PDS-FPGAs

Práctica P2
Grupo 09

Moreno Suay, Sergio
Ruiz Escorihuela, Víctor Javier

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Índice

1. Descripción del módulo

ENUNCIADO 10

Breve explicación del bloque implementado indicando su funcionalidad y estructura interna. Se puede añadir (de apoyo a la explicación) una figura con el diagrama de bloques interno del modulo.

En esta práctica se ha implementado un Modulador configurable AM/FM, que tiene como objetivo modular una señal de entrada utilizando modulación de amplitud (AM) o modulación de frecuencia (FM). Esto dependerá del valor de la entrada `"ic_am_fm"`. En el caso de que el bit tenga un valor de 1, la modulación será FM; en caso contrario, que valga 0, la modulación será AM.

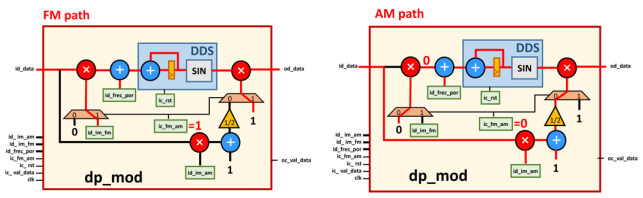


Figura 1: Esquema del bloque Modulador configurable AM/FM.

2. Interfaz

ENUNCIADO 11

Incluir las tablas en las que se definan el interfaz, sus formatos y sus parámetros.

Tabla 1: Interfaz del módulo DP_mod

NOMBRE	TIPO	FORMATO	DESCRIPCIÓN
id_data	in	S[16,15]	Señal de entada que se va a modular
id_im_am	in	U[16,15]	Índice de modulación AM
id_im_fm	in	U[16,16]	Índice de modulación FM
id_frec_por	in	U[24,24]	Frecuencia de la portadora
ic_fm_am	in	bit	Selección de la modulación AM o FM
ic_rst	in	bit	Reset del acumulador
ic_val_data	in	bit	Validación de muestras de entrada
clk	in	bit	Entrada de reloj
od_data	out	S[16,15]	Datos de salida modulados con modulación AM o FM
oc_val_data	out	bit	Señal de muestras validas de salida

3. Recursos hardware

ENUNCIADO 12

Incluir:

- *Una tabla detallando los recursos del dispositivo FPGA que se requieren para implementar el módulo. En esta práctica solo hay que indicar el número de LEs configurados en modo normal, LEs en modo aritmético, los FFs y memorias M9K.*
- *Una captura del resumen de recursos obtenidos con la herramienta Quartus tras la etapa de emplazamiento y rutado (Fitter), que se muestra en el report "Resource Usage Summary". En esta primera memoria se ha incluido una captura de este report como ejemplo, que deberá sustituirla por la que se obtenga en su proyecto.*

El objetivo de esta sección es que el alumno sea capaz de identificar si los recursos obtenidos en la implementación son coherentes con los esperables para el circuito realizado.

Tabla 2: Recursos hardware

RECURSO	NÚMERO OBTENIDO
FFs	174
LEs-normal	138
LEs-aritméticos	61
Mults 18x18	6
M9Ks	16

A continuación se muestra la captura realizada de los recursos obtenidos tras la etapa de emplazamiento y rutado (Fitter) en el “ Resource Usage Summary”:

	Resource	Usage
1	Total logic elements	240 / 114,480 (< 1 %)
1	-- Combinational with no register	66
2	-- Register only	41
3	-- Combinational with a register	133
2		
3	Logic element usage by number of LUT inputs	
1	-- 4 input functions	0
2	-- 3 input functions	76
3	-- <=2 input functions	123
4	-- Register only	41
4		
5	Logic elements by mode	
1	-- normal mode	138
2	-- arithmetic mode	61
6		
7	Total registers*	174 / 117,053 (< 1 %)
1	-- Dedicated logic registers	174 / 114,480 (< 1 %)
2	-- I/O registers	0 / 2,573 (0 %)
8		
9	Total LABs: partially or completely used	17 / 7,155 (< 1 %)
10	Virtual pins	0
11	I/O pins	93 / 529 (18 %)
1	-- Clock pins	2 / 7 (29 %)
2	-- Dedicated input pins	0 / 9 (0 %)
12		
13	M9Ks	16 / 432 (4 %)
14	Total block memory bits	131,072 / 3,981,312 (3 %)
15	Total block memory implementation bits	147,456 / 3,981,312 (4 %)
16	Embedded Multiplier 9-bit elements	6 / 532 (1 %)
17	PLLs	0 / 4 (0 %)
18	Global signals	1
1	-- Global clocks	1 / 20 (5 %)
19	JTAGs	0 / 1 (0 %)
20	CRC blocks	0 / 1 (0 %)
21	ASMI blocks	0 / 1 (0 %)
22	Oscillator blocks	0 / 1 (0 %)
23	Impedance control blocks	0 / 4 (0 %)
24	Average interconnect usage (total/H/V)	0.2% / 0.2% / 0.2%
25	Peak interconnect usage (total/H/V)	3.4% / 3.4% / 3.7%
26	Maximum fan-out	193
27	Highest non-global fan-out	167
28	Total fan-out	1389
29	Average fan-out	2.31

