Verificación Funcional De Circuitos Integrados

Proyecto 2

Estudiantes

Ivannia Fernández Rodríguez 2020026764

Irán Medina Aguilar 2020146906

Profesor: Ronny García Ramírez

Semestre II 2023

La aleatorización del número de transacciones, de la terminal que envía, de la terminal que recibe, del dato a enviar, y del retardo se realizó en el agente. En el siguiente fragmento se puede ver como se aleatoriza el número de transacciones para que vaya en un rango entre 1 y la profundidad, y en el caso de lo demás se aleatoriza con la instrucción transaccion.randomize.

```
varios_dispositivos_envio_recibido: begin //Caso en el que se envían varios paquetes aleatorios desde cualquier dispositivo hacia cualquier otro
  num_transacciones = $urandom_range(1, 32); //Define un número aleatorio de transacciones
    for(int i = 0; i< num_transacciones; i++)begin
        espera = 0;
        transaccion = new();
        transaccion.randomize();
        transaccion.fun_pckg;
        while (espera < transaccion.tiempo_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end

        transaccion.tiempo_envio  = $time;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        transaccion_copia = new();
        transaccion_copia = transaccion;
        agente_sb_mbx.put(transaccion_copia);
    end

end
```

El parámetro de la profundidad de las fifos de entrada se aleatorizó en el testbench con el uso de my_package por lo que este es aleatorio pero una vez se definen para el resto de la prueba se mantienen fijos.

```
module test_bench();

    reg clk = 0;

    import my_package::*;

    //Definición de la variable virtual
    mesh_if #(.ROWS(my_package::ROWS), .COLUMNS(my_package::COLUMNS), .PAKG_SIZE(my_package::PAKG_SIZE), .FIFO_DEPTH(my_package::FIFO_DEPTH)) _if(.c

    always #5 clk = ~clk;

    //Definición del test
    test #(.ROWS(my_package::ROWS), .COLUMNS(my_package::COLUMNS), .PAKG_SIZE(my_package::PAKG_SIZE), .FIFO_DEPTH(my_package::FIFO_DEPTH)) test_inst

    //Instancia del dispositivo
    mesh_gnrtr #(.ROWS(my_package::ROWS), .COLUMS(my_package::COLUMNS), .pckg_sz(my_package::PAKG_SIZE), .fifo_depth(my_package::FIFO_DEPTH), .bdcst
    (
        .pndng(_if.pndng),
        .data_out(_if.data_out),
        .popin(_if.popin),
        .pop(_if.pop),
        .data_out_i_in(_if.dato_out_i_in),
        .pndng_i_in(_if.pdng_i_in),
        .clk(_if.clk_i),
        .reset(_if.rst_i)
    );
```

```
module generador_parametros;

    class A;

        integer f;

        rand int unsigned ROWS;
        rand int unsigned COLUMNS;
        rand int unsigned PAKG_SIZE;
        rand int unsigned FIFO_DEPTH;

        constraint C0 { ROWS == 4;}
        constraint C1 { COLUMNS == 4;}
        constraint C2 { PAKG_SIZE == 32;}
        constraint C3 { FIFO_DEPTH > 0; FIFO_DEPTH < 17;}


        function void printPackage;
            f = $fopen("my_package.sv", "w");
            $fdisplay(f, "package my_package;");
            $fdisplay(f, "  parameter ROWS = %0d;", ROWS);
            $fdisplay(f, "  parameter COLUMNS = %0d;", COLUMNS );
            $fdisplay(f, "  parameter PAKG_SIZE = %0d;", PAKG_SIZE );
            $fdisplay(f, "  parameter FIFO_DEPTH = %0d;", FIFO_DEPTH );
            $fdisplay(f, "endpackage");
        endfunction

    endclass

A a;

initial begin
    a = new();
    a.randomize();
    a.printPackage();
end

endmodule
```

Implementación de los escenarios de uso común

Los escenarios de uso común se implementaron en el agente de la siguiente forma:

- Envío de un solo paquete aleatorio por parte de cualquier dispositivo a cualquiera de los otros terminales dentro del rango aleatorio de terminales existentes.
-

```
un_paquete: begin //Caso en el que se envía un solo paquete aleatorio desde cualquier dispositivo hacia cualquier otro dispositivo
        espera = 0;
        transaccion = new();
        transaccion.randomize(); //Vuelve aleatorios los valores de la transacción
        transaccion.fun_pckg;
        while (espera < transaccion.tiempo_retardo) begin //Hace el retardo antes del envío
          @(posedge vif.clk_i)
            espera = espera + 1;
        end
        transaccion.tiempo_envio  = $time;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion); //Envía la transacción al mailbox del agente al driver en la posición de la ter
        transaccion_copia = new();
        transaccion_copia = transaccion;
        agente_sb_mbx.put(transaccion_copia); //Envía la transacción al mailbox del agente al scoreboard

end
```

- Envío de una cantidad aleatoria de paquetes aleatorios por parte uno o varios terminales cualesquiera a cualquiera de los otros terminales dentro de un rango aleatorio de terminales existentes.

```
varios_dispositivos_envio_recibido: begin //Caso en el que se envían varios paquetes aleatorios desde cualquier dispositivo hacia cualquier otro
  num_transacciones = $urandom_range(1, 32); //Define un número aleatorio de transacciones
    for(int i = 0; i< num_transacciones; i++)begin
        espera = 0;
        transaccion = new();
        transaccion.randomize();
        transaccion.fun_pckg;
        while (espera < transaccion.tiempo_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end

        transaccion.tiempo_envio  = $time;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        transaccion_copia = new();
        transaccion_copia = transaccion;
        agente_sb_mbx.put(transaccion_copia);
    end

end
```

- Llenado de todas las FIFOS de todos los dispositivos existentes con datos aleatorios.

```
llenado_fifos: begin //Caso en el que se llenan las FIFOS de todos los drivers disponibles

    for(int i = 0; i < 16 ; i++)begin //For para recorrer todas las filas disponibles
        for (int j = 0; j < FIFO_DEPTH; j++)begin//For para llenar una por una las FIFOS
          transaccion = new();
          transaccion.randomize(); //Vuelve aleatorios los valores de la transacción
          transaccion.fun_pckg;
          transaccion.tiempo_envio  = $time;
          agente_drv_mbx[i].put(transaccion); //Envía la transacción al mailbox del agente al driver en la posición de la terminal de envío
          transaccion_copia = new();
          transaccion_copia = transaccion;
          agente_sb_mbx.put(transaccion_copia);
        end
    end
end
```

Implementación de los escenarios de esquina

Los escenarios de esquina se implementaron en el agente de la siguiente forma:

- Reset antes del envío de una cantidad aleatoria de paquetes aleatorios por parte uno o varios terminales cualesquiera a cualquiera de los otros terminales dentro de un rango aleatorio de terminales existentes.
-

```
reset_inicio: begin //Caso de esquina en el que se realiza un reset antes de que se envíe alguna transacción
  vif.rst_i = '1;
  @(posedge vif.clk_i);
  vif.rst_i = '0;
  num_transacciones = $urandom_range(1, 32); //Define un número aleatorio de transacciones
    for(int i = 0; i< num_transacciones; i++)begin
        espera = 0;
        transaccion = new();
        transaccion.randomize();
        transaccion.fun_pckg;
        while (espera < transaccion.tiempo_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end

        transaccion.tiempo_envio  = $time;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        transaccion_copia = new();
        transaccion_copia = transaccion;
        agente_sb_mbx.put(transaccion_copia);
    end

end
```

