DISEÑO AVANZADO DE HARDWARE
FACULTAD DE INGENIERÍA ELECTRÓNICA
UNIVERSIDAD PONTIFICIA BOLIVARIANA
BUCARAMANGA

Sebastian Augusto Baquero Peña ID:244379

"GENERATE"

OBJETIVO GENERAL

Implementación y uso de funciones con "generate".

OBJETIVOS ESPECÍFICOS

- Entender el funcionamiento de la función "generate".
- Conocer la estructura utilizada para realizar un "archivo generate".
- .Visualizar en RTL la estructura física de un archivo "generate".
- Conocer los límites en la implementación de un módulo "generate".

MARCO TEÓRICO

Un archivo "generate" se utiliza en especial para conectar varios registros sin necesidad que definirlos uno por uno. Es decir, cuando se desea hacer un módulo y se necesita que haya cierta cantidad "x" de registros conectados en cascada, no hacer la conexión entre cada uno de éstas línea a línea con la utilización del "generate" se puede realizar la conexión de estas utilizando lógica convencional. Es también utilizado para realizar un loop de lo que se necesite.

Preguntas

1. ¿Que se genera el RTL?

Si se realiza el código generate y luego se procese de generar el RTL aparece lo que se muestra a continuación en la Figura 1. Correspondiente a las entradas y salidas creadas en "System Builder" y el modulo creado que en mi caso se llama "generate1".

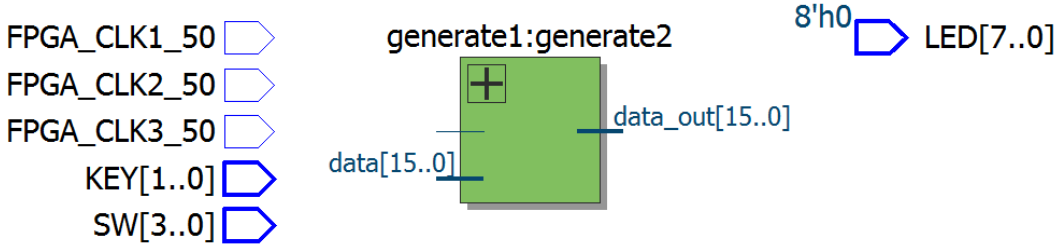


Figura 1. Visualización en el RTL

Ya analizando en el interior del módulo “generate 1” se observa la conexión en cascada de los registros o elementos.

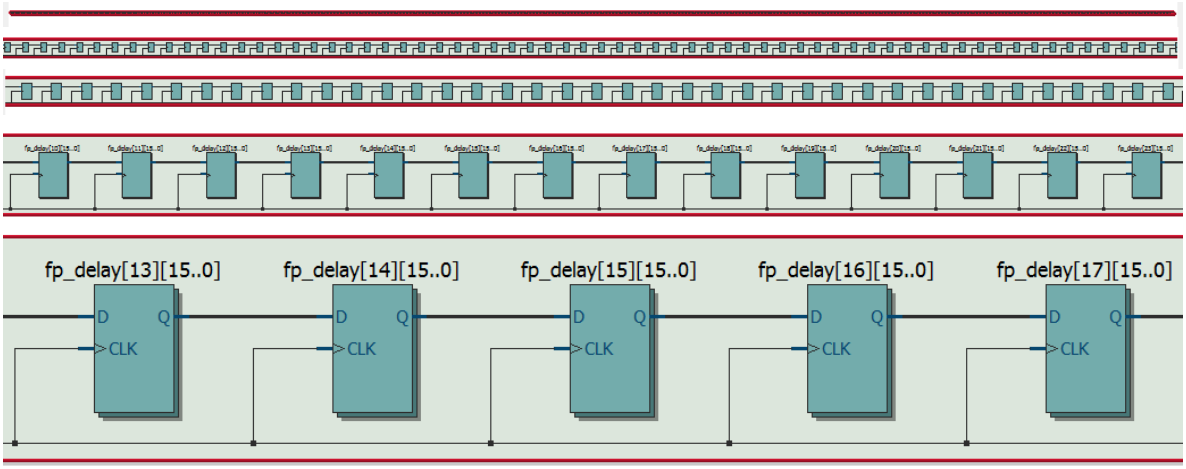


Figura 2. Estructura interna del archivo “generate1”.

2. ¿Para qué es útil el generate?

El generate es útil para realizar conexiones en cascada de cierta cantidad limitada de elementos o registros del mismo tipo o clase, de tal manera que se utilicen menos líneas y se logre optimizar más el tiempo tanto de elaboración como de ejecución.

3. Modifique el parámetro local "NUMBER_OF" a 100 y vuelva a generar el RTL.

Luego de haber realizado un valor cualquiera, se cambió este valor por 100 y se volvió a generar el RTL, mostrando lo siguiente.

