

ETAPA 06) Obter o sinal PWM (usar o teorema da amostragem passa-faixa) do sinal BPSK ($f_c = 10 \text{ MHz}$) da representação digital de $m(t)$ por simulação de FPGA e no Scilab.

NOME: LINCOLN WALLACE VELOSO ALMEIDA

MATRÍCULA: 2018018715

NOME: GABRIEL MEDEIROS CARDOSO

MATRÍCULA: 2018014574

NOME: ITALO BARBOSA BARROS

MATRÍCULA: 2018008924

NOME: BRUNO DE MELLO DUARTE

MATRÍCULA: 2016010988

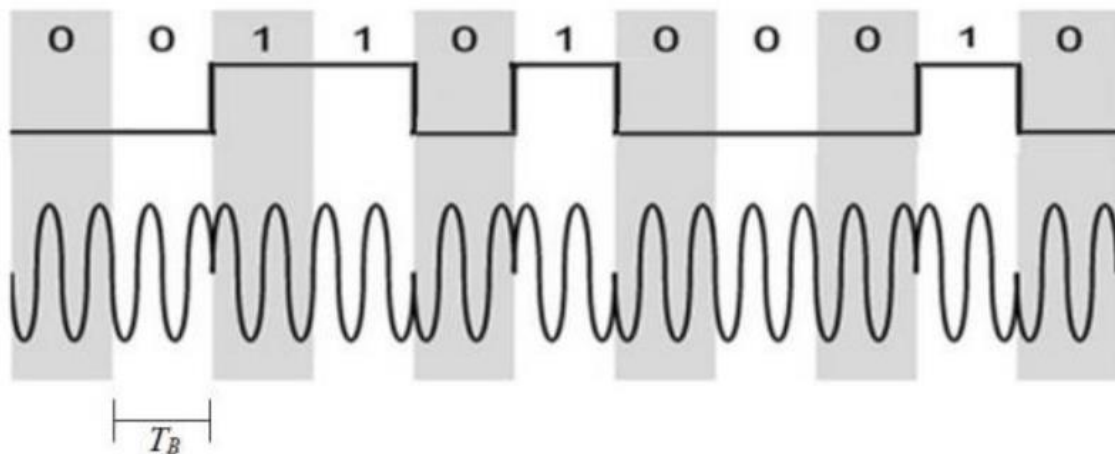
Assim como na etapa 4, para aplicar a modulação digital BPSK escolheu-se uma amostra do sinal PCM criado na etapa 1. A técnica de modulação BPSK terá o seguinte funcionamento:

- 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:

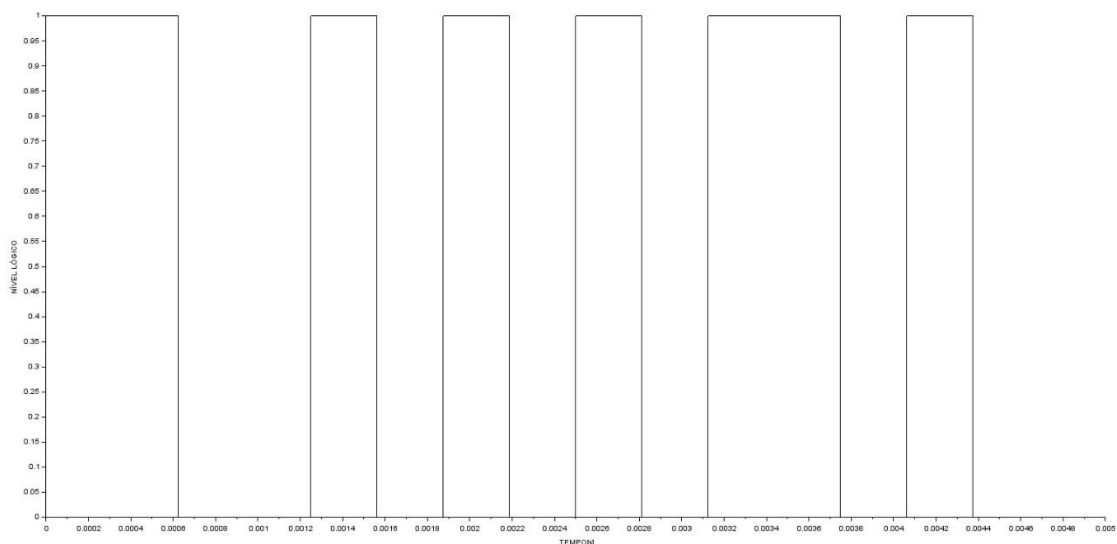
84	1	1	0	0	1	0	1	0
85	1	0	1	1	0	1	0	0

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: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:

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

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:

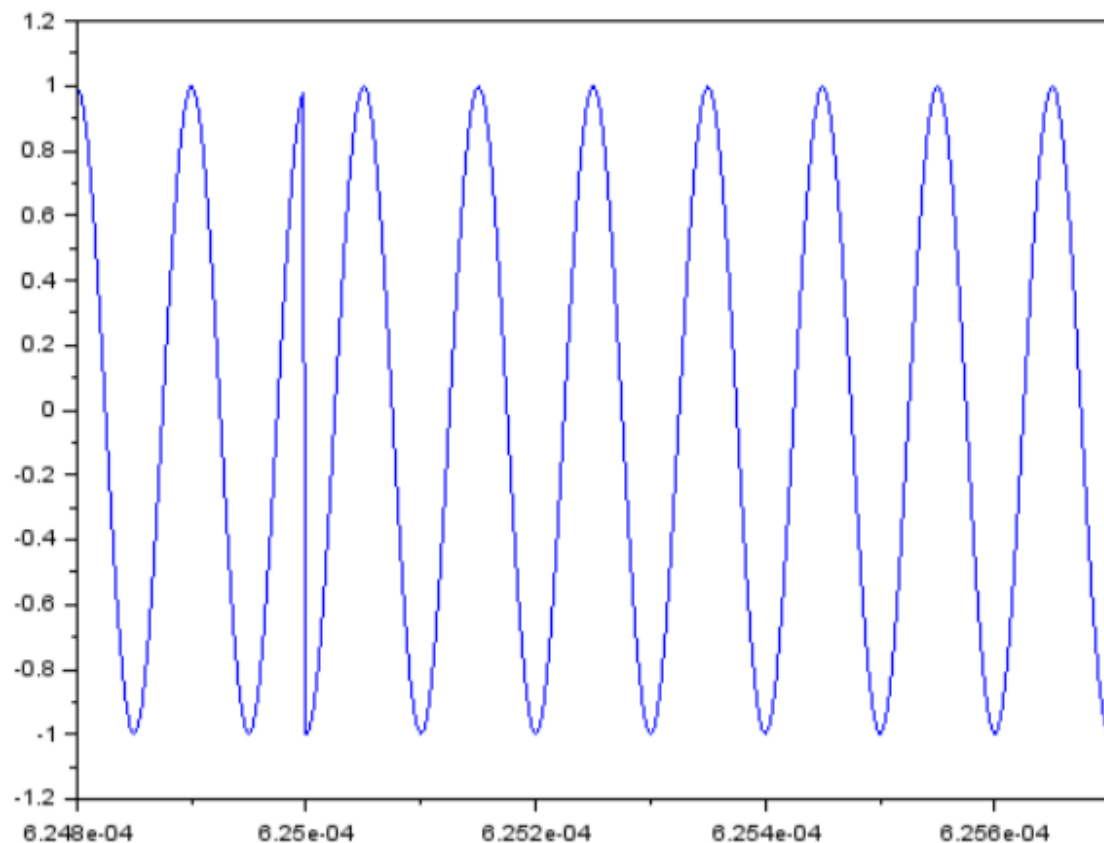
```
18 t2 = 0:bp/100000:bp - bp/100000;  
19 m = [];  
20 for (i=1:length(x))  
21     if (x(i)==1)  
22         y=1*cos(2*pi*10e6*t2);  
23     else  
24         y=0;  
25     end  
26     m = [m y];  
27 end  
28 t3 = 0:bp/(100000):(bp*length(x))-bp/100000;  
29 plot2d(t3,m);
```

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:

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



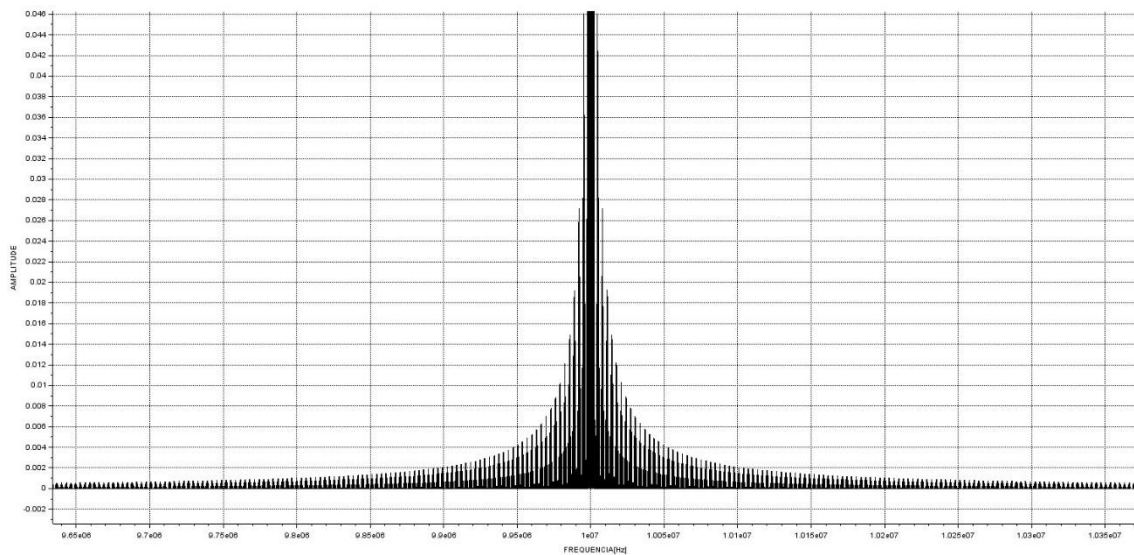
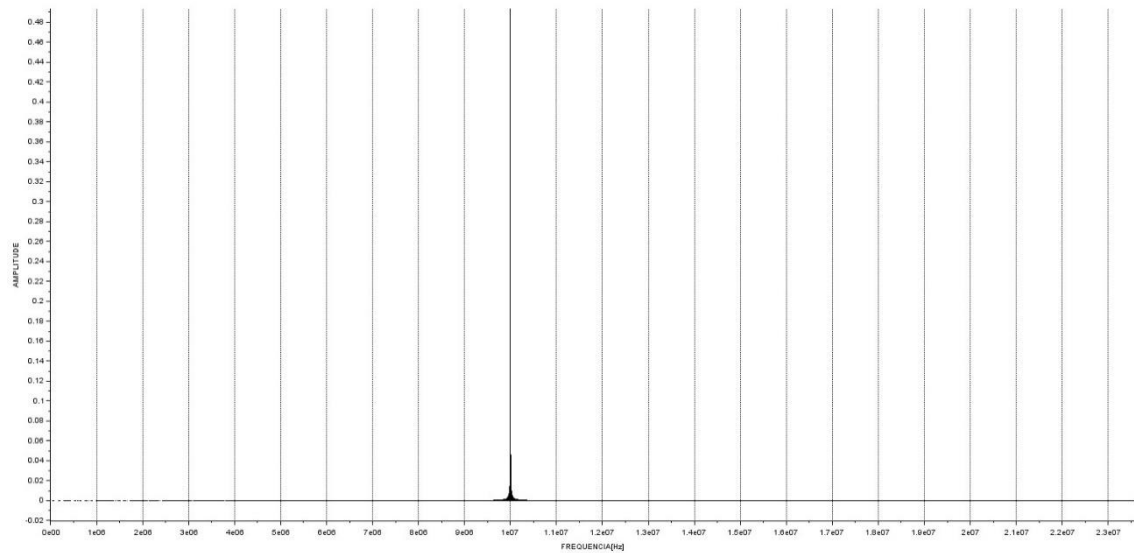
Nota-se claramente os cossenos, bem como sua transição.

