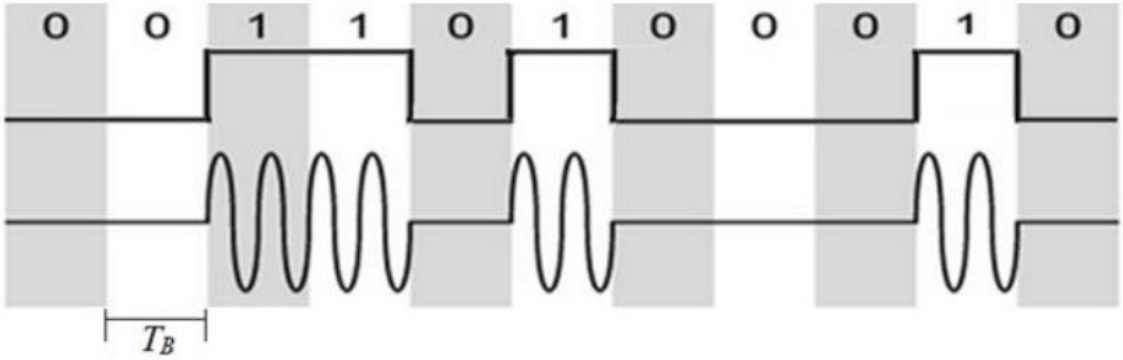
Para a técnica de modulação BASK foram escolhidas duas amostras do sinal PCM criado a partir dos métodos já citados na etapa 01. No entanto, foi utilizada uma frequência de amostragem para o sinal PCM de 400[Hz] ao invés de 800[Hz], como feito originalmente. A razão da escolha de uma menor frequência 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: $m(t) = 1 * \cos(2\pi\ fc\ t)$, onde fc 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:

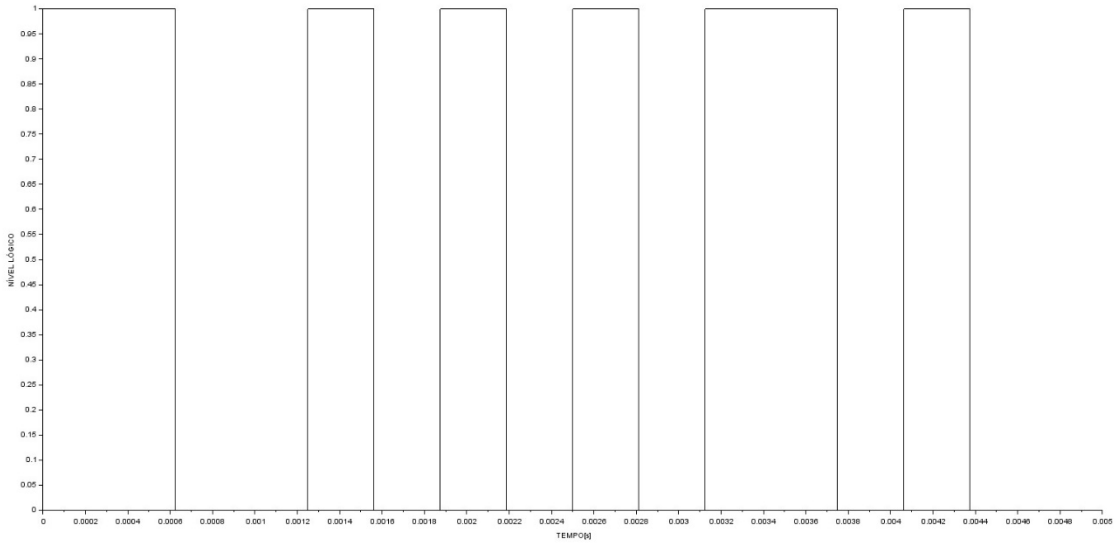


Foram escolhidas duas amostras consecutivas do sinal PCM que apresentam uma variação de 0 para 1 bit a bit, que são justamente as amostras 84 e 85, visto a seguir:

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

Construção da sequência no Scilab:

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

for n=1:1:length(x)
    if x(n) == 1;
        se = ones(1,1000);
    else
        se = zeros(1,1000);
    end
    bit = [bit se];
end
t1 = 0:bp/1000:(bp*length(x))-bp/1000;
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:

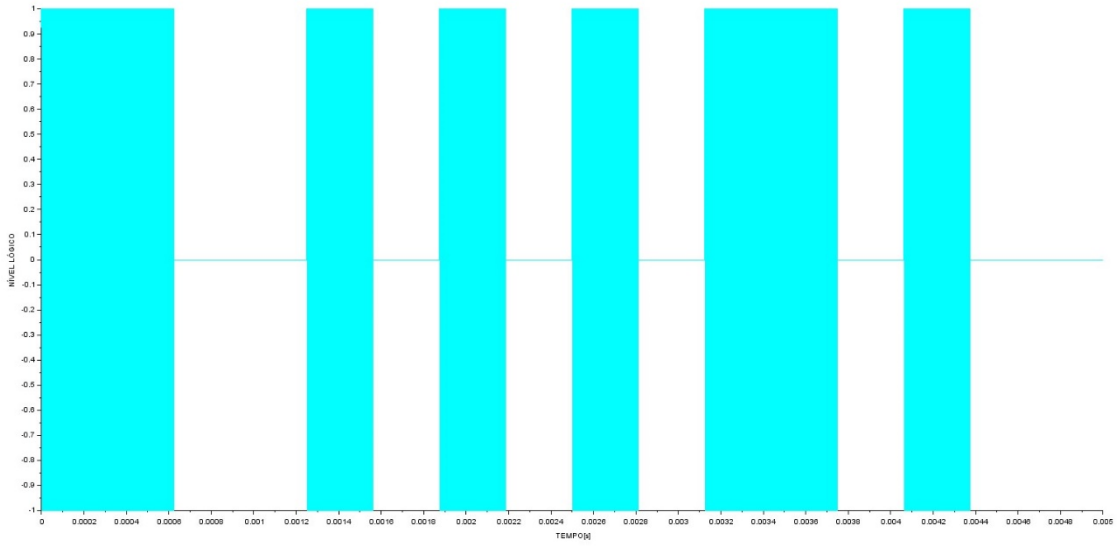
$$f_s = \frac{100000}{p_b} = \frac{100000}{0{,}0003125} = 320[MHz]$$

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

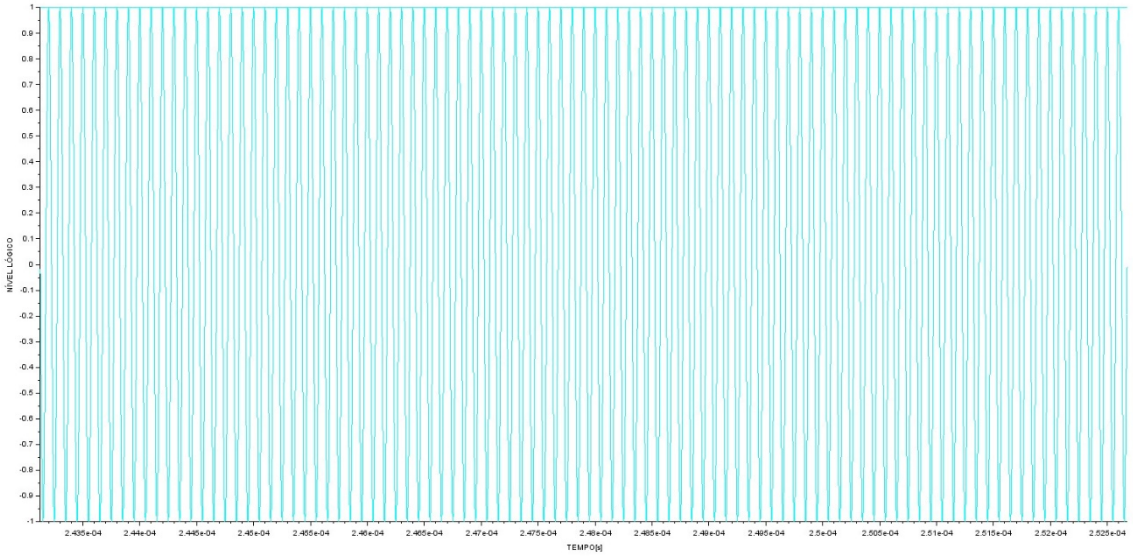
Vale ressaltar que esta não é a frequência de amostragem final, e sim uma frequência provisória para gerar uma boa visualização de como é o sinal modulado e o seu espectro de amplitude. A frequência de amostragem final será determinada aplicando-se o teorema de amostragem passa-faixa.