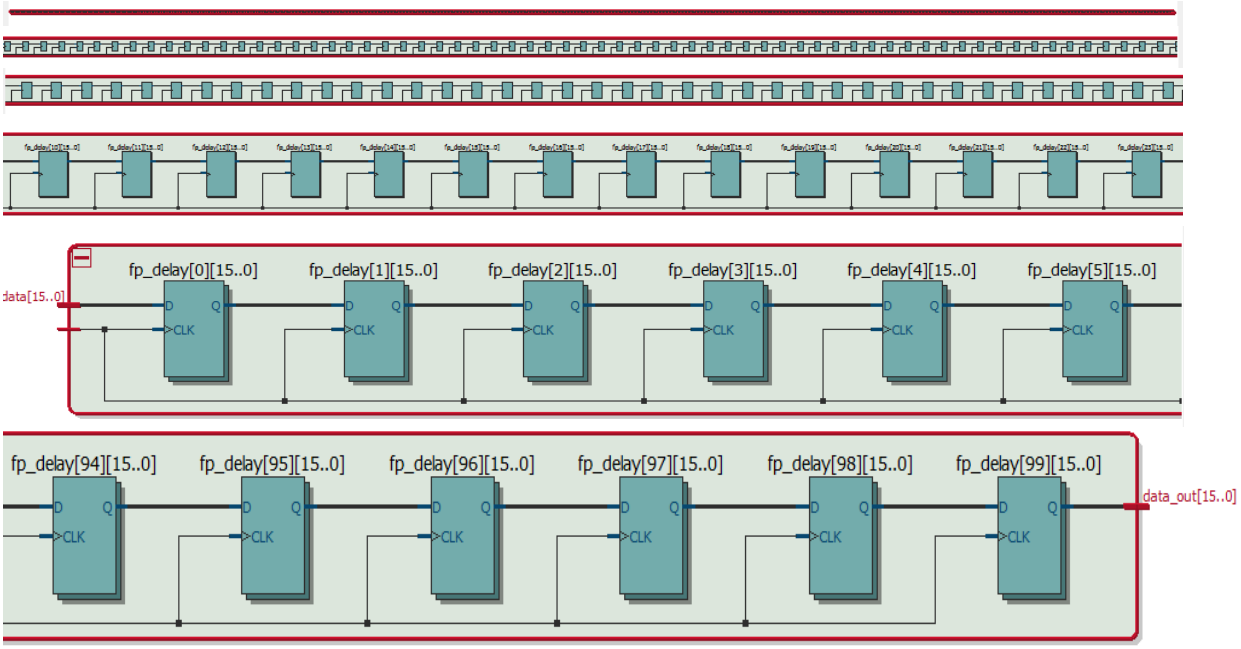


Figura 3. RTL con parámetro “NUMBER_OF” DE 100

4. Modifique el parámetro local "NUMBER_OF" a 1000, 2000, 3000, 4999, 5000, 10000 y vuelva a generar el RTL.
   a) ¿Qué sucede?

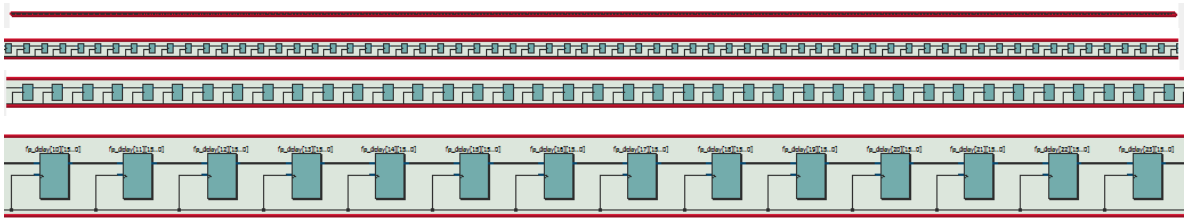


Figura 4. Visualización general de cada parametro que se puede generar con RTL.

NUMBER_OF de 1000:

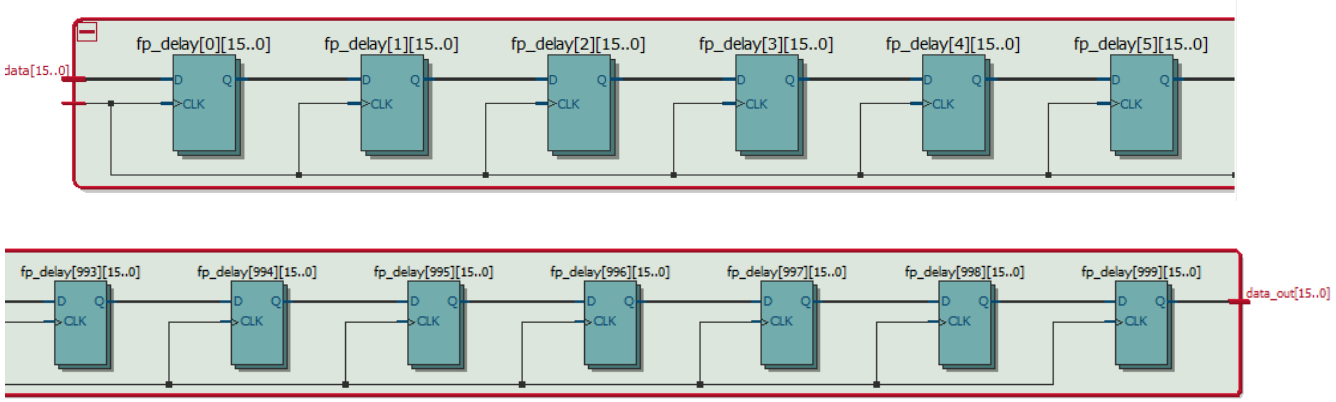


Figura 5. Visualización RTL de los 1000 flip-flops tipo D en cascada.