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.

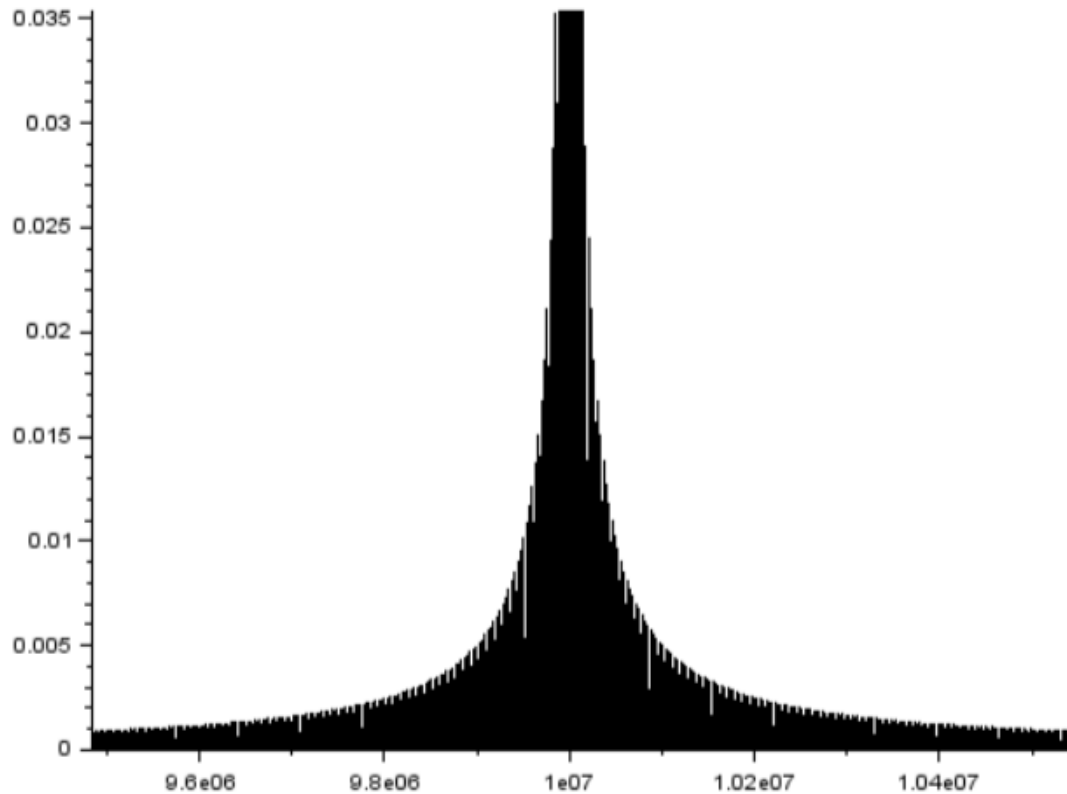
Com o teorema da amostragem passa-faixa faz-se possível recuperar o sinal $m(t)$, levando em conta que este foi amostrado com uma frequência f_s dada por:

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

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].

$$10,5[MHz] - 9,5[MHz] = 1[MHz]$$

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:

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

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

$$f_{s(ESCOLHIDO)} = 40[MHz]$$

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 1 0 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: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);
```

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

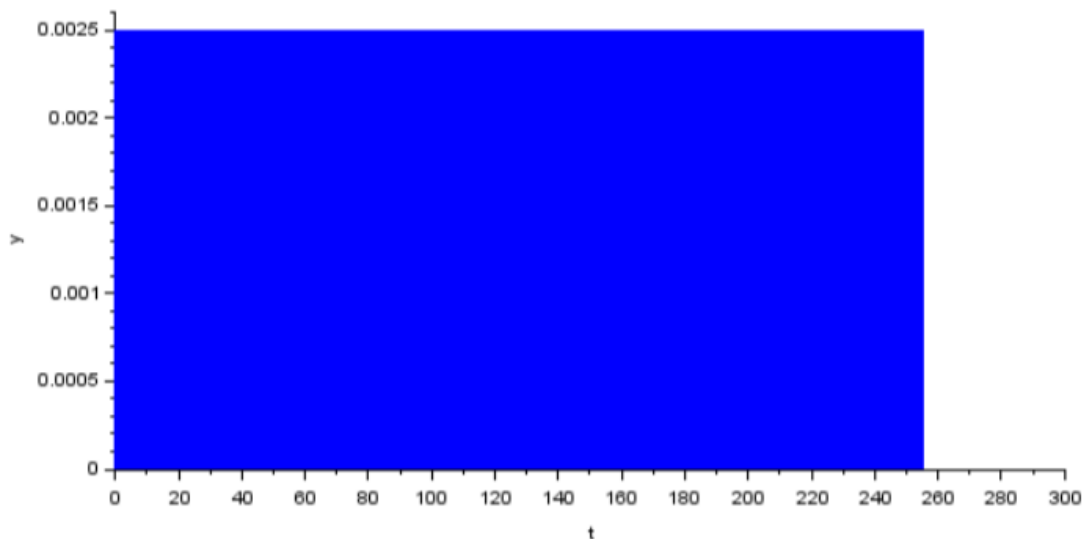
$$p_b * f_s = \frac{1}{(400 * 8)} * 40 * 10^6 = 12500$$

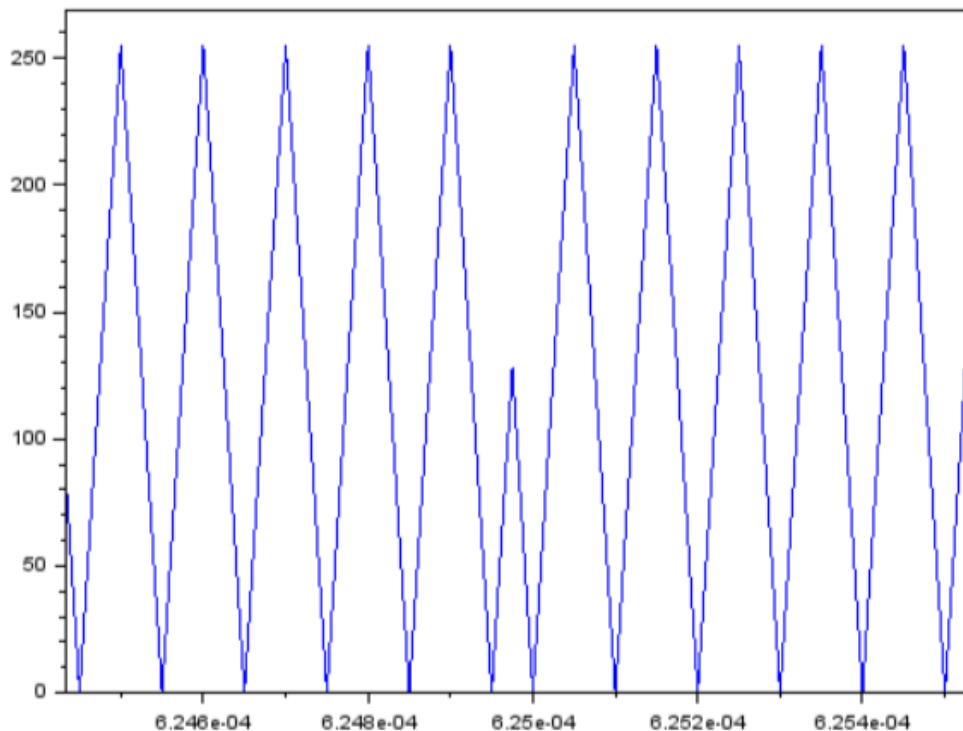
Há 8 bits na sequência binária, então o número total de amostras do sinal modulado é igual a:

$$12500 * 8 = 100000[amostras]$$

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 Cycle 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:

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

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 vai 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

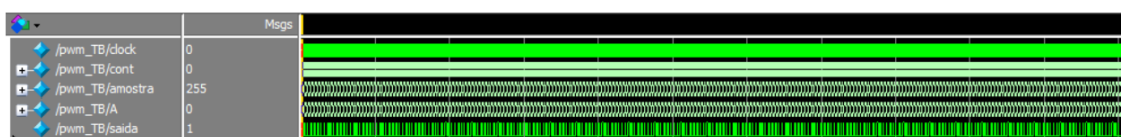
initial begin
    $init_signal_spy("/pwm_TB/DUT/cont", "cont", 1);
    $init_signal_spy("/pwm_TB/DUT/amostra", "amostra", 1);
    $init_signal_spy("/pwm_TB/DUT/A", "A", 1);
end

initial
    #550000 $stop;

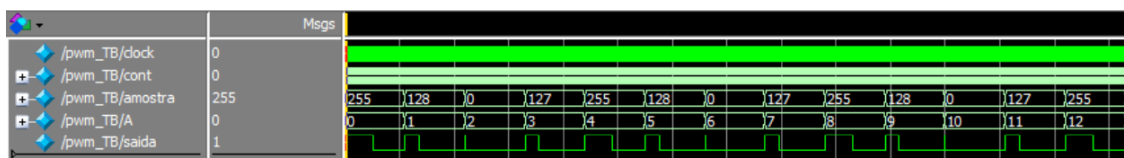
endmodule