- Reset a la mitad del envío de una cantidad aleatoria de paquetes aleatorios por parte uno o varios terminales cualesquiera a cualquiera de los otros terminales dentro de un rango aleatorio de terminales existentes.
-

```
reset_mitad: begin //Caso de esquina en el que se realiza un reset a la mitad del envío de las transacciones
  num_transacciones = $urandom_range(1,FIFO_DEPTH);
    for(int i = 0; i< num_transacciones/2; i++)begin
        espera = 0;
        transaccion = new();
        transaccion.randomize();
        transaccion.fun_pckg;
        while (espera < transaccion.tiempo_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end

        transaccion.tiempo_envio  = $time;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        transaccion_copia = new();
        transaccion_copia = transaccion;
        agente_sb_mbx.put(transaccion_copia);
    end

    vif.rst_i = '1;
    @(posedge vif.clk_i);
    vif.rst_i = '0;

    for(int i = 0; i< num_transacciones/2; i++)begin
        espera = 0;
        transaccion = new();
        transaccion.randomize();
        transaccion.fun_pckg;
        while (espera < transaccion.tiempo_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end
```

```
        transaccion.tiempo_envio  = $time;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        transaccion_copia = new();
        transaccion_copia = transaccion;
        agente_sb_mbx.put(transaccion_copia);
    end
```

- Reset después del envío de una cantidad aleatoria de paquetes aleatorios por parte uno o varios terminales cualesquiera a cualquiera de los otros terminales dentro de un rango aleatorio de terminales existentes.

```
reset_final: begin  //Caso de esquina en el que se realiza un reset anteal final de que se envían las transacciones

  num_transacciones = $urandom_range(1,FIFO_DEPTH);
    for(int i = 0; i< num_transacciones/2; i++)begin
        espera = 0;
        transaccion = new();
        transaccion.randomize();
        transaccion.fun_pckg;
        while (espera < transaccion.tiempo_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end

        transaccion.tiempo_envio  = $time;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        transaccion_copia = new();
        transaccion_copia = transaccion;
        agente_sb_mbx.put(transaccion_copia);
    end
  vif.rst_i = '1;
  @(posedge vif.clk_i);
  vif.rst_i = '0;
end
```

- Envío de una cantidad aleatoria de paquetes aleatorios por parte de uno o varios terminales cualesquiera a cualquier dirección fuera de un rango aleatorio de terminales existentes.

```
envio_fuera_de_rango: begin //Caso de esquina en el que se envían transacciones a una dirección fuera del rango de terminales existentes
  num_transacciones = $urandom_range(1,FIFO_DEPTH);
    for(int i = 0; i< num_transacciones; i++)begin
        espera = 0;
        transaccion = new;
        transaccion.randomize();
        transaccion.row = $urandom_range(ROWS+1,ROWS +20) ;
        transaccion.colum = $urandom_range(COLUMNS + 1, COLUMNS +20);
        transaccion.fun_pckg;
        while (espera < transaccion.tiempo_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end
        transaccion.tiempo_envio  = $time;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        transaccion_copia = new();
        transaccion_copia = transaccion;
        agente_sb_mbx.put(transaccion_copia);
    end

end
```

Aserciones

Las aserciones se implementaron en el monitor donde las que se usaron fueron:

- Comprobar que el dato vaya hacia la fila correcta
- Comprobar que el dato vaya hacia la columna correcta
- Comprobar que si el pending está en cero entonces no haya ningún pop

A continuación se muestra la implementación de esto

```
    assert(transaccion_checker.pckg[PAKG_SIZE-9 : PAKG_SIZE-12] == filas[id_terminal] )//Aserción para comprobar que el paquete se esté enviando a
else $warning("El paquete se envió a una fila erronea");

    assert(transaccion_checker.pckg[PAKG_SIZE-13 : PAKG_SIZE-16] == columnas[id_terminal] )//Aserción para comprobar que el paquete se esté enviar
else $warning("El paquete se envió a una columna erronea");

    if(vif.pndng[id_terminal] == 0) begin
    assert(vif.pop[id_terminal] == 0 )//Aserción para comprobar que no haya un pop si el pending es cero
    else $warning("Hubo un pop cuando el pending estaba en cero");
    end
```

