Verificación Funcional De Circuitos Integrados

Proyecto 1

Estudiantes

Ivannia Fernández Rodríguez 2020026764

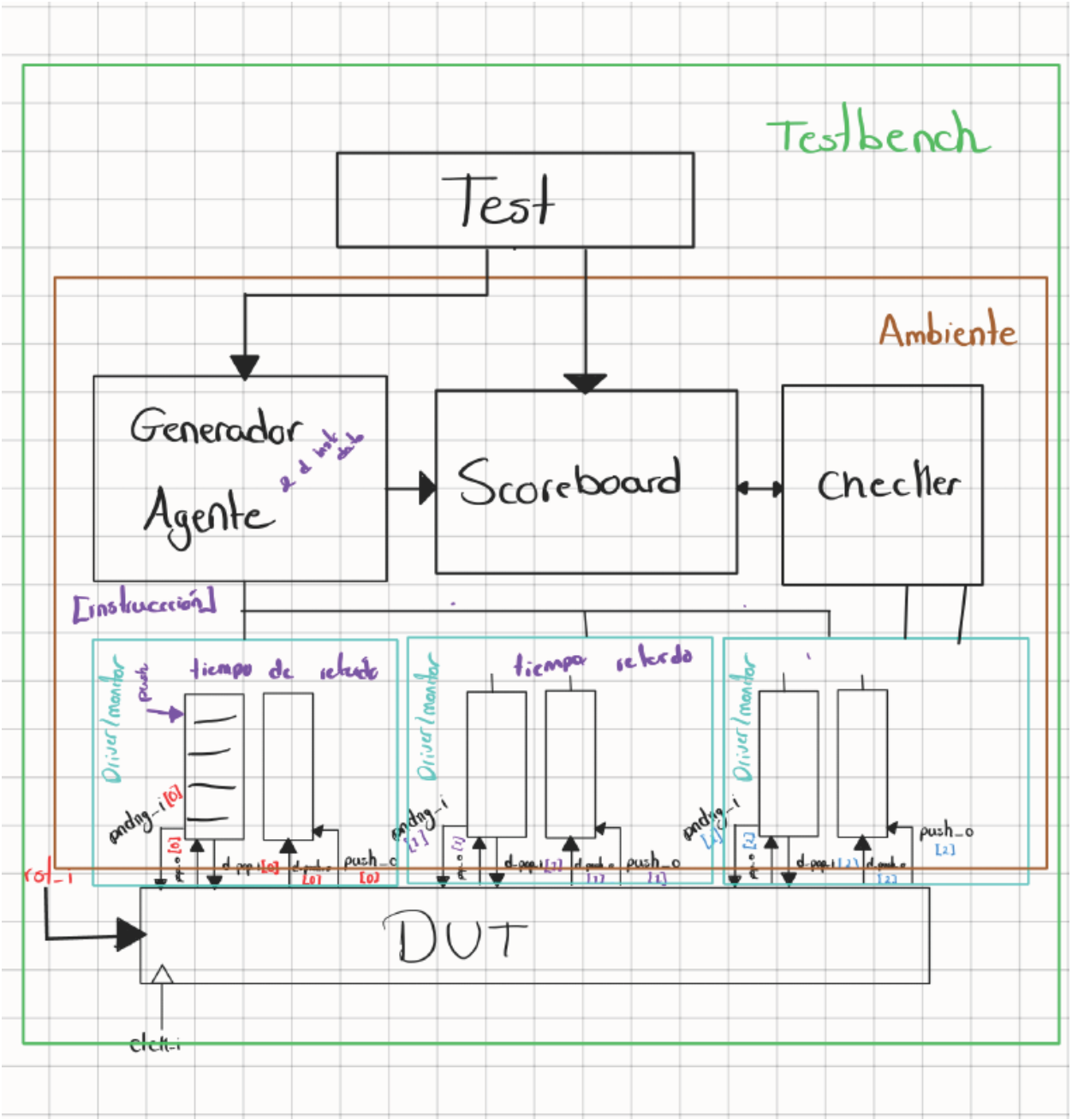
Irán Medina Aguilar 2020146906

Profesor: Ronny García Ramírez

Semestre II 2023

Diagrama

El siguiente diagrama presenta la estructura seguida para la prueba al DUT.