NUMBER_OF de 2000:

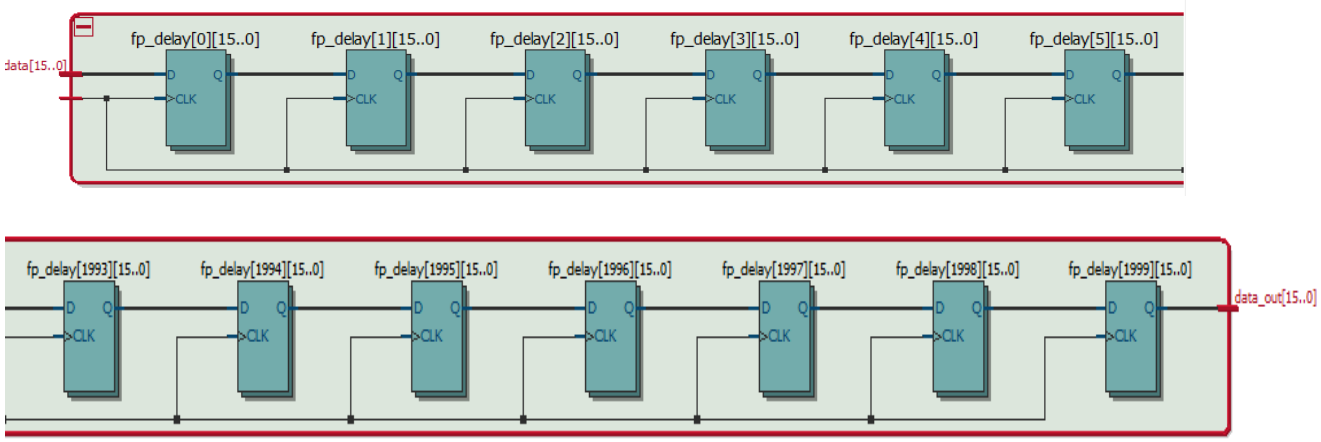


Figura 6. Visualización RTL de los 2000 flip-flops tipo D en cascada.

NUMBER_OF de 3000:

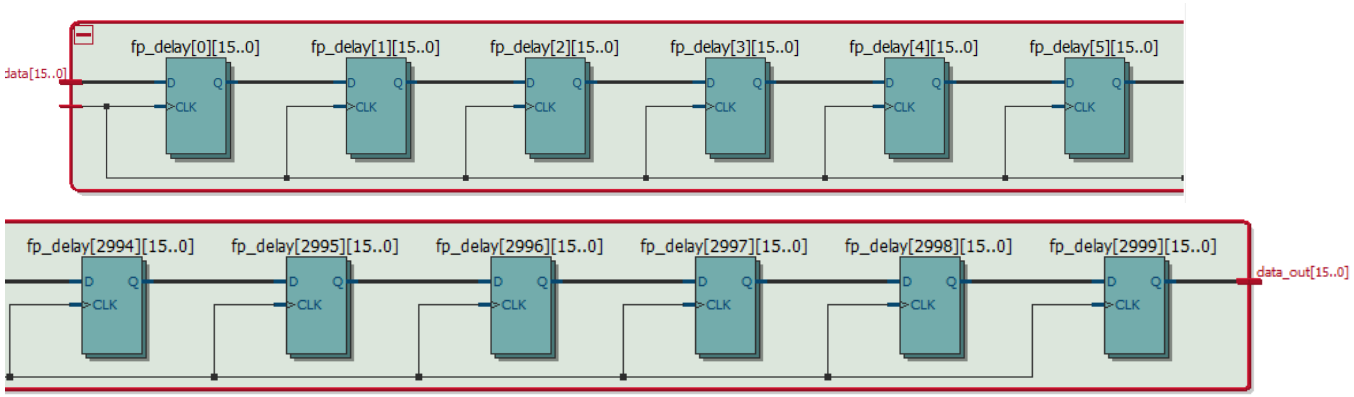


Figura 7. Visualización RTL de los 3000 flip-flops tipo D en cascada.

NUMBER_OF de 4999:

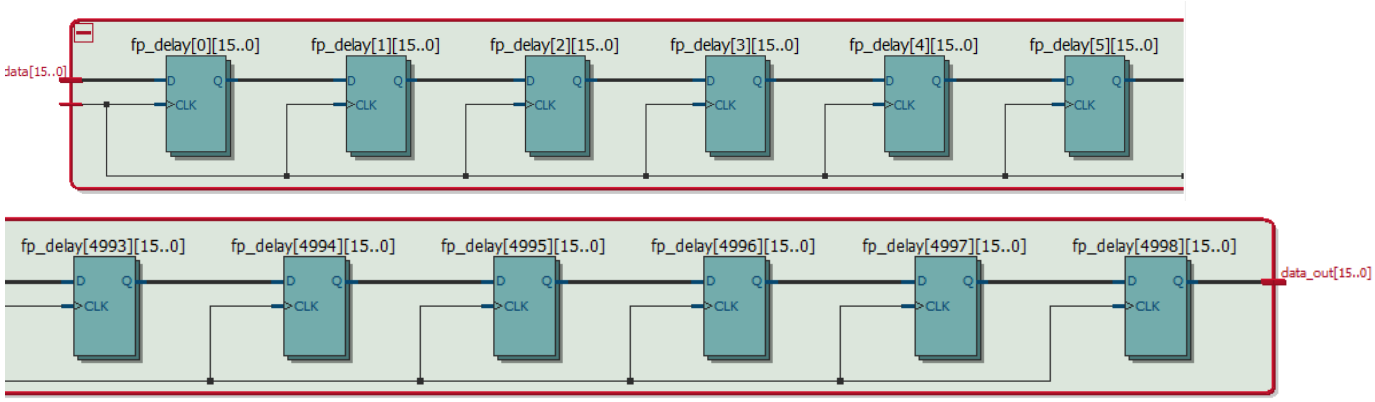


Figura 8. Visualización RTL de los 4999 flip-flops tipo D en cascada.

NUMBER_OF de 5000:

Con esta cantidad de elementos, el programa no acepta esa cantidad porque supera su capacidad.