Figura 2: Report de “Resource Usage Summary”.

4. Frecuencia de operación

ENUNCIADO 13

La frecuencia máxima de operación del módulo implementado en el dispositivo y una indicación de dónde se encuentra el camino crítico del circuito.

Tabla 3: Frecuencia de reloj y camino crítico

f_{clk} (MHz)	CAMINO CRÍTICO
114.3 MHz	Desde la dirección registrada de la dds_rom hasta el postproc_data registrado.

5. Verificación

ENUNCIADO 14

Resultados de la verificación al excitarlo con diferentes entradas para demostrar su completo funcionamiento. Solo se incluirá:

- *La lista de las pruebas realizadas, que debe coincidir con los casos de test que se hayan configurado en el script de simulación de Matlab.*
- *Las capturas de las formas de onda y el resumen (mostrado en la consola de Modelsim) de la simulación realizada de un máximo de 2 casos, mostrando claramente el comportamiento del circuito y que no existen errores al compararlo con el modelo de Simulink.*

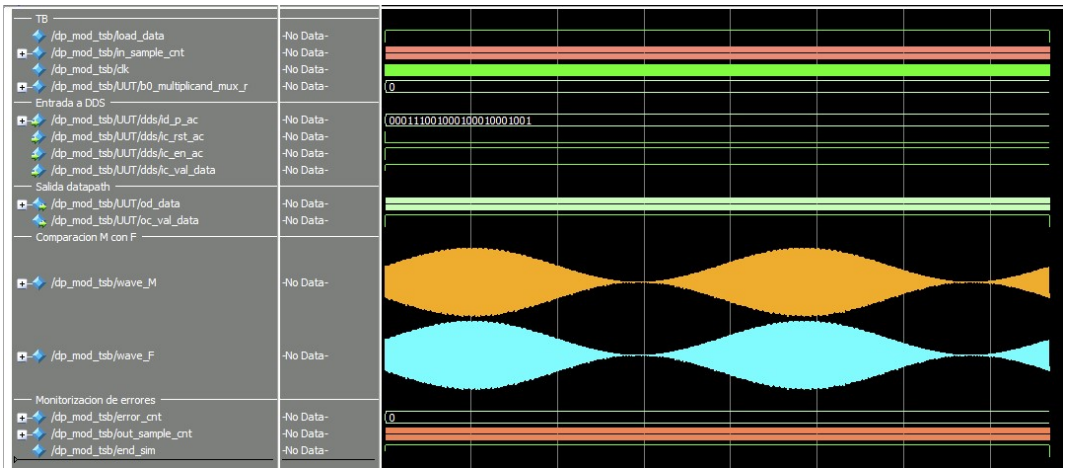
El módulo ha funcionado correctamente. Su funcionamiento se ha verificado con las siguientes pruebas:

1. Caso 1: conf_fm_am = 0, fmod = 5 kHz, fc = 10.7 MHz, im_am = (1-2^15), im_fm = 0 y n_periods_to_display = 2.
2. Caso 2: conf_fm_am = 1, fmod = 5 kHz, fc = 1 MHz, im_am = 0, im_fm = 700 y n_periods_to_display = 1.
3. Caso 3: conf_fm_am = 0, fmod = 20 kHz, fc = 40 MHz, im_am = (1-2^15), im_fm = 0 y n_periods_to_display = 10.
4. Caso 4: conf_fm_am = 1, fmod = 20 kHz, fc = 40 MHz, im_am = 0, im_fm = 500 y n_periods_to_display = 10.