Aleatorización

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, al igual que se aleatoriza la dirección y el identificador donde ambos van en un rango de entre 0 y drivers-1, y de igual forma se aleatoriza el dato a enviar. En el caso del tiempo de retardo se aleatoriza con la instrucción transacción.randomize.

```
varios_dispositivos_envio_recibido: begin
  num_transacciones = $urandom_range(1,profundidad);
    for(int i = 0; i< num_transacciones; i++)begin
        espera = 0;
        transaccion = new;
        transaccion.randomize();
        direccion = $urandom_range(0,drivers-1);
        identificador = $urandom_range(0,drivers-1);
        transaccion.terminal_envio = identificador;
        dato = $random;
        dato_enviado = {direccion,dato,identificador};
        transaccion.dato_enviado = dato_enviado;
        while (espera < transaccion.t_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end
        transaccion.t_envio  = $time;
        tpo_spec = enviar;
        transaccion.tipo = tpo_spec;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        agente_sb_mbx.put(transaccion);
    end

end
```

Los parámetros como lo son el número de terminales, la profundidad de las fifos de entrada, y el identificador de broadcast se aleatorizaron en el testbench por lo que estos son aleatorios pero una vez se definen para el resto de la prueba se mantienen fijos.

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
        direccion = $urandom_range(0,drivers-1);
        identificador = transaccion.terminal_envio;
        transaccion.terminal_envio = identificador;
        dato = $random;
        dato_enviado = {direccion,dato,identificador};//Concatena el valor de la dirección, el dato y el identificador
        transaccion.dato_enviado = dato_enviado;
        while (espera < transaccion.t_retardo) begin //Hace el retardo antes del envío
          @(posedge vif.clk_i)
            espera = espera + 1;
        end
        transaccion.t_envio  = $time;
        tpo_spec = enviar; //Define el tipo específico
        transaccion.tipo = tpo_spec;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion); //Envía la transacción al mailbox del agente al driver en la posición de la
        agente_sb_mbx.put(transaccion); //Envía la transacción al mailbox del agente al scoreboard

end
```

- Envío de una cantidad aleatoria de paquetes con direcciones aleatorias y diferentes tiempos de retraso por parte de un solo dispositivo aleatorio a cualquier otro dispositivo dentro de un rango aleatorio de terminales existentes.

```
un_dispositivo_envio: begin //Caso en el que se envían varios paquetes aleatorios desde un único dispositivo hacia cualquier otro dispositivo
        identificador = $urandom_range(0,drivers-1); //Define cual dispositivo es al que va a enviar siempre
        num_transacciones = $urandom_range(1,profundidad); //Define un número aleatorio de transacciones
    for(int i = 0; i< num_transacciones; i++)begin
        espera = 0;
        transaccion = new;
        transaccion.randomize();
        direccion = $urandom_range(0,drivers-1);
        transaccion.terminal_envio = identificador; //Asigna a la terminal de envío el identificador definido anteriormente
        dato = $random;
        dato_enviado = {direccion,dato,identificador};
        transaccion.dato_enviado = dato_enviado;
        while (espera < transaccion.t_retardo) begin
          @(posedge vif.clk_i);
           espera = espera +1;
        end
        transaccion.t_envio  = $time;
        tpo_spec = enviar;
        transaccion.tipo = tpo_spec;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        agente_sb_mbx.put(transaccion);
    end

  end
```

- Envío de una cantidad aleatoria de paquetes con diferentes tiempos de retraso por parte de terminales aleatorias a un solo dispositivo dentro de un rango aleatorio de terminales existentes.