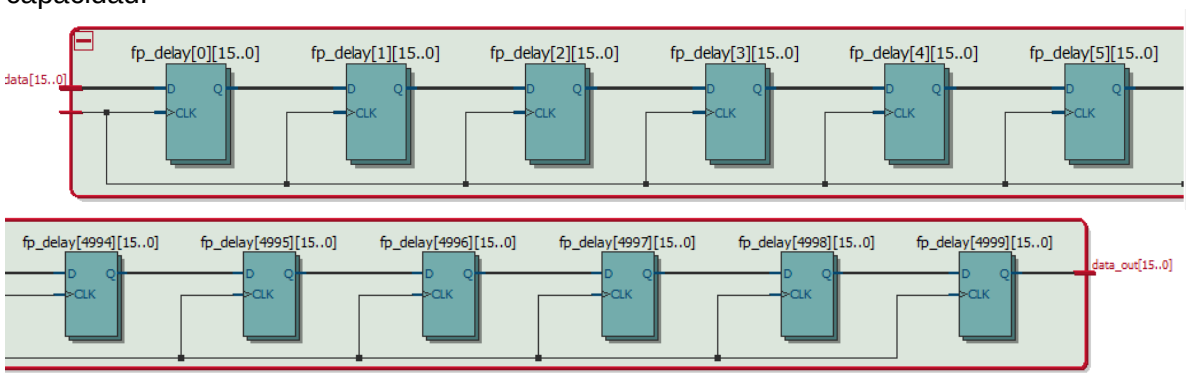


Figura 9. Visualización RTL de los 5000 flip-flops tipo D en cascada.

NUMBER_OF de 10000:

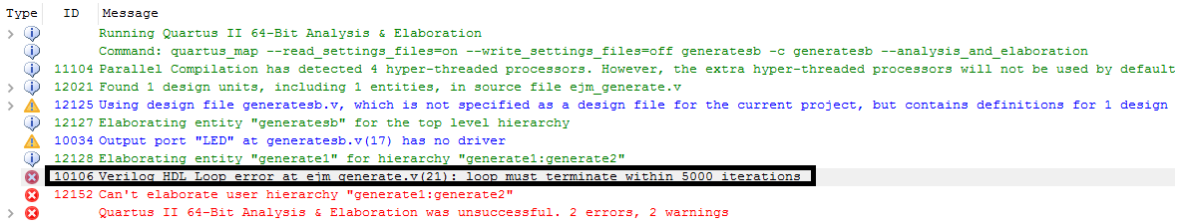


Figura 10. Error en la implementación del RTL.

Al igual que el valor anterior, el archivo o modulo “generate1” no es capaz de conectar 10000 elementos en cascada debido a que ha pasado el límite máximo de elementos posibles encontrado anteriormente.

Nota: A medida que se aumenta el número de elementos o registros que se van a conectar en cascada, aumenta el tiempo que se demora Eclipse en generar el RTL.

b) ¿Cuál es el límite de generate?

El límite máximo en el cual se pueden conectar en cascada varios elementos o registros es de 500, después de dicho valor no se puede realizar conexiones en cascada, el programa no lo acepta porque se ha superado su capacidad.

5. ¿Qué hace el circuito que está describiendo?

Se crean primeramente los elementos de entrada y salida que se van a utilizar en el módulo que en este caso van a ser: clock, data y dataout; que corresponden a dos entradas y una salida respectivamente.

Una vez se hayan definido estos parámetros, se procede a utilizar el “generate” para ello se crean parámetros locales los cuales nos van a permitir utilizarlos en la programación lógica. Después se crea un registro con los parámetros creados anteriores, en el cual uno de ellos corresponde a la cantidad de registros a conectar y el otro parámetros establece la cantidad de bits de entrada que se van a utilizar.

Seguidamente se le asigna la unión en cascada de una cantidad “x” al parámetro de salida que en este caso es “dataout” con la misma cantidad de bits de entrada.

Luego se crea el procedimiento del registro que depende si el flanco del reloj (“clock”) es positivo ejecuta la acción. Cuando se ha definido el proceso se implementa el archivo “genearte” creando una variable a utilizar (“index”) la cual va a ser utilizada para realizar las iteraciones en el ciclo “for” tomando el parámetro de la cantidad de elementos (“NUMBER_OF”), y dentro de este ciclo for se realiza el proceso de conexión de un elemento con el siguiente que en esta práctica se realizó conexión de flip-flops tipo D.

Analizando el funcionamiento global del circuito se observó que funciona como un divisor de frecuencia pero con frecuencia de salida en número entero que dependiendo de la cantidad de flip-flops que se conecten en cascada será dicha frecuencia.

```
module generate1(data, clock, dataout);

input clock;
input [15:0]data;
output [15:0]dataout;

localparam NUMBER_OF=2400;
localparam BUS_SIZE=16;
reg [BUS_SIZE-1:0]fp_delay[0:NUMBER_OF-1];

assign dataout[15:0]=fp_delay[NUMBER_OF-1][BUS_SIZE-1:0];

always@(posedge clock)
begin
    fp_delay[0][BUS_SIZE-1:0]<=data[15:0];
end

genvar index;
generate

for (index=NUMBER_OF-1; index >= 1; index=index-1)
    begin: delay_generate
            always@(posedge clock)
                begin
                    fp_delay[index][BUS_SIZE-1:0]<=fp_delay[index-1][BUS_SIZE-1:0];
                end

    end
endgenerate


endmodule
```

Figura 11. Código Verilog de un archivo con “generate”.

