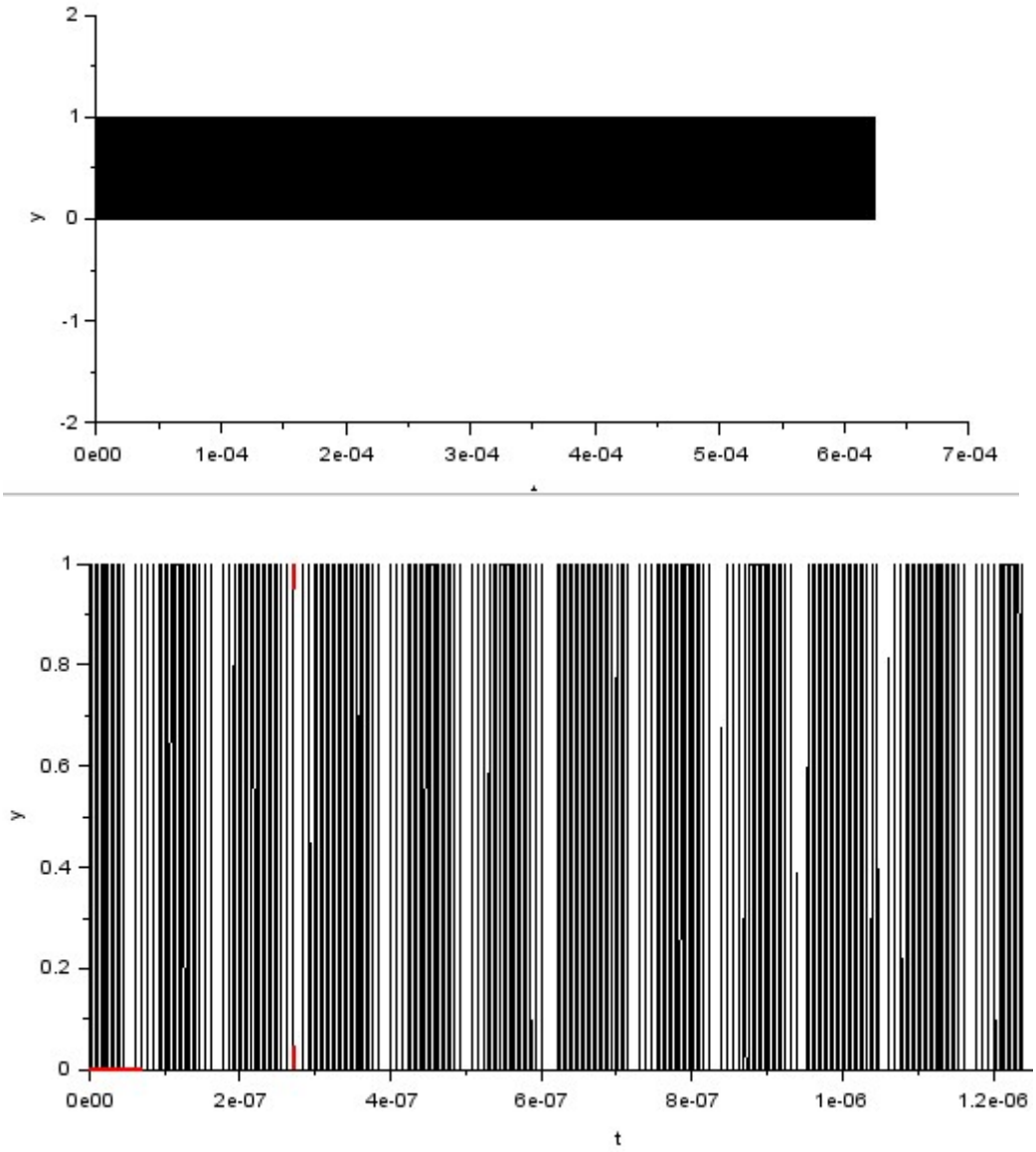
```
77 PWM = PWM';
78 v = struct('time',t','values',PWM')
```

O diagrama de blocos no Xcos fica da seguinte forma:

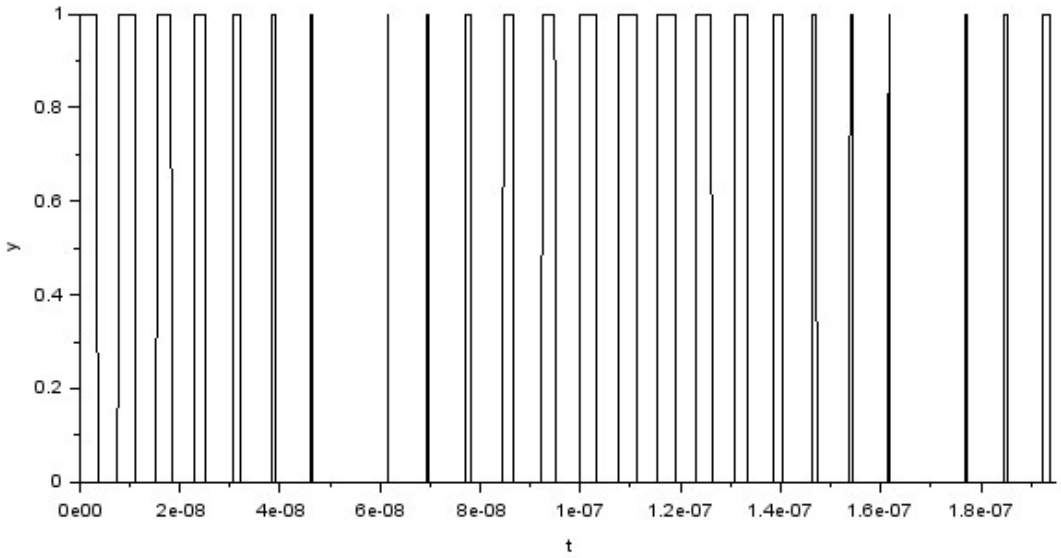




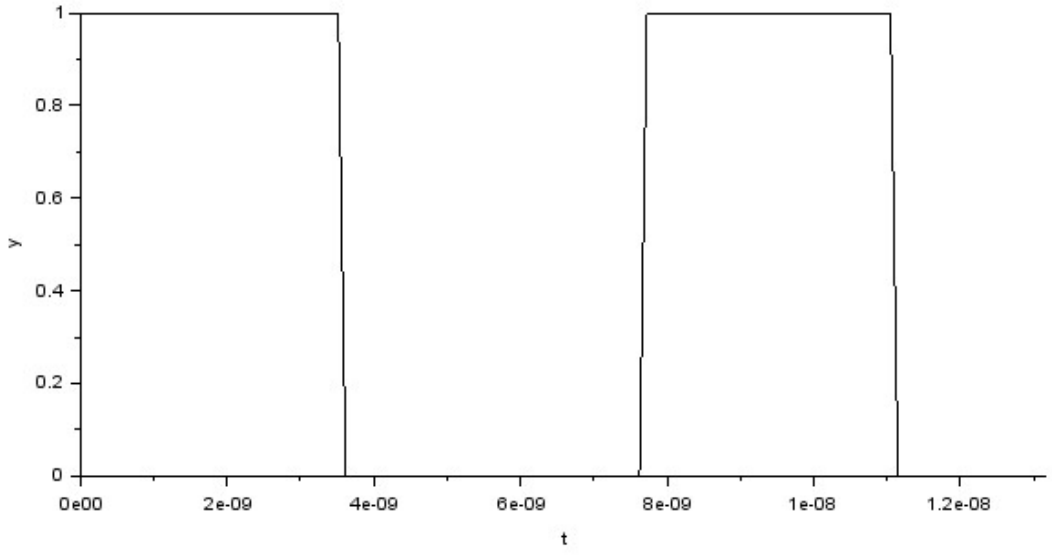
Onde o período do clock foi determinado como sendo igual a $1 * 10^{-11}$, valor inferior ao período do PWM, igual a: $\frac{1}{f_{clk}} = \frac{1}{(130*10^6*550)} = 1{,}40 * 10^{-11}$. Com isso, se poderá ter uma representação fiel da onda resultante, cujo resultado da simulação no osciloscópio, com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