```
un_dispositivo_recibido: begin //Caso en el que se envían varios paquetes aleatorios desde cualquier dispositivo hacia un único dispositivo
            direccion = $urandom_range(0,drivers-1); //Define a cuál dispositivo es al que siempre se le van a enviar los datos
            num_transacciones = $urandom_range(1,profundidad); //Define un número aleatorio de transacciones
        for(int i = 0; i< num_transacciones; i++)begin
            espera = 0;
            transaccion = new;
            transaccion.terminal_recibido = direccion; //Asigna a la terminal de recibido la dirección obtenida anteriormente
            transaccion.randomize();
            identificador = $urandom_range(0,drivers-1);
            transaccion.terminal_envio = identificador;
            dato = $random;
            dato_enviado = {direccion,dato,identificador};
            transaccion.dato_enviado = dato_enviado;
            while (espera < transaccion.t_retardo) begin
                @(posedge vif.clk_i);
                espera = espera +1;
            end
            transaccion.t_envio  = $time;
            tpo_spec = enviar;
            transaccion.tipo = tpo_spec;
            transaccion.print();
            agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
            agente_sb_mbx.put(transaccion);
        end

    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 dis
    num_transacciones = $urandom_range(1,profundidad); //Define un número aleatorio de transacciones
        for(int i = 0; i< num_transacciones; i++)begin
            espera = 0;
            transaccion = new;
            transaccion.randomize();
            direccion = $urandom_range(0,drivers-1);
            identificador = $urandom_range(0,drivers-1);
            transaccion.terminal_envio = identificador;
            dato = $random;
            dato_enviado = {direccion,dato,identificador};
            transaccion.dato_enviado = dato_enviado;
            while (espera < transaccion.t_retardo) begin
                @(posedge vif.clk_i)
                espera = espera +1;
            end
            transaccion.t_envio  = $time;
            tpo_spec = enviar;
            transaccion.tipo = tpo_spec;
            transaccion.print();
            agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
            agente_sb_mbx.put(transaccion);
        end

    end
```

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

```
llenado_fifos: begin //Caso en el que se llenan las FIFOS de todos los drivers disponibles
    direccion = 8'b00000000; // Define la dirección inicial en 0
    for(int i = 0; i < drivers; i++)begin //For para recorrer todos los drivers disponibles
        direccion = direccion + 1; //Suma 1 a la dirección cada vez que se llenan las FIFOS del driver
        for (int j = 0; j < profundidad-1; j++)begin//For para llenar una por una las FIFOS
          transaccion = new;
          transaccion.randomize();
          identificador = $urandom_range(0,drivers-1);
          transaccion.terminal_envio = identificador;
          transaccion.terminal_recibido = direccion; //Asigna a la terminal de recibido el valor de la dirección con la que se está trabajando
          dato = $random;
          dato_enviado = {direccion,dato,identificador};
          transaccion.dato_enviado = dato_enviado;
          transaccion.t_envio  = $time;
          tpo_spec = enviar;
          transaccion.tipo = tpo_spec;
          transaccion.print();
          agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
          agente_sb_mbx.put(transaccion);

        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
  transaccion = new; //Crea la transacción
  tpo_spec = reset; //Define el tipo como reset
  transaccion.tipo = tpo_spec;
  identificador = $urandom_range(0,drivers-1); //Define un identificador aleatorio
  transaccion.terminal_envio = identificador; //Asigna a la terminal de invío el identificar obtenido
  transaccion.print();
  agente_sb_mbx.put(transaccion); //Envía la transacción al scoreboard
  agente_drv_mbx[transaccion.terminal_envio].put(transaccion); //Envía la transacción al driver correspondiente