6. Intente lo mismo instanciando un módulo dentro del bloque generate.

Instanciación del módulo dentro del bloque “generate1”

Se realizó el mismo procedimiento instanciando el modulo dentro del bloque generate y arroja el mismo funcionamiento sin instanciarlo, para ello es necesario definir las entradas y salidas para que se conecten al módulo.

```
module generatesb(

    ////////////// CLOCK //////////
    input                       FPGA_CLK1_50,
    input                       FPGA_CLK2_50,
    input                       FPGA_CLK3_50,

    ////////////// KEY //////////
    input             [1:0]     KEY,

    ////////////// LED //////////
    output            [7:0]     LED,

    ////////////// SW //////////
    input             [3:0]     SW
);



//=======================================================
//  REG/WIRE declarations
//=======================================================


//=======================================================
//  Structural coding
//=======================================================
generate1 generate2
(
    .clock(FPGA_CLK1_50) ,  // input  clock_sig
    .data(16'd64000) ,    // input [15:0] data_sig
    .data_out(LED[7:0])   // output [15:0] data_out_sig
);
endmodule
```

Figura 12. Código principal con la instanciación del “generate”.

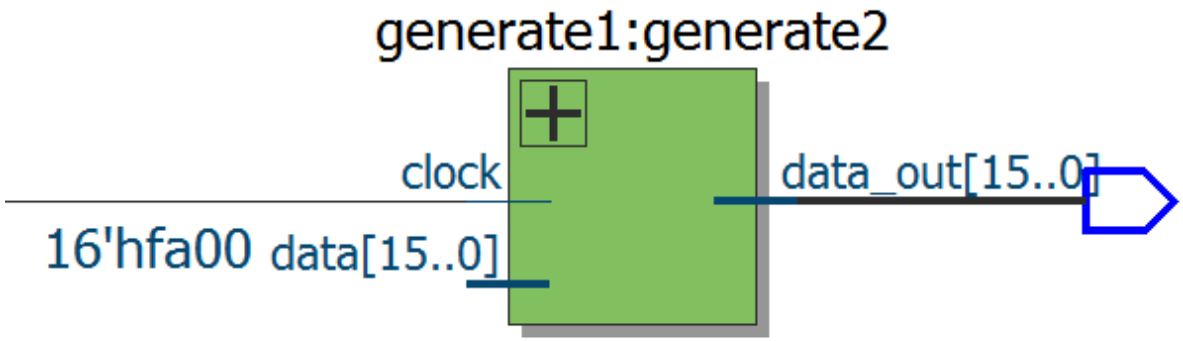


Figura 13. Modulo “generate1” instanciado con “generate2”

En la Figura 13. Se ha conectado el módulo al clock de la FPGA, además se le ingresa un dato de 16 bits y se define una salida, como la cantidad de bit es grande sólo se pueden visulizar 8 bits correspondiente a la cantidad de LEDs que posee la FPGA DE0 SOC.

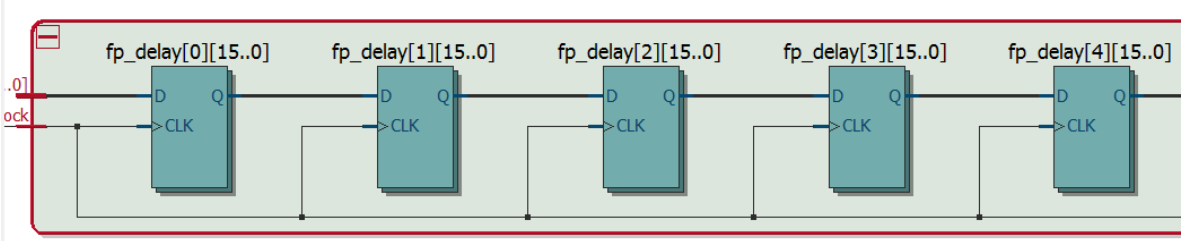
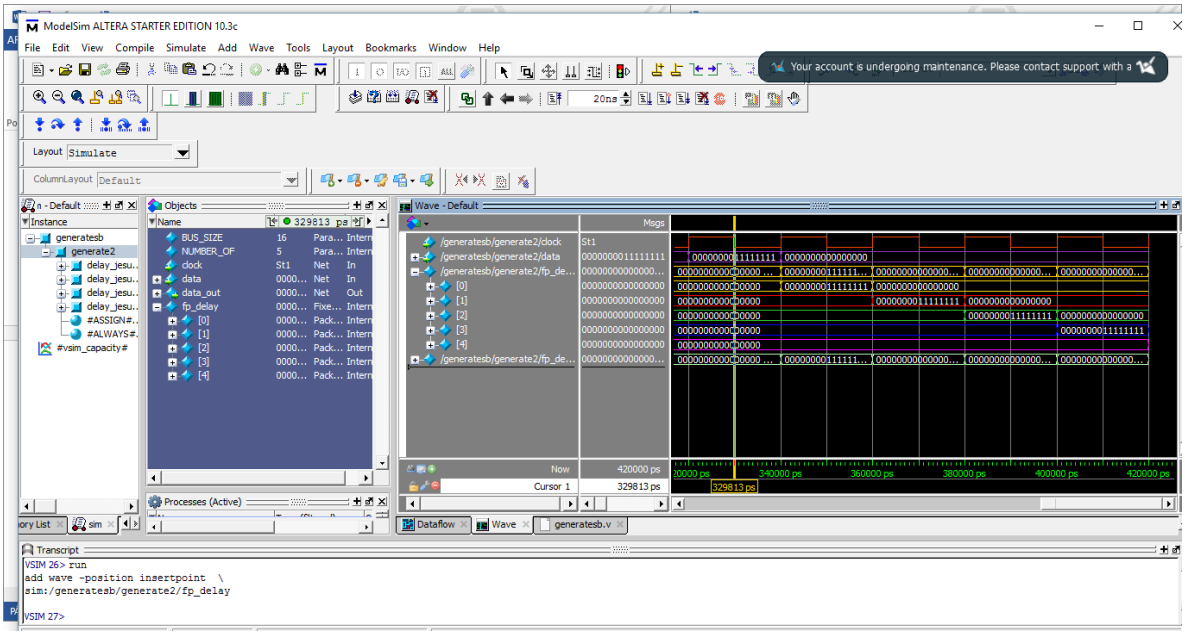
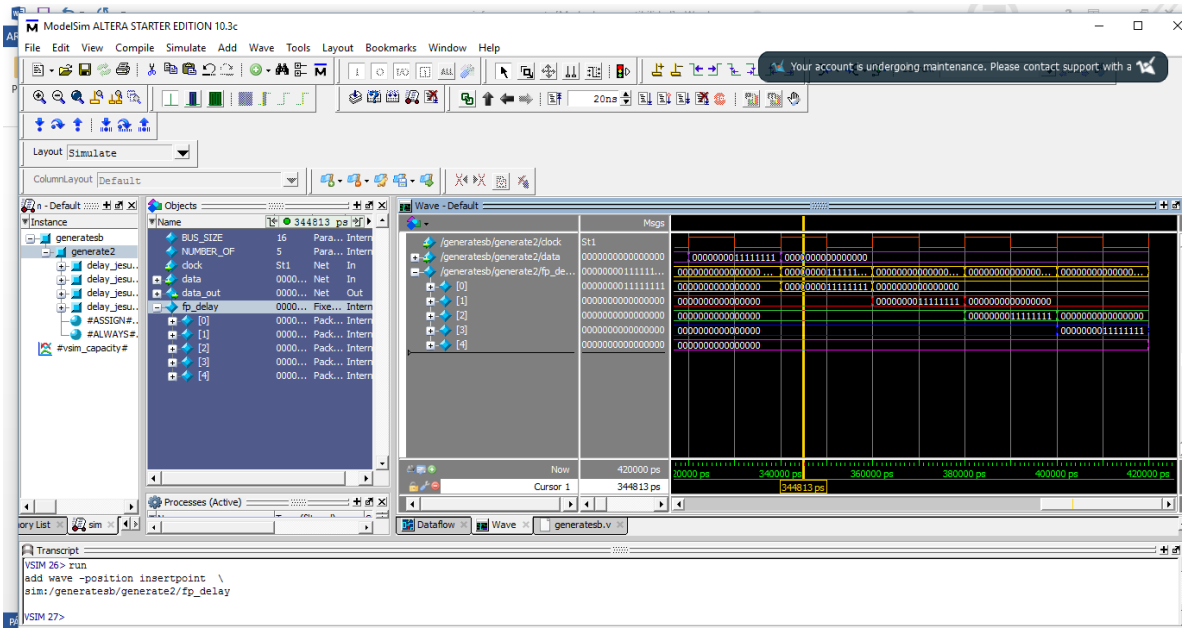