A continuación, se describen dos de los escenarios de test enumerados anteriormente:

5.1. Caso 1:

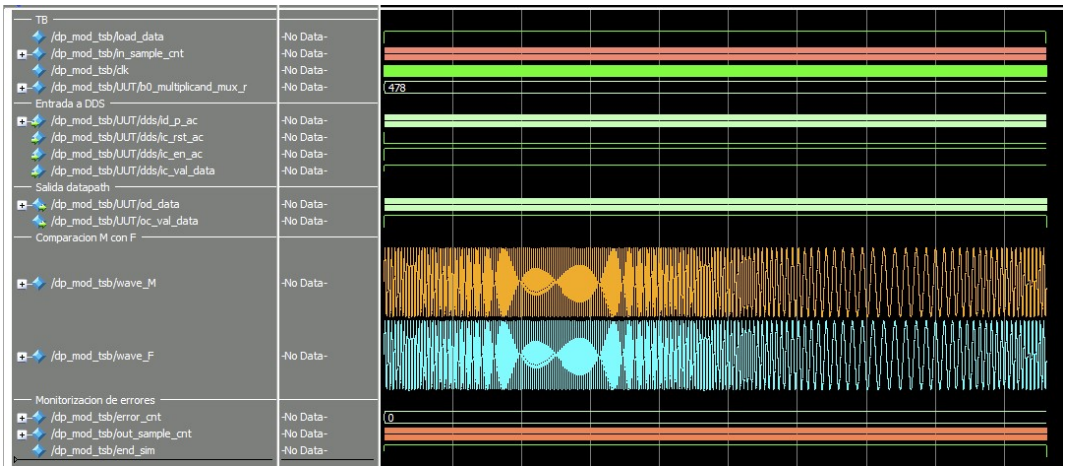
A continuación se muestra el test case 1: se ha configurado para que realice una modulación AM, conf_fm_am = 0. Con una frecuencia de modulación de 5kHz, fmod = 5, una portadora de 10,7 MHz, fc = 10.7, y un indice de modulacion AM de 1000, im_am = (1-2÷15) y im_fm = 0. Ademas de mostrar dos periodos por el display, n_periods_to_display = 2. Se ha realizado la simulación y no presenta ningún error entre la señal simulada y la del modelo de referencia. A continuación se muestran las diferentes señales y el resumen de la simulación mostrada en la consola de Modelsim.




```
############################################
# START TEST #           1
############################################
#  Number of input samples       38401
############################################
# TEST #           1
# M =          24
# L =          15
# W =          16
############################################
# Number of checked samples       38401
# Number of errors           0
############################################
```

5.2. Caso 2

A continuación se muestra el test case 2: se ha configurado para que realice una modulación FM, conf_fm_am = 1. Con una frecuencia de modulación de 5kHz, fmod = 5, una portadora de 1 MHz, fc = 1, y un indice de modulacion AM de 700, im_am = 0 y im_fm = 700. Ademas de mostrar un periodos por el display, n_periods_to_display = 1. Se ha realizado la simulación y no presenta ningún error entre la señal simulada y la del modelo de referencia. A continuación se muestran las diferentes señales y el resumen de la simulación mostrada en la consola de Modelsim.




```
############################################
# START TEST #          2
############################################
#  Number of input samples       19201
############################################
# TEST #          2
# M =          24
# L =          15
# W =          16
############################################
# Number of checked samples       19201
# Number of errors          0
############################################
```

6. Conclusiones

ENUNCIADO 15

Conclusiones de la práctica. Valore el grado de cumplimiento de los objetivos de la práctica. Haga un breve análisis de si los resultados obtenidos de área y frecuencia máxima son coherentes.

Hemos conseguido implementar el modulo de un modulador configurable AM/FM, el cuál nos permite modular una señal de entrada con una modulación AM o FM dependiendo de como se le indique. Nuestro diseño ha sido verificado comparando con un golden model generado en Simulink y Matlab. Hemos probado cuatro casos distintos y el diseño ha cumplido en todos ellos. El gasto de recursos lógicos coincide con lo esperado de nuestro diseño, y la frecuencia máxima de 114.3 MHz.