Cobertura

Para la cobertura se creó una clase llamada de esta forma donde para la cobertura funcional se comprobaron las filas, las columnas, y los pop. A continuación se muestra la implementación:

```
class coverage #(parameter PAKG_SIZE= 32);

    covergroup filas;

        coverpoint testbench.DUT.data_out[0][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[1][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[2][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[3][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[4][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[5][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[6][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[7][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[8][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[9][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[10][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[11][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[12][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[13][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[14][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}
        coverpoint testbench.DUT.data_out[15][PAKG_SIZE-9:PAKG_SIZE-12] {bins fila = {[0:5]};}

    endgroup

    covergroup columnas;

        coverpoint testbench.DUT.data_out[0][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[1][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[2][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[3][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[4][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[5][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[6][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[7][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[8][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[9][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[10][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[11][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[12][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
```

```
        coverpoint testbench.DUT.data_out[13][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[14][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}
        coverpoint testbench.DUT.data_out[15][PAKG_SIZE-13:PAKG_SIZE-16] {bins columna = {[0:5]};}

    endgroup

    covergroup pops;
        coverpoint testbench.DUT.pop[0] {bins pop = {1};}
        coverpoint testbench.DUT.pop[1] {bins pop = {1};}
        coverpoint testbench.DUT.pop[2] {bins pop = {1};}
        coverpoint testbench.DUT.pop[3] {bins pop = {1};}
        coverpoint testbench.DUT.pop[4] {bins pop = {1};}
        coverpoint testbench.DUT.pop[6] {bins pop = {1};}
        coverpoint testbench.DUT.pop[7] {bins pop = {1};}
        coverpoint testbench.DUT.pop[8] {bins pop = {1};}
        coverpoint testbench.DUT.pop[9] {bins pop = {1};}
        coverpoint testbench.DUT.pop[10] {bins pop = {1};}
        coverpoint testbench.DUT.pop[11] {bins pop = {1};}
        coverpoint testbench.DUT.pop[12] {bins pop = {1};}
        coverpoint testbench.DUT.pop[13] {bins pop = {1};}
        coverpoint testbench.DUT.pop[14] {bins pop = {1};}
        coverpoint testbench.DUT.pop[15] {bins pop = {1};}

    endgroup

    function new();
        filas = new();
        columnas = new();
        pops = new();

    endfunction
```

```
    task run();
        forever begin
            #25
            filas.sample();
            columnas.sample();
            pops.sample();
        end
    endtask

    function print_cobertura();

        $display("Cobertura de filas: %0.2f", filas.get_coverage(),$time);
        $display("Cobertura de columnas: %0.2f", columnas.get_coverage(),$time);
        $display("Cobertura de pops: %0.2f", pops.get_coverage(),$time);
    endfunction

endclass
```

Y los resultados fueron los siguientes:

```
Se cumplio el tiempo maximo de la prueba
Se imprimió el reporte
Cobertura de filas: 100.00              600605
Cobertura de columnas: 100.00              600605
Cobertura de pops: 26.67              600605
$finish called from file "test.sv", line 211.
$finish at simulation time           6006050000
```