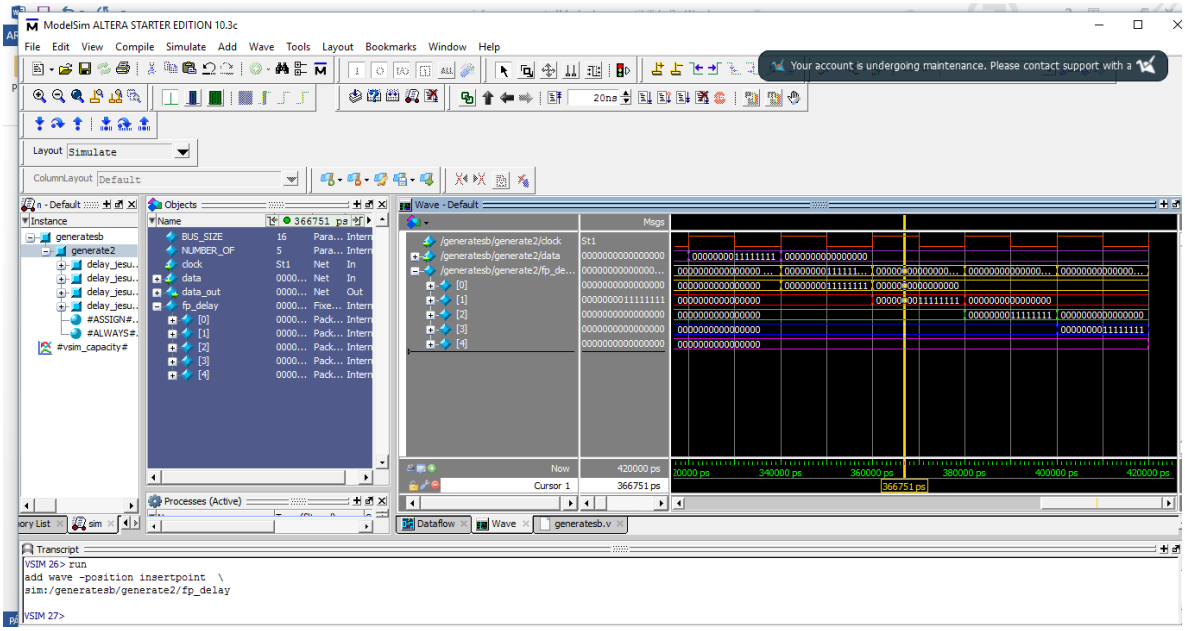
Figura 14. Interior del Módulo Generate1