7. Ficheros entregados

ENUNCIADO 16

Lea el documento desarrollo y entrega de prácticas e indique cómo se han nombrado los ficheros que se indican en la tabla. En esta práctica esta tabla está por completar,

para que sirva de referencia para las siguientes prácticas.

Los autores de la memoria confirman que han entregado los ficheros siguientes, siguiendo las pautas indicadas:

FICHERO	NOMBRE
Memoria de prácticas	Memoria_P2_G9.pdf
Entrega de los ficheros de prácticas	Proy_sim_P2_G9.zip
Proyecto Modelsim	dp_mod.mpf
Fichero .do	dp_mod_tsb.do
Fichero .tcl	dp_mod_tsb.tcl
Proyecto Quartus	dp_mod.qpf

8. ANEXO 1: Código HDL del módulo dp_mod

ENUNCIADO 17

Incluir un esquema de la división en bloques realizada y el código Verilog de los módulos de la práctica. El código debe ser legible, ordenado, bien tabulado y comentado. El código debe seguir las pautas indicadas en el documento desarrollo y entrega de prácticas.

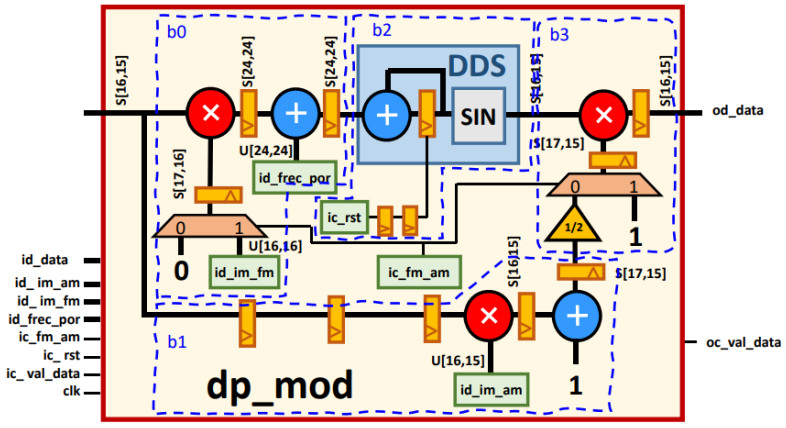


Figura 3: Esquema de la división en bloques realizada del bloque Modulador configurable AM/FM.

```
module dp_mod
  (
  input signed [15:0] id_data,  // S[16,15]
  input [23:0] id_frec_por,    // U[24,24]
  input [15:0] id_im_am,      // U[16,15]
  input [15:0] id_im_fm,      // U[16,16]
  input ic_fm_am,        // Control modo fm/am
  input ic_rst,          // rst sincrono activo a 1
  input ic_val_data,
  input clk,
  output signed [15:0] od_data,  // S[16,15]
  output oc_val_data
  );

/* DECLARACIONES ------------------------- */

// b0: ruta datos FM
logic signed [16:0] b0_multiplicand_mux_r; // S[17,16]
logic [32:0] b0_mult_res_full_s; // S[33,31]
logic signed [23:0] b0_mult_res_r; // S[24,24]
logic signed [24:0] b0_sum_res_extended_s; // S[25,24]
logic signed [23:0] b0_sum_res_r; // S[24,24]

// b1: ruta de datos AM
logic [2:0][15:0] b1_id_data_delayed_r; // Each element S[16,15]
logic signed [32:0] b1_mult_res_full_s; // S[33,30]
logic signed [15:0] b1_mult_res_r; // S[16,15]
logic signed [16:0] b1_mult_res_trunc_extend_s; // S[17,15]
logic signed [16:0] b1_mult_res_added_r; // S[17,15]

// b2: DDS
logic [1:0] b2_rst_delayed_r;
logic signed [15:0] b2_sine_s; // S[16,15]

// b3: etapa final
logic signed [16:0] b3_multiplicand_mux_r; // S[17,15]
logic signed [32:0] b3_mult_res_full_s; // S[33,30]
logic signed [15:0] b3_mult_res_r; // S[16,15]

// b4: Generacion  de oc_val_data
logic [6:0] b4_delayed_val_data_r;