```

Tem-se como resultado de simulação:



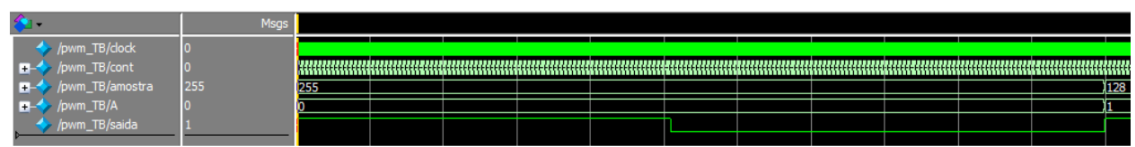
É visualizada as 12 primeiras amostras do sinal modulado BPSK onde era representado o bit 1, e em seguida as primeiras amostras do primeiro bit 0 da amostra PCM escolhida, nota-se que:



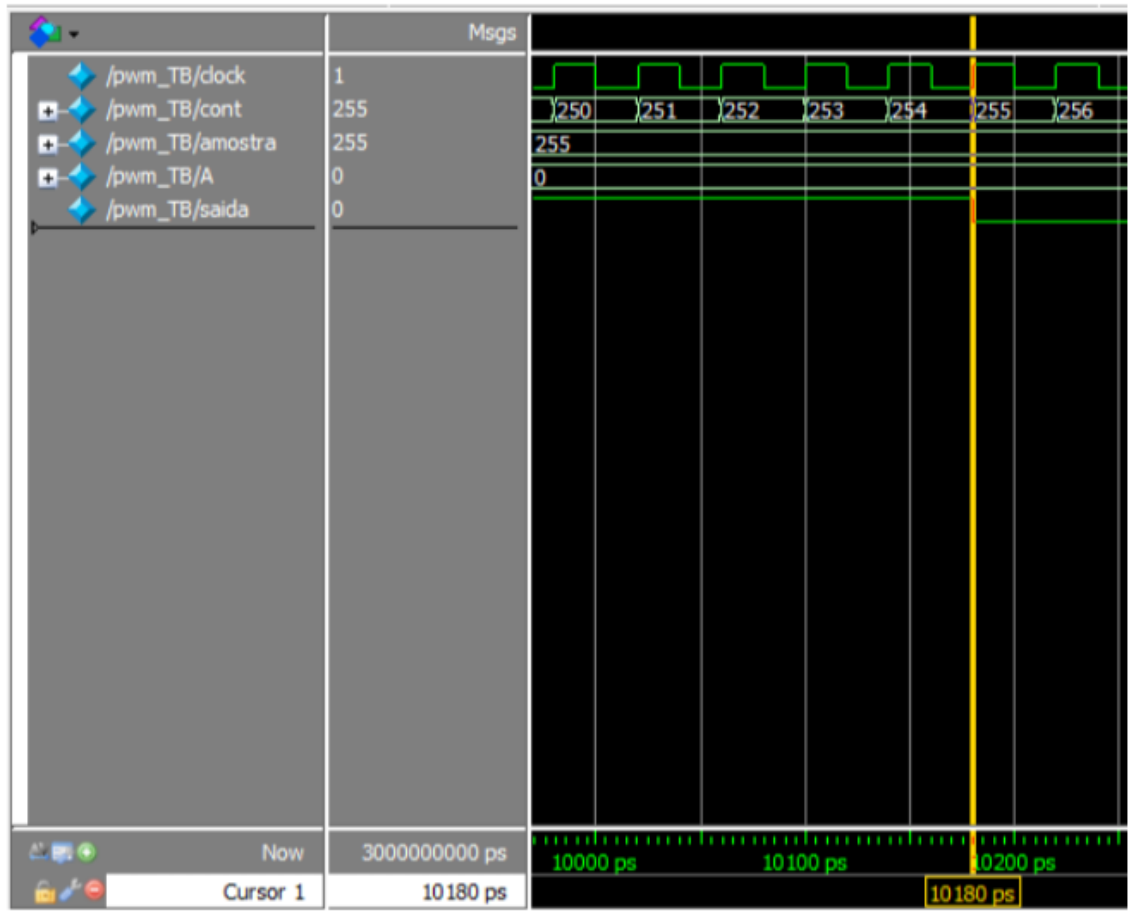
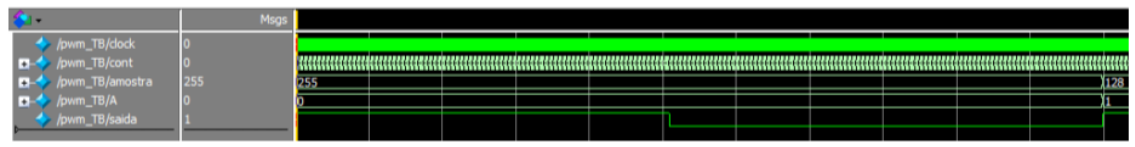
Pode-se ver então que quando a amostra é igual a 0, o sinal PWM resultante também é igual a 0 e quando a amostra é igual a 255, o sinal PWM terá um período em estado alto até o contador passar esse valor e então volta para o estado baixo.