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);
```



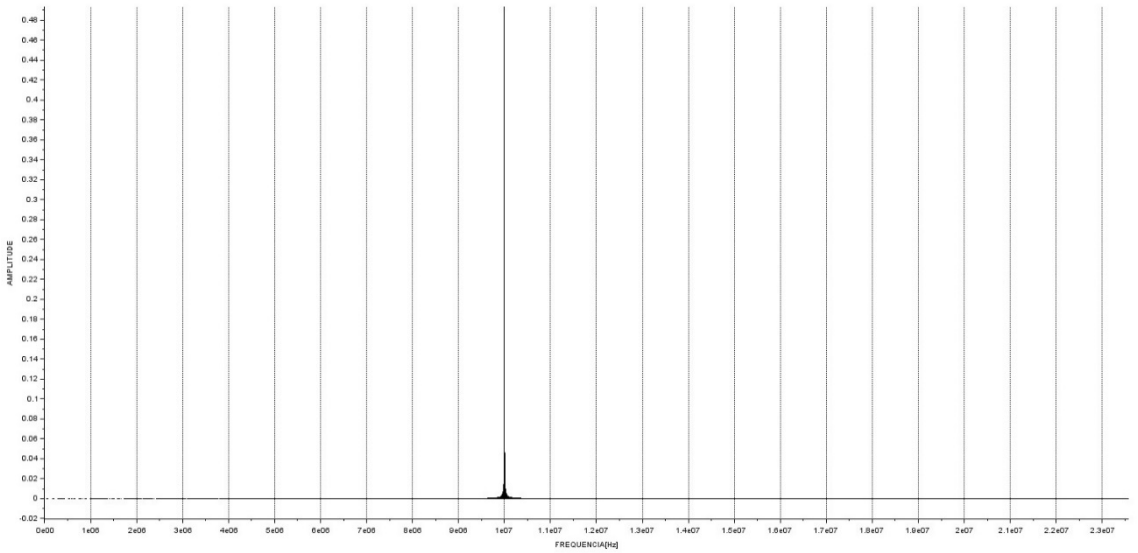
Visualizando melhor a região onde o bit representado é 1:

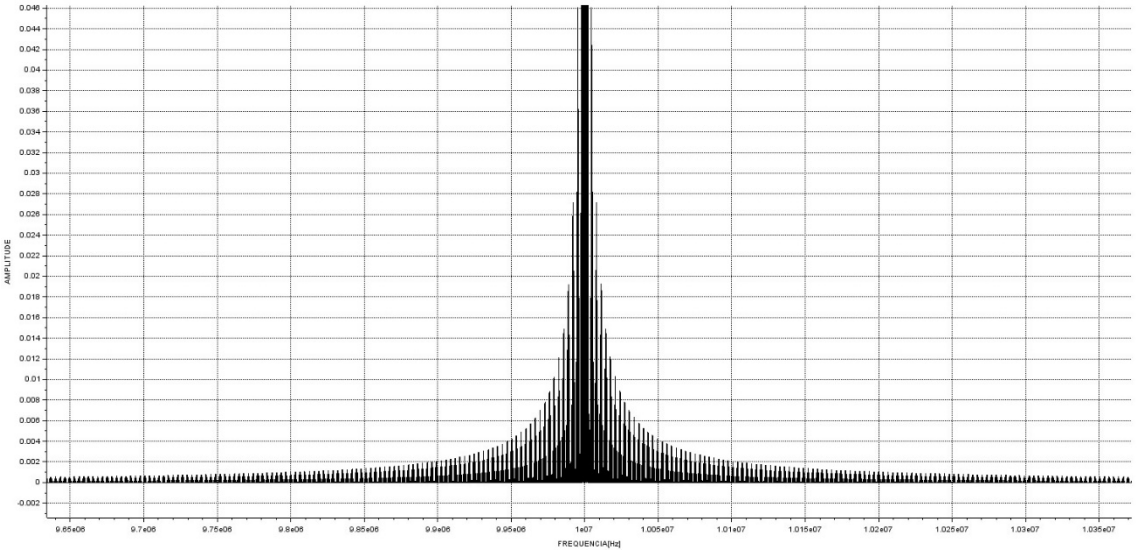


Nota-se claramente os cossenos dados por: $m(t) = 1 * \cos(2\pi * 10 * 10^6 * t)$

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[Mhz] – 9,9[MHz] = 200[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:

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

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.