/* DESCRIPCION ------------------------- */

// b0: ruta datos FM
always_ff @(posedge clk) begin
  if(ic_fm_am) begin
    b0_multiplicand_mux_r <= $signed({1'b0, id_im_fm});
  end else begin
    b0_multiplicand_mux_r <= 17'd0;
  end
end

always_ff @(posedge clk) begin
```

```
  if(ic_rst) begin
    b0_mult_res_r <= 0;
    b0_sum_res_r <= 0;
  end else begin
    b0_mult_res_full_s = id_data * b0_multiplicand_mux_r;
    b0_mult_res_r <= b0_mult_res_full_s[30:30-23];
    b0_sum_res_extended_s = {b0_mult_res_r[23], b0_mult_res_r} + $signed({1'b0, id_frec_por});
    b0_sum_res_r <= b0_sum_res_extended_s[23:0];
  end
end

// b1: ruta de datos AM
always_ff @(posedge clk) begin
  if(ic_rst) begin
    b1_id_data_delayed_r <= 0;
    b1_mult_res_r <= 0;
    b1_mult_res_added_r <= 0;
  end else begin
    b1_id_data_delayed_r <= {b1_id_data_delayed_r[1:0], id_data};
    b1_mult_res_full_s = $signed(b1_id_data_delayed_r[2]) * $signed({1'b0, id_im_am});
    b1_mult_res_r <= b1_mult_res_full_s[30:30-15];
    b1_mult_res_trunc_extend_s = b1_mult_res_r;
    b1_mult_res_added_r <= b1_mult_res_trunc_extend_s + $signed({1'b0, 1'b1, 15'b0});
  end
end

// b2: DDS
always_ff @(posedge clk) begin
  if(ic_rst) begin
    b2_rst_delayed_r <= {ic_rst, ic_rst};
  end else begin
    b2_rst_delayed_r <= {b2_rst_delayed_r[0], ic_rst};
  end
end

dds_mod_dds #(.M(24),.L(15),.W(16)) dds
       (.id_p_ac(b0_sum_res_r),
      .ic_rst_ac(b2_rst_delayed_r[1]),
      .ic_en_ac(!b2_rst_delayed_r[1]),
      .ic_val_data(1),
      .clk(clk),
      .od_sin_wave(b2_sine_s),
      .oc_val_data()
      );

// b3: Etapa final
always_ff @(posedge clk) begin
  if(ic_rst) begin
    b3_multiplicand_mux_r <= 17'b0;
    b3_mult_res_r <= 0;
  end else begin
    if(ic_fm_am) begin
      b3_multiplicand_mux_r <= {1'b0, 1'b1, 15'b0};
    end else begin
```

```
      b3_multiplicand_mux_r <= b1_mult_res_added_r >>> 1;
    end

    b3_mult_res_full_s = b2_sine_s * b3_multiplicand_mux_r;
    b3_mult_res_r <= b3_mult_res_full_s[30:30-15];
  end
end

// b4: propagacion de val_data
always_ff @(posedge clk) begin
  if(ic_rst) begin
    b4_delayed_val_data_r <= 0;
  end else begin
    b4_delayed_val_data_r <= {b4_delayed_val_data_r[5:0], ic_val_data};
  end
end

/* ASIGNACION SALIDAS ------------------------ */
assign od_data = b3_mult_res_r;
assign oc_val_data = b4_delayed_val_data_r[6]; // HAY QUE MODIFICARLO

endmodule
```