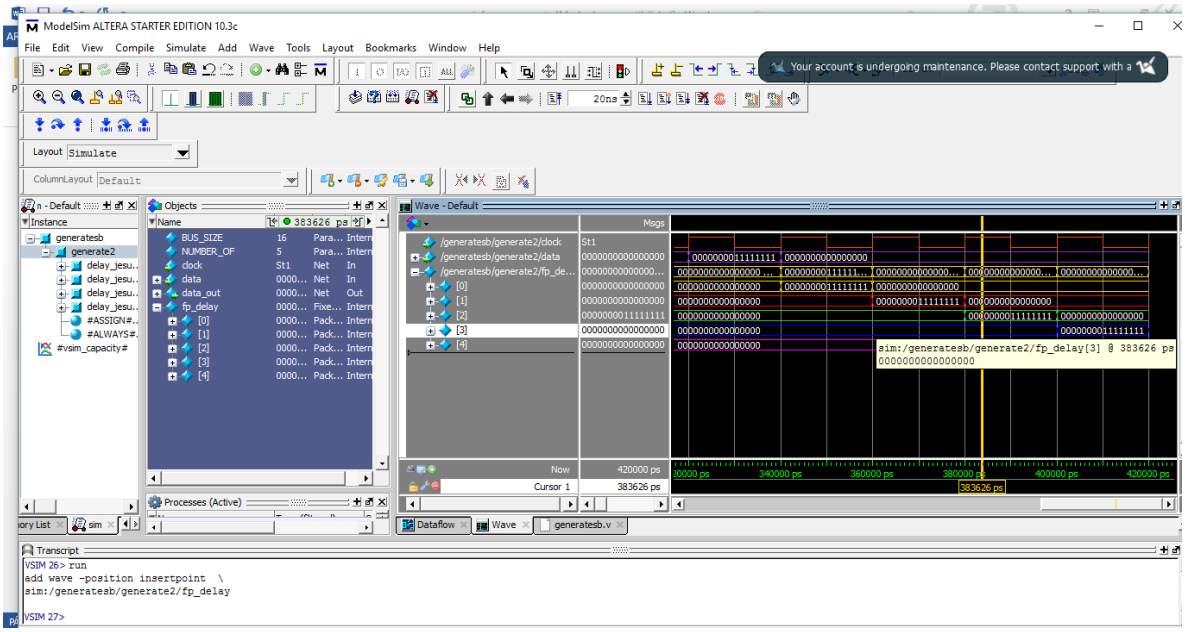
Como se observa en la Figura 14. al igual que el procedimiento anterior se observa la conexión en cascada de cada registro que va depender de la cantidad de registro que deseemos realizar teniendo el cuenta el límite superior anteriormente encontrado.