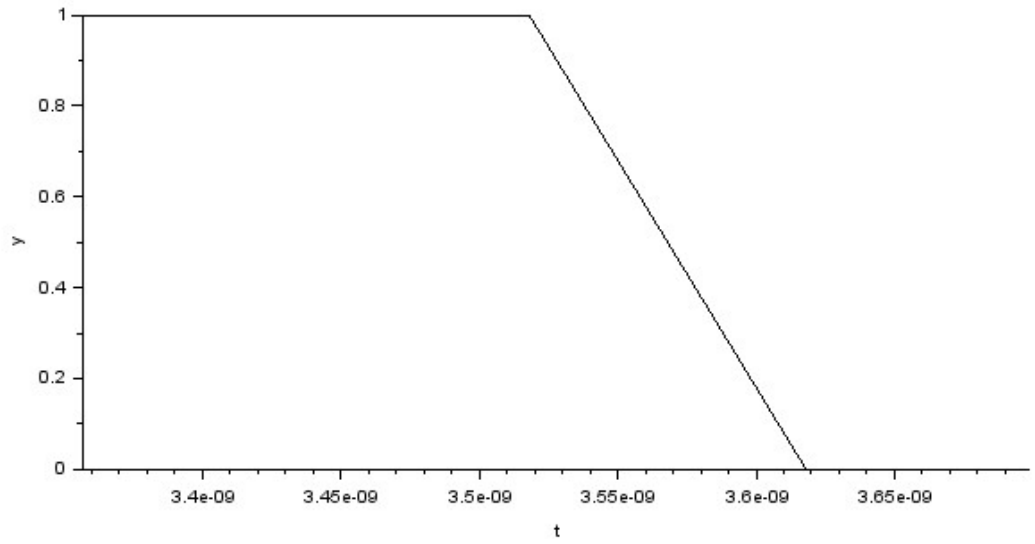


Percebe-se o comportamento geral da onda PWM, em que a sua largura será maior de acordo com o tamanho do valor da amostra. Pode-se dar um zoom para ter uma verificação mais clara deste comportamento.



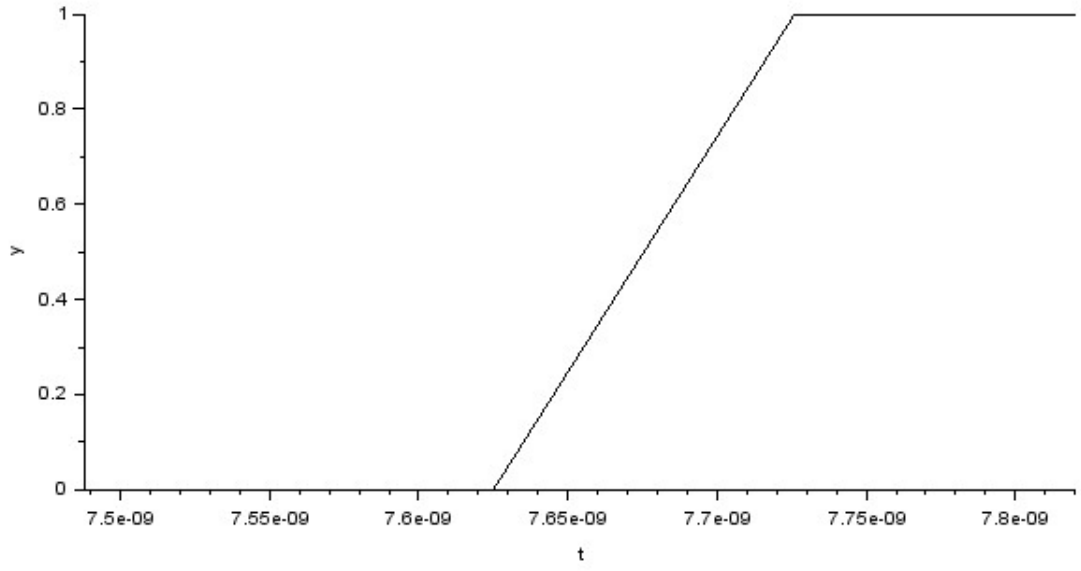
Pode-se aplicar novamente a comparação entre as proporções de tempo e valores já explicada anteriormente. Escolhendo mais uma vez como exemplo a primeira amostra, tem-se que o momento em que é mudado seu nível lógico alto para baixo no pwm é:





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

Já o contador extrapola no seguinte estante:



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

Assim, as proporções ficam da seguinte forma:

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

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

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.