9. ANEXO 2: Código HDL del banco de pruebas dp_mod_tsb

ENUNCIADO 18

Incluir el código Verilog del banco de pruebas realizado.

```
`timescale 1ns/1ps
import dp_mod_tsb_pkg::*; // PACKAGE WITH PARAMETERS TO CONFIGURE THE TB

module dp_mod_tsb();

parameter PER=10; // CLOCK PERIOD
parameter M= 24;
parameter L= 15;
parameter W = 16;

logic rst_ac;
logic clk;
logic val_in;
logic conf_fm_am;
logic [23:0] p_frec_por;
logic [15:0] im_am;
logic [15:0] im_fm;
logic val_out;
logic signed [15:0] in_data;
logic signed [15:0] out_data;

logic signed [15:0] wave_M, wave_F;
```

```
// contadores y control
integer in_sample_cnt; // Contador de muestras de entrada
integer out_sample_cnt; // Contador de muestras de salida
integer error_cnt; // Contador de errores

logic load_data;  // Inicio de lectura de datos
logic end_sim; // Indicación de simulación on/off

// Gestion I/O texto
integer data_in_file_val;
logic signed [15:0] data_in_file;
integer scan_data_in;

integer config_file_val;

integer data_out_file_val;
logic signed [15:0] data_out_file;
integer scan_data_out;

// Reloj
always #(PER/2) clk = !clk&end_sim;

// UUT
dp_mod UUT
       (.id_data(in_data),
      .id_frec_por(p_frec_por),
      .id_im_am(im_am),
      .id_im_fm(im_fm),
      .ic_fm_am(conf_fm_am),
      .ic_rst(rst_ac),
      .ic_val_data(val_in),
      .clk(clk),
      .od_data(out_data),
      .oc_val_data(val_out)
      );

initial
  begin
    $display("############################################ ");
    $display("START TEST # ","%d", test_case);
    $display("############################################ ");
    data_in_file_val = $fopen("./iof/id_dp_mod.txt", "r");

    assert (data_in_file_val) else begin
      $display("---> Error opening file id_dp_mod.txt");
      $stop;
    end

    data_out_file_val = $fopen("./iof/od_dp_mod.txt", "r");
    assert (data_out_file_val) else begin
      $display("---> Error opening file od_dp_mod.txt");
      $stop;
    end
```

```
    config_file_val = $fopen("./iof/id_config_dp_mod.txt", "r");

    $fscanf(config_file_val, "%d\n", p_frec_por);
    $fscanf(config_file_val, "%d\n", im_am);
    $fscanf(config_file_val, "%d\n", im_fm);
    $fscanf(config_file_val, "%d\n", conf_fm_am);

    $fclose(config_file_val);
    //-----------------------------------------------------

    in_data = 0;
    error_cnt = 0;
    out_sample_cnt = 0;

    wave_M = 0;
    wave_F = 0;

    //-----------------------------------------------------
    end_sim = 1'b1;
    in_sample_cnt = 0;
    clk = 1'b1;
    val_in = 1'b0;
    rst_ac = 1'b1;
    load_data = 1'b0;
    #(10*PER);
    load_data = 1'b1;
  end

// Proceso de lectura de datos de entrada
always@(posedge clk)
     if (load_data)
         begin
      if (!$feof(data_in_file_val))
        begin
        in_sample_cnt = in_sample_cnt + 1;
        scan_data_in = $fscanf(data_in_file_val, "%b", data_in_file);
        in_data <= #(PER/10) data_in_file; //Salida del fichero
        rst_ac  <= #(PER/10) 1'b0;
        val_in  <= #(PER/10) 1'b1;
        end
      else
        begin
        val_in <=   #(PER/10) 1'b0;
        load_data =   #(PER/10) 1'b0;
        end_sim = #(10*PER) 1'b0; // Hay que dejar un numero de peridodos
                                  // mayor que la latencia del UUT
        $display(" Number of input samples ","%d", in_sample_cnt);

        end

    end

always@(posedge clk)
       if (val_out)
```

```
      begin
        out_sample_cnt = out_sample_cnt +1;
        if (!$feof(data_out_file_val))
          begin
          scan_data_out = $fscanf(data_out_file_val, "%b", data_out_file);
          wave_F <= #(PER/10) data_out_file; //Salida del fichero
          wave_M <= #(PER/10) out_data; //Salida del modulo UUT
          end
        else
          end_sim = #(10*PER) 1'b0;
      end

// Contador de errores y muestras
always@(wave_F,wave_M)
  Assert_error_out:
    assert (wave_F==wave_M)
      //$display("OK ","%d", sample_cnt);
      else begin
        error_cnt = error_cnt + 1;
        $display("Error in sample number ","%d", out_sample_cnt);
        end

// Fin de simulación
always@(end_sim)
  if (!end_sim)
    begin

      $display("################################################ ");
      $display("TEST # %d", test_case);
      $display("M = %d", M);
      $display("L = %d", L);
      $display("W = %d", W);
      $display("################################################ ");
      $display("Number of checked samples ","%d", out_sample_cnt);
      $display("Number of errors ","%d", error_cnt);
      $display("################################################ ");
      #(PER*2) $stop;

    #(PER*2) $stop;
    end



endmodule
```