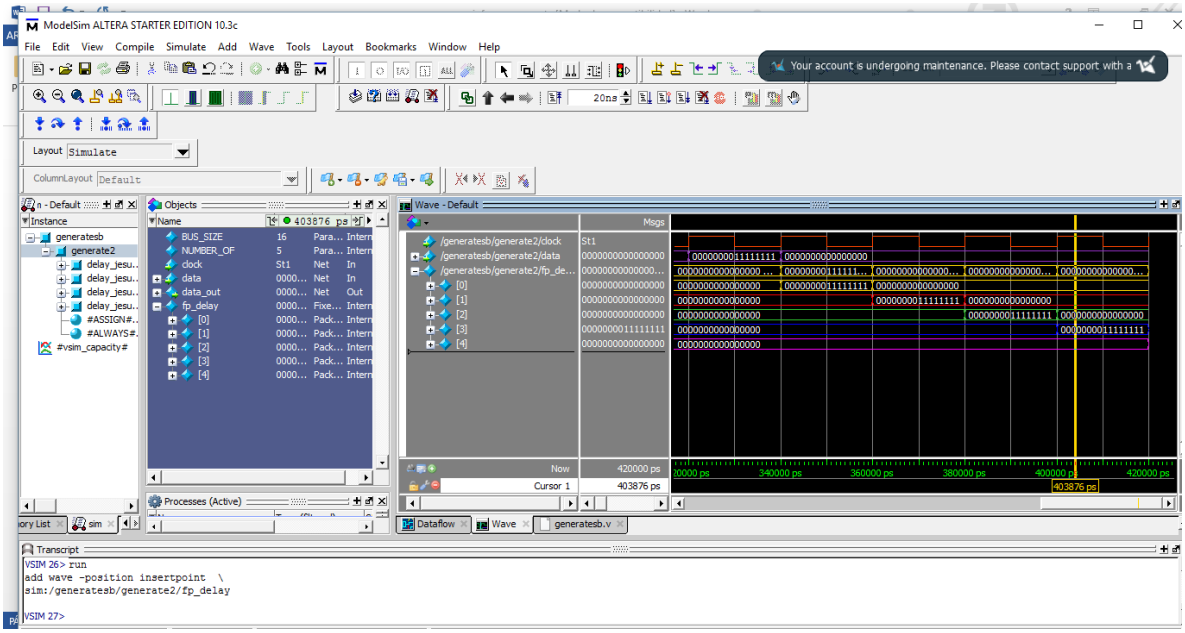
ModelSim

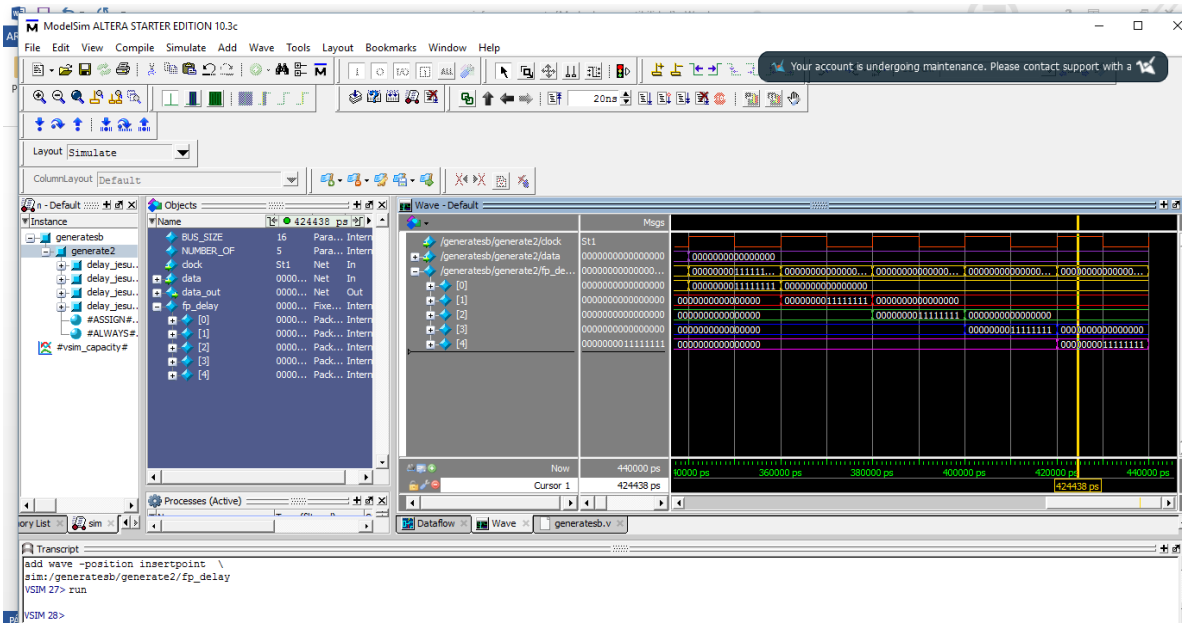


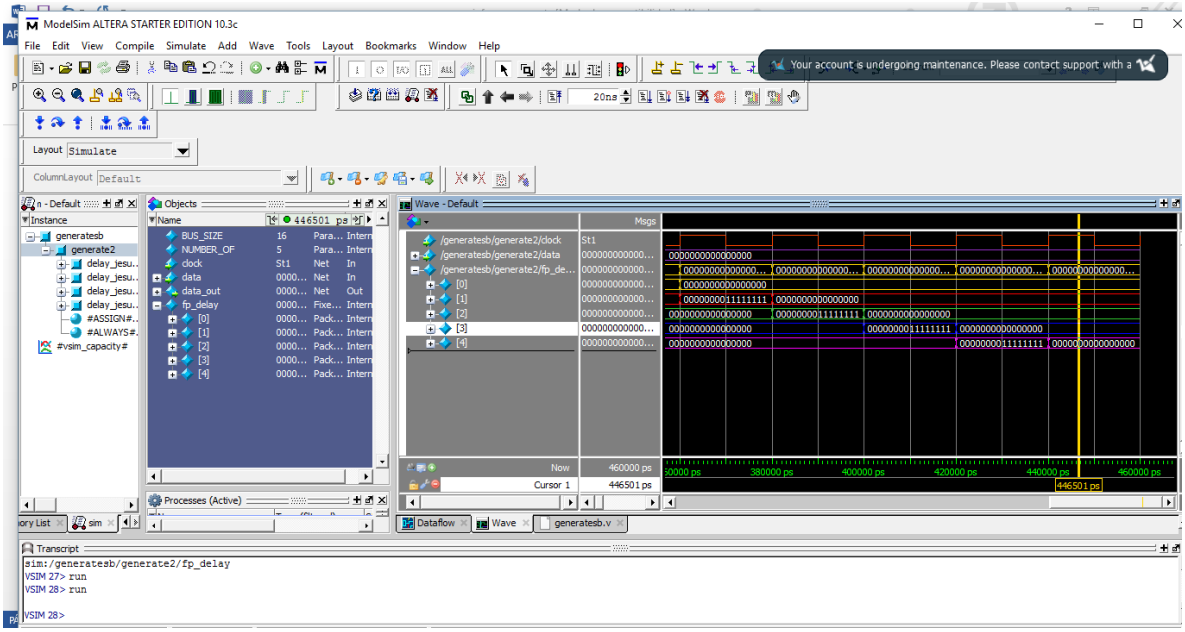

ModelSim ALTERA STARTER EDITION 10.3c
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Your account is undergoing maintenance. Please contact support with a
Transcript
VSIM 26> run
add wave -position insertpoint \
sim:/generatesb/generate2/fp_delay
VSIM 27>


ModelSim ALTERA STARTER EDITION 10.3c
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Your account is undergoing maintenance. Please contact support with a
Transcript
VSIM 26> run
add wave -position insertpoint \
sim:/generatesb/generate2/fp_delay
VSIM 27>


ModelSim ALTERA STARTER EDITION 10.3c
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Your account is undergoing maintenance. Please contact support with a
Wave - Default
Objects
Processes (Active)
Transcript
VSIM 26> run
add wave -position insertpoint \
sim:/generatesb/generate2/fp_delay
VSIM 27>


ModelSim ALTERA STARTER EDITION 10.3c
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Your account is undergoing maintenance. Please contact support with a
Wave - Default
Objects
Processes (Active)
Transcript
VSIM 26> run
add wave -position insertpoint \
sim:/generatesb/generate2/fp_delay
VSIM 27>





Conclusiones

Es importante la utilización del archivo “generate” ya que nos permite conectar en cascada una gran cantidad elementos o registros del mismo de una manera más optima en cuando a contenido de código y tiempo de ejecución.

Con la opción “RTL viewer” podemos ver tanto a nivel externo del módulo como a nivel interno. Y Se pudo encontrar el número máximo de elementos posibles para conectar en cascada, que en nuestro caso no puede superar los 5000 elementos o registros.

Se conoció los parámetros y funciones que se utilizan para implementar un archivo “generate”.