  num_transacciones = $urandom_range(1,profundidad);
    for(int i = 0; i< num_transacciones; i++)begin
        espera = 0;
        transaccion = new;
        transaccion.randomize();
        direccion = $urandom_range(0,drivers-1);
        identificador = $urandom_range(0,drivers-1);
        transaccion.terminal_envio = identificador;
        dato = $random;
        dato_enviado = {direccion,dato,identificador};
        transaccion.dato_enviado = dato_enviado;
        while (espera < transaccion.t_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end
        transaccion.t_envio  = $time;
        tpo_spec = enviar;
        transaccion.tipo = tpo_spec;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        agente_sb_mbx.put(transaccion);
    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,profundidad);
    for(int i = 0; i< num_transacciones/2; i++)begin //Se divide a la mitad las transacciones
        espera = 0;
        transaccion = new;
        transaccion.randomize();
        direccion = $urandom_range(0,drivers-1);
        identificador = $urandom_range(0,drivers-1);
        transaccion.terminal_envio = identificador;
        dato = $random;
        dato_enviado = {direccion,dato,identificador};
        transaccion.dato_enviado = dato_enviado;
        while (espera < transaccion.t_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end
        transaccion.t_envio  = $time;
        tpo_spec = enviar;
        transaccion.tipo = tpo_spec;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        agente_sb_mbx.put(transaccion);
    end

    transaccion = new; //Crea la transacción
    tpo_spec = reset; //Define el tipo como reset
    transaccion.tipo = tpo_spec;
    identificador = $urandom_range(0,drivers-1); //Define un identificador aleatorio
    transaccion.terminal_envio = identificador; //Asigna a la terminal de invío el identificar obtenido
    transaccion.print();
    agente_sb_mbx.put(transaccion); //Envía la transacción al scoreboard
    agente_drv_mbx[transaccion.terminal_envio].put(transaccion); //Envía la transacción al driver correspondiente


    for(int i = 0; i< num_transacciones/2; i++)begin //Se hace la otra mitad de las transacciones
        espera = 0;
        transaccion = new;
        transaccion.randomize();
        direccion = $urandom_range(0,drivers-1);
        identificador = $urandom_range(0,drivers-1);
        transaccion.terminal_envio = identificador;
        dato = $random;
        dato_enviado = {direccion,dato,identificador};
        transaccion.dato_enviado = dato_enviado;
        while (espera < transaccion.t_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end
        transaccion.t_envio  = $time;
        tpo_spec = enviar;
        transaccion.tipo = tpo_spec;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        agente_sb_mbx.put(transaccion);
    end
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,profundidad);
    for(int i = 0; i< num_transacciones; i++)begin
        espera = 0;
        transaccion = new;
        transaccion.randomize();
        direccion = $urandom_range(0,drivers-1);
        identificador = $urandom_range(0,drivers-1);
        transaccion.terminal_envio = identificador;
        dato = $random;
        dato_enviado = {direccion,dato,identificador};
        transaccion.dato_enviado = dato_enviado;
        while (espera < transaccion.t_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end
        transaccion.t_envio  = $time;
        tpo_spec = enviar;
        transaccion.tipo = tpo_spec;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        agente_sb_mbx.put(transaccion);

    end
  transaccion = new; //Crea la transacción
  tpo_spec = reset; //Define el tipo como reset
  transaccion.tipo = tpo_spec;
  identificador = $urandom_range(0,drivers-1); //Define un identificador aleatorio
  transaccion.terminal_envio = identificador; //Asigna a la terminal de invío el identificar obtenido
  transaccion.print();
  agente_sb_mbx.put(transaccion); //Envía la transacción al scoreboard
  agente_drv_mbx[transaccion.terminal_envio].put(transaccion); //Envía la transacción al driver correspondiente
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,profundidad);
    for(int i = 0; i< num_transacciones; i++)begin
        espera = 0;
        transaccion = new;
        transaccion.randomize();
        direccion = $urandom_range(drivers,drivers +20); //Asigna a dirección un valor aleatorio mayor a la cantidad de drivers existentes
        identificador = $urandom_range(0,drivers-1);
        transaccion.terminal_envio = identificador;
        dato = $random;