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:



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 extrapolação 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
```

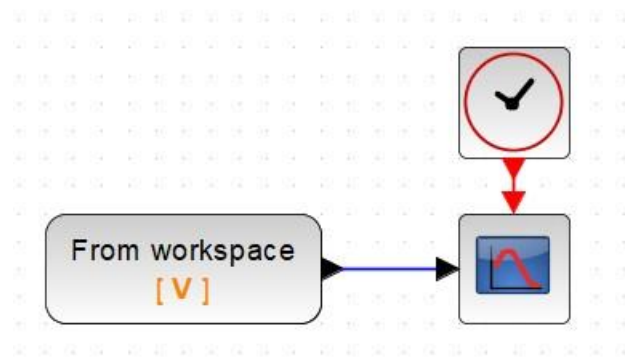
Para averiguar o código, plota-se o gráfico de **t** em função do valor do vetor PWM, onde **t** é o vetor temporal que vai de 0 a 0,0025s (tempo total da sequência binária) incrementado pelo período do clock igual ao inverso de 22000 MHz. Desta forma, tem-se:

```
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 feita pelo Xcos, onde os valores do console foram passados a ele através do comando struct().

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

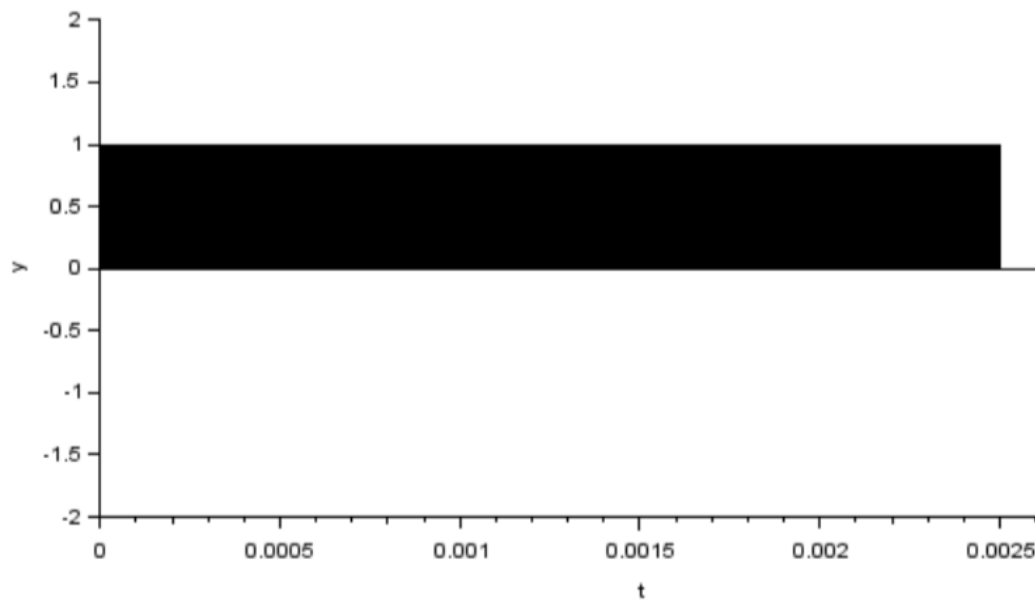
Obteve-se o seguinte diagrama de blocos:

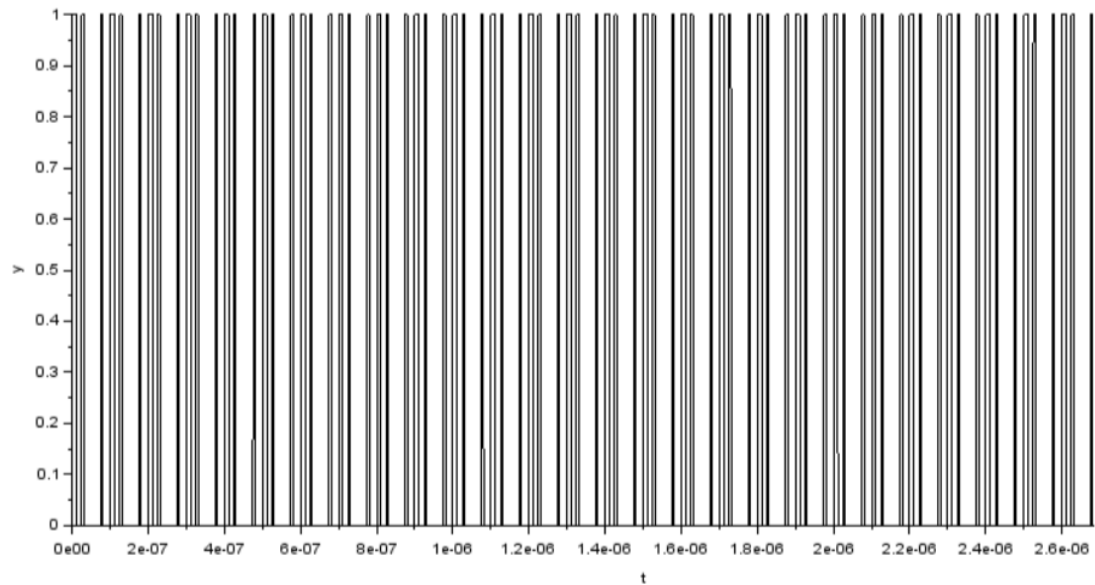


O período de clock foi determinado sendo igual a $f_{clk} = 1 * 10^{-11}$, valor este inferior ao período do PWM, sendo igual a:

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

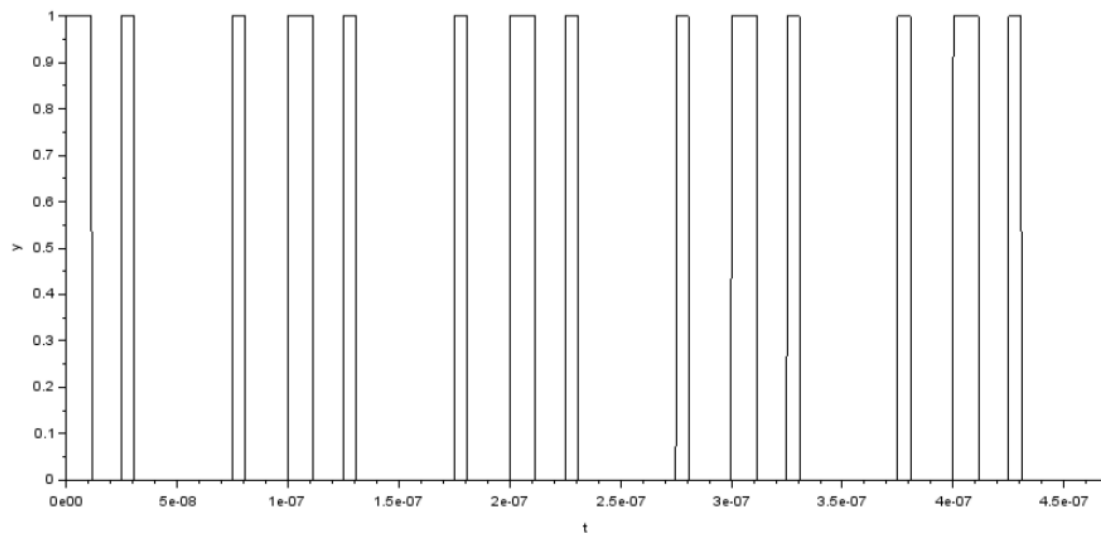
Assim, é obtida uma representação fiel da onda resultante, cujo resultado da simulação no osciloscópio, com tempo total de 0,003[s], é mostrado logo abaixo:



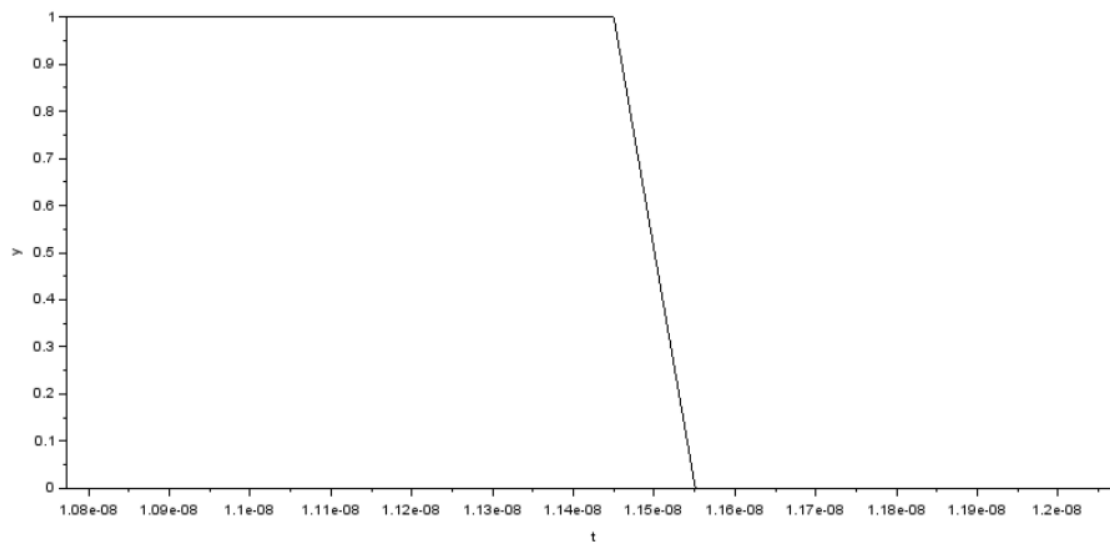
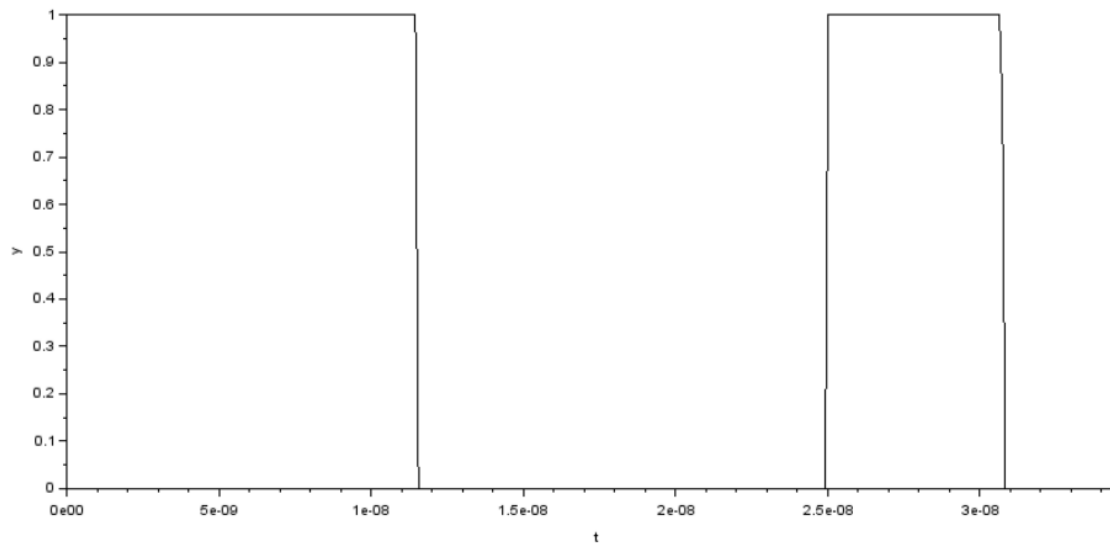


Onde pode-se ver o comportamento geral da onda PWM, em que o comprimento do estado lógico alto varia entre $\frac{255}{550}$, $\frac{125}{550}$ e 0, isto é, ele será maior de acordo com o tamanho da amostra. Pode-se dar um zoom para ter uma verificação mais clara deste comportamento.

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:

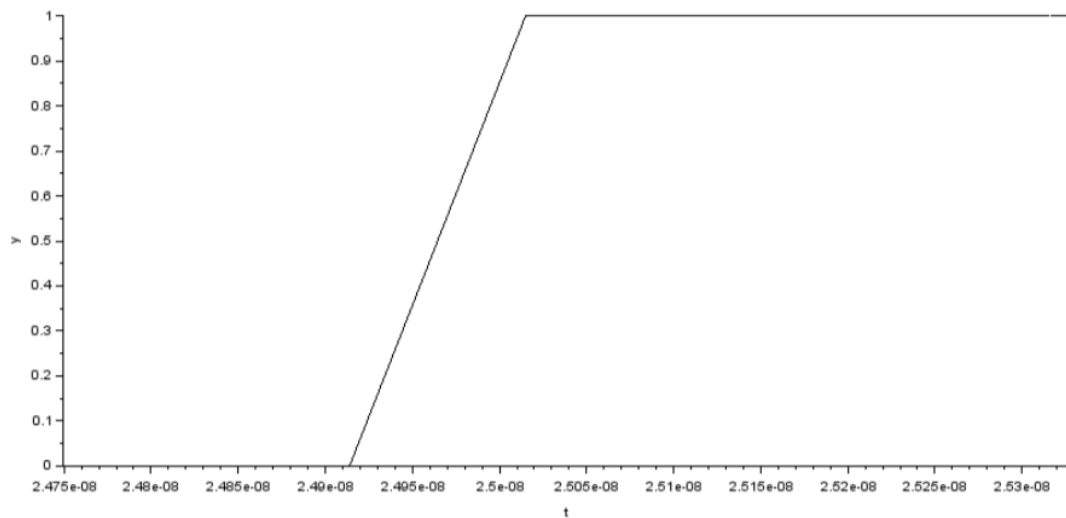


Pode-se aplicar novamente a comparação entre as proporções de tempo e valores já explicadas anteriormente. Tomando-se mais uma vez como exemplo a primeira amostra, tem-se que o momento em que é mudado seu nível lógico alto para baixo no pwm é:



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

O instante de extrapolação do contador é:



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

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

$$\frac{Valordaamostra}{Valormáximocontador} = \frac{255}{550} = 0,46364$$

$$\frac{Tempoemnívellógicoalto}{Períododaondaparaamostra} = \frac{1,155 * 10^{-8}}{2,491 * 10^{-8}} = 0,46367$$

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.