```
x = [1 1 0 0 1 0 1 0 10 1 1 0 1 0 0];
fs = 10000000;
bp = 1/(400*8);
ts = 1/fs;
t2 = 0:ts:bp-ts;
m=[];
for (i=1:1:length(x))
    if(x(i)==1)
        y=1*cos(2*%pi*10e6*t2)
    else
        y=0;
    end
    m = [m y];
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 cosseno 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 cosseno, 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
    $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 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.

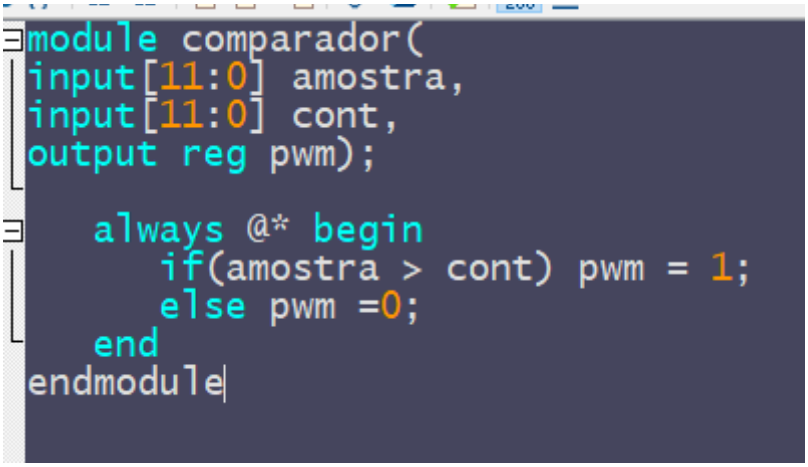
O módulo do contador é dado por:

```
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:



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:

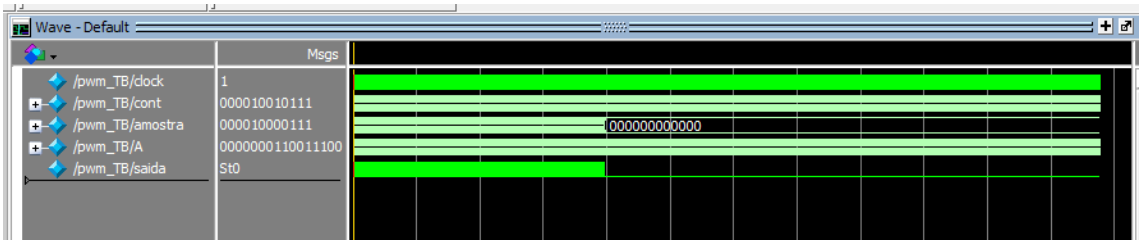
```
module pwm(
input clk,
output saida);

    (*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.

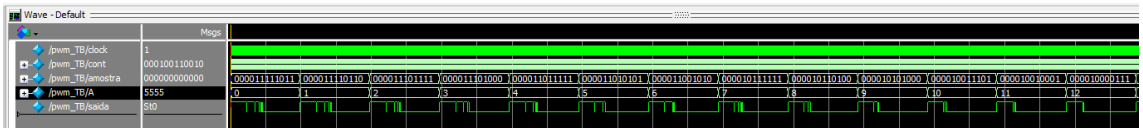
É implementado um clock com período de 0,182[ns], o que gera uma frequência aproxima de 5500[MHz]:

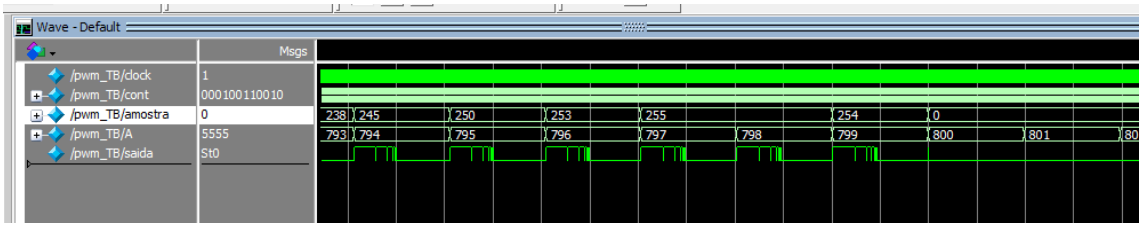
```
module pwm_TB;
reg clock;
reg [11:0] cont;
reg [11:0] amostra;
reg [15:0] A;
wire saida;

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:



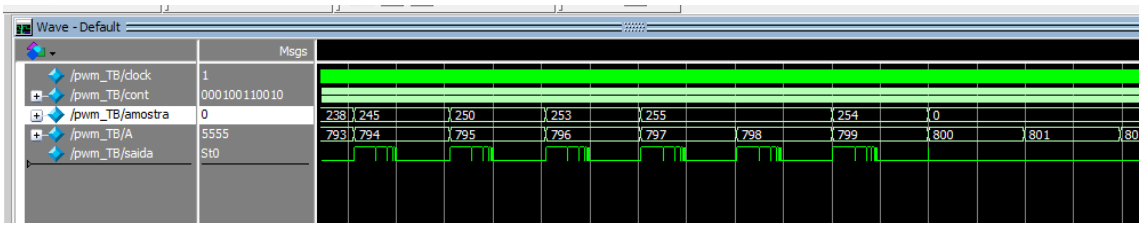
É visualizada as 12 primeiras amostras do sinal modulado BASK onde era representado o bit 1, e em seguida as primeiras amostras do primeiro bit 0 da amostra PCM escolhida, nota-se que:



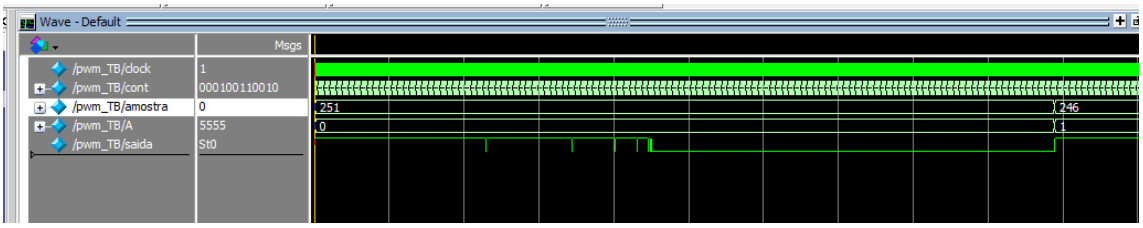


Pode-se ver então que quando a amostra é igual a 0, o sinal PWM resultante também é igual a 0 e quando a amostra é igual a 255, o sinal PWM terá um período em estado alto até o contador passar esse valor e então volta para o estado baixo.

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:

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

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

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:

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

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 extrapolação do contador. Assim, tem-se:

```
for i=1:50000
    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
```

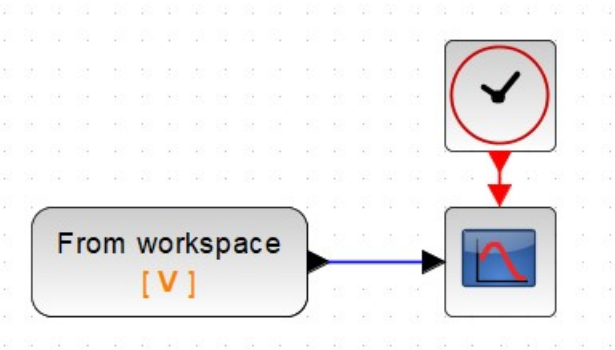
É plotado um gráfico de t em função do valor do vetor PWM, onde t é o vetor temporal que vai de 0 a 0,005[s], sendo incrementado pelo período do clock igual ao inverso de 5500 MHz. Sendo assim:

```
fclk = fs*550;
tclk = 1/fclk;
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().