Generación de datos

El retraso promedio en la entrega de paquetes por terminal y general en función de la cantidad de dispositivos y la profundidad de las fifos, el ancho de banda promedio máximo y mínimo en función de la cantidad de dispositivos y la profundidad de las fifos, y la generación del reporte de los paquetes enviados y recibidos en formato csv con el tiempo de envío, terminal de procedencia, terminal de destino, tiempo de recibido, y el retraso en el envío, se generaron en el scoreboard donde el código con lo anterior se encuentra a continuación.

```
while (test_sb_mailbox.num() > 0) begin

    //$display("Se recibio una transaccion de reporte desde el test");

    test_sb_mailbox.get(transaccion_test);

    if (transaccion_test == reporte) begin

        //$display("Se imprimirá el reporte");
        tiempo = 0;
        linea = "";
        linea_agregar = "";
        informacion = {};
        for (int i=0; i < verificadas.size(); ++i) begin


            transaccion_auxiliar = new();
            transaccion_auxiliar = verificadas[i];
            tiempo = tiempo + transaccion_auxiliar.latencia;

            $sformat(linea_agregar, "%h,%g,%g,%g,%g,%g,%g\n",

                transaccion_auxiliar.pckg,
                transaccion_auxiliar.tiempo_envio,
                transaccion_auxiliar.tiempo_recibido,
                transaccion_auxiliar.terminal_envio,
                transaccion_auxiliar.terminal_recibido,
                transaccion_auxiliar.latencia,
                FIFO_DEPTH

            );

            informacion.push_back(linea_agregar);

        end

        archivo_1 = $fopen("Reporte_transacciones.csv", "a" );
```

```
            end
            $fclose(archivo_1);
            tpromedio = tiempo / verificadas.size();
            bw =  PAKG_SIZE * 10e9 / (tpromedio);
            archivo_2 = $fopen("Reporte_Anchos_de_banda_Tiempo_promedio.csv", "a" );
            $sformat (linea, "\n%g,%g,%g,%g,%g,%g", ROWS, COLUMNS, FIFO_DEPTH, PAKG_SIZE, tpromedio, bw);
            $fwrite(archivo_2, "%s", linea);
            $fclose(archivo_2);
            $display("Se imprimió el reporte");
        end else begin
            $display("Solicutud del test al scoreboard desconocido");
        end
    end
```

Resultados de las pruebas

- Reporte de transacciones en .csv

1	Package	t_envio	t_recibido	emisor	receptor	latencia	p_fifo
2	0004128d	105	525	9	3	420	16
3	0004128d	105	525	9	3	420	16
4	0002e120	100275	100355	2	1	80	16
5	0002ce61	100305	100615	8	1	310	16
6	00011274	100395	100695	6	0	300	16
7	0002ad23	100485	100915	8	1	430	16
8	0004128d	105	525	9	3	420	16
9	0002e120	100275	100355	2	1	80	16
10	0002ce61	100305	100615	8	1	310	16
11	00011274	100395	100695	6	0	300	16
12	0002ad23	100485	100915	8	1	430	16
13	00047501	200435	200475	3	3	40	16
14	00030062	200375	200595	14	2	220	16
15	000445cd	200455	200625	15	3	170	16
16	000344d6	200575	200805	13	2	230	16
17	0003c954	200495	200865	14	2	370	16
18	0004fb27	200755	200995	11	3	240	16
19	000116fc	200925	201015	1	0	90	16

111	00037ff3	165	385	14	2	220	23
112	00037ff3	165	385	14	2	220	23
113	0004843b	100375	100415	3	3	40	23
114	0004d309	100425	100525	3	3	100	23
115	00032ad0	100215	100535	5	2	320	23
116	000105e9	100375	100575	5	0	200	23
117	00025478	100235	100575	5	1	340	23
118	00035ecd	100295	100575	4	2	280	23
119	0003edcc	100555	100615	2	2	60	23
120	000118b7	100285	100695	11	0	410	23
121	0002a461	100455	100715	14	1	260	23
122	0001c51e	100665	100735	4	0	70	23
123	00048c35	100585	100745	13	3	160	23
124	00047062	100625	100785	1	3	160	23
125	0003df60	100635	100885	14	2	250	23
126	0002a983	100535	100895	10	1	360	23
127	00031b7d	100455	100925	7	2	470	23
128	00020348	100585	100935	8	1	350	23
129	0003b460	100755	101125	11	2	370	23