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

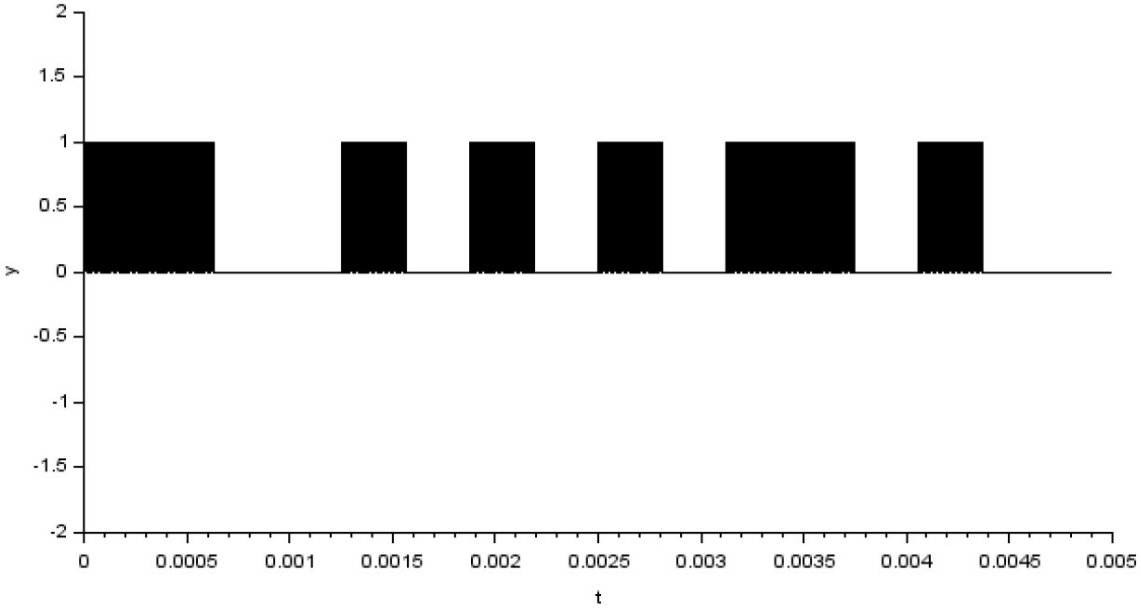
Obteve-se o seguinte diagrama de blocos:



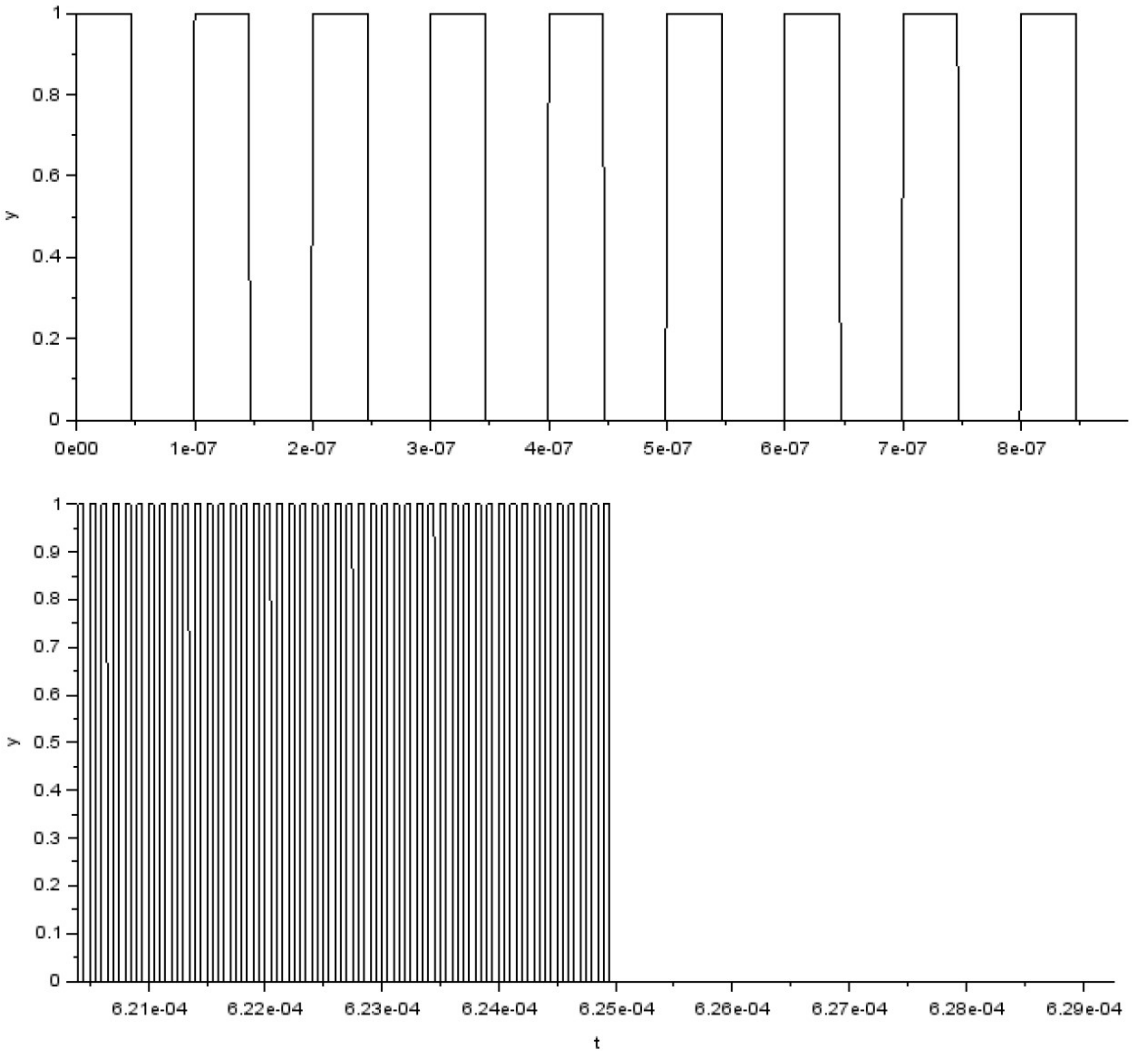
O período de clock foi determinado sendo igual a $1 * 10^{-10}$, valor este inferior ao período do PWM, sendo igual a:

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

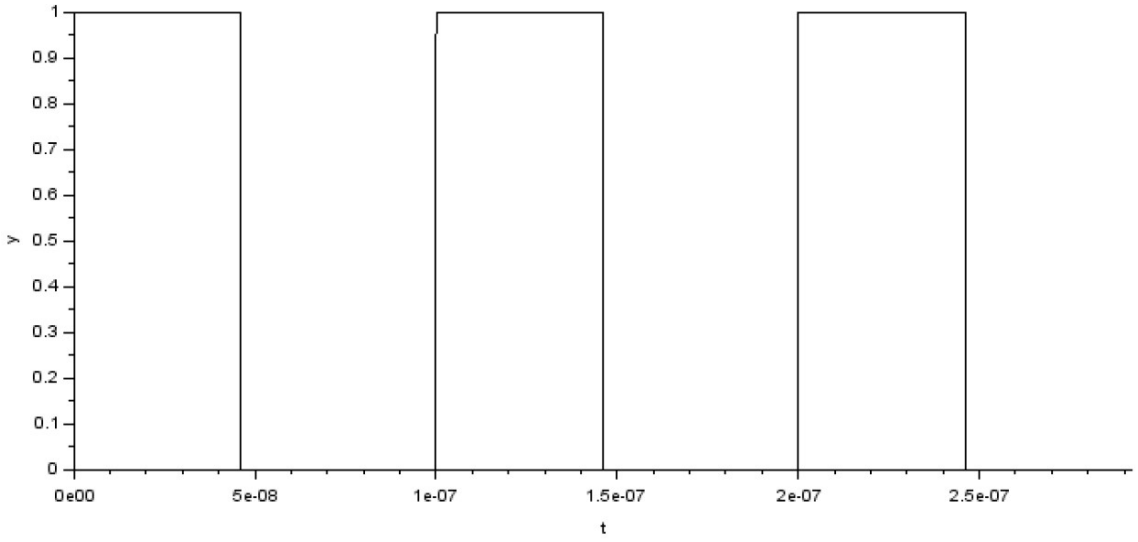
Assim, é obtida uma representação fiel da onda resultante, cujo resultado da simulação no osciloscópio, com tempo total de 0,005[s], é mostrado logo abaixo:

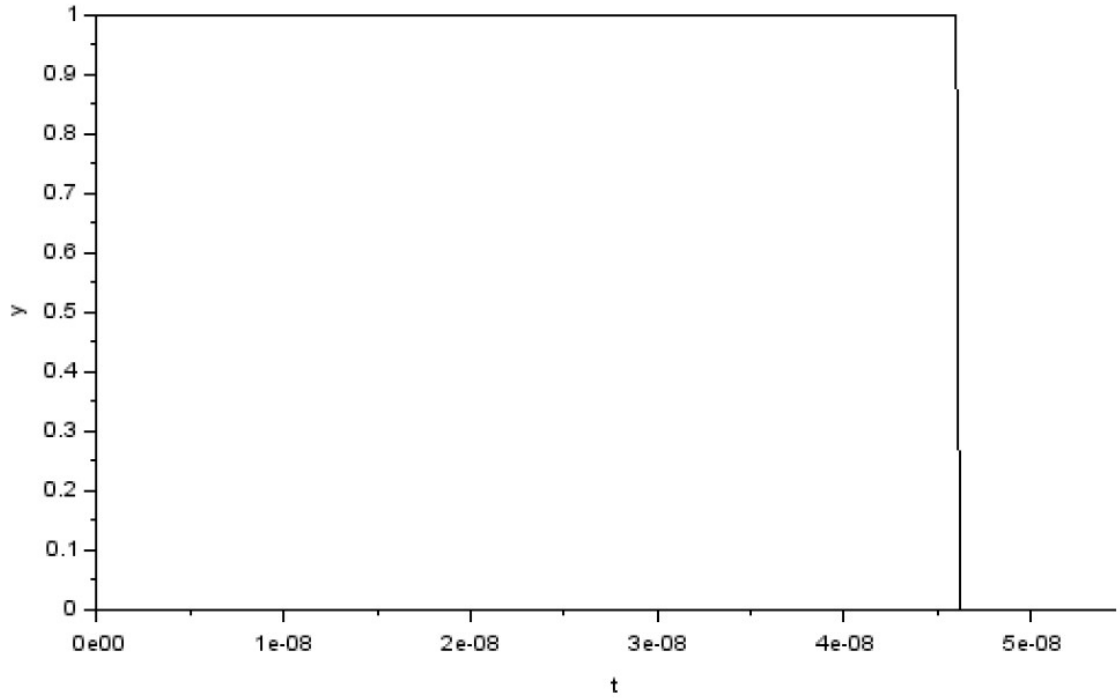


Pode-se ver o comportamento geral da onda PWM, em que há a variação constante de estado lógico alto e baixo quando o bit é 1 e completamento nulo quando o bit é 0.



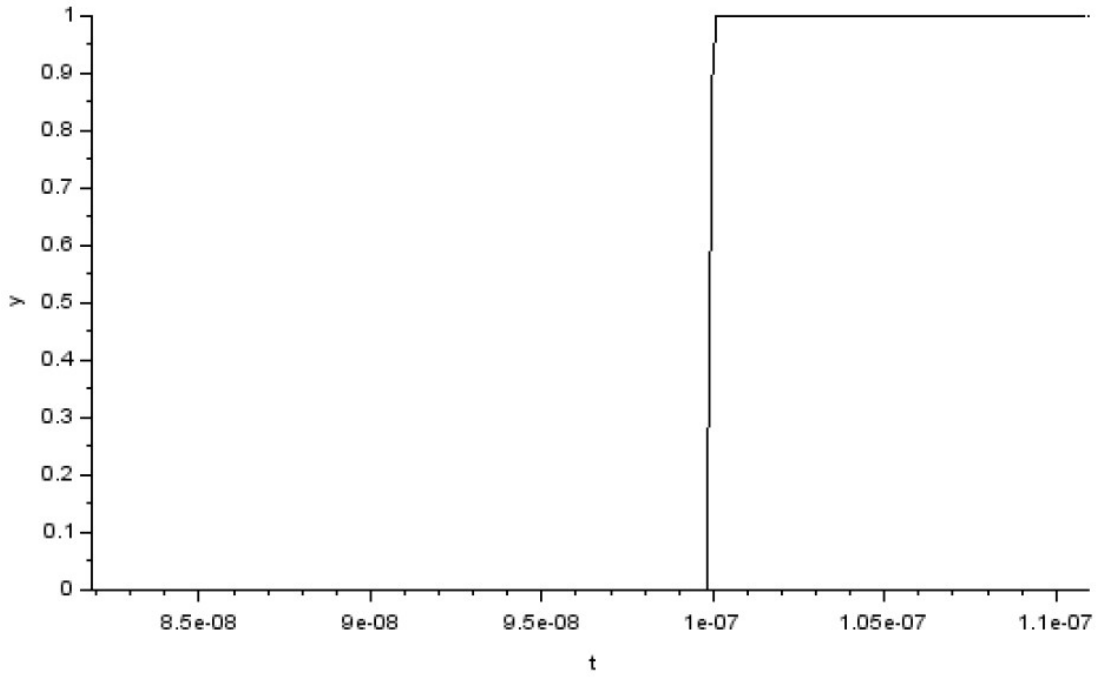
Aplicando novamente a comparação entre as proporções de tempo e valores já explicada anteriormente. Escolhendo mais uma vez como exemplo a primeira amostra, tem-se que o momento em que é mudado seu nível lógico alto para baixo no PWM, sendo este:





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

O instante de extrapolação do contador é:



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

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

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

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

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.