275	0004914b	115	575	6	3	460	3
276	0004914b	115	575	6	3	460	3
277	00031db6	100235	100335	12	2	100	3
278	0004e77e	100335	100475	14	3	140	3
279	0002564a	100235	100505	5	1	270	3
280	00048062	100355	100515	1	3	160	3
281	000405b2	100325	100615	2	3	290	3
282	00039208	100305	100665	9	2	360	3
283	0001ece5	100455	100715	8	0	260	3
284	0003054b	100615	100775	0	2	160	3
285	00018544	100395	100805	11	0	410	3
286	0002c608	100535	100845	9	1	310	3
287	000395d8	100555	100975	6	2	420	3
288	0002411a	100685	101055	10	1	370	3
289	00034271	100835	101145	10	2	310	3
290	0002332c	100785	101195	7	1	410	3
291	0001065f	100985	101215	12	0	230	3
292	0004b17c	100735	101235	6	3	500	3
293	00011e08	101055	101315	12	0	260	3

586	0001ac09	100275	100405	2	0	130	6
587	000418ea	100325	100555	0	3	230	6
588	0003ebe5	100385	100645	4	2	260	6
589	0001724e	100535	100795	3	0	260	6
590	00015dd8	100445	100835	11	0	390	6
591	0004f2e2	100625	100875	15	3	250	6
592	0001fad1	100645	100995	13	0	350	6
593	00010e47	100715	101235	11	0	520	6
594	00015b7e	125	245	2	0	120	6
595	0001ac09	100275	100405	2	0	130	6
596	000418ea	100325	100555	0	3	230	6
597	0003ebe5	100385	100645	4	2	260	6
598	0001724e	100535	100795	3	0	260	6
599	00015dd8	100445	100835	11	0	390	6
600	0004f2e2	100625	100875	15	3	250	6
601	0001fad1	100645	100995	13	0	350	6
602	00010e47	100715	101235	11	0	520	6
603	00032d44	200445	200565	1	2	120	6
604	0002403c	200435	200595	4	1	160	6

- Tiempo promedio y ancho de banda en .csv según la profundidad, el ancho, y los dispositivos

1	Rows	Colums	Profundidad	Pack_Size	Tiempo_promedio	Ancho de banda	Dispositivos
2	4	4	16	32	420	7.62E+08	16
3	4	4	16	32	308	1.04E+09	16
4	4	4	16	32	282	1.13E+09	16
5	4	4	16	32	291	1.10E+09	16
6	4	4	16	32	286	1.12E+09	16
7	4	4	23	32	220	1.45E+09	16
8	4	4	23	32	245	1.31E+09	16
9	4	4	23	32	232	1.38E+09	16
10	4	4	23	32	348	9.20E+08	16
11	4	4	23	32	341	9.38E+08	16
12	4	4	3	32	460	6.96E+08	16
13	4	4	3	32	367	8.72E+08	16
14	4	4	3	32	329	9.73E+08	16
15	4	4	3	32	321	9.97E+08	16
16	4	4	3	32	305	1.05E+09	16
17	4	4	6	32	120	-1.63E+09	16
18	4	4	6	32	278	1.15E+09	16
19	4	4	6	32	252	1.27E+09	16

20	4	4	6	32	273	1.17E+09	16
21	4	4	6	32	296	1.08E+09	16
22	4	4	4	32	340	9.41E+08	16
23	4	4	4	32	251	1.27E+09	16
24	4	4	4	32	235	1.36E+09	16
25	4	4	4	32	227	1.41E+09	16
26	4	4	4	32	244	1.31E+09	16
27	4	4	8	32	160	2.00E+09	16
28	4	4	8	32	246	1.30E+09	16
29	4	4	8	32	246	1.30E+09	16
30	4	4	8	32	237	1.35E+09	16
31	4	4	4	32	200	1.60E+09	16
32	4	4	4	32	326	9.82E+08	16
33	4	4	4	32	307	1.04E+09	16
34	4	4	4	32	326	9.82E+08	16
35	4	4	4	32	1326	2.41E+08	16
36	4	4	5	32	260	1.23E+09	16
37	4	4	5	32	310	1.03E+09	16
38	4	4	5	32	299	1.07E+09	16

Gráficas de retraso promedio y ancho de banda promedio en función de la profundidad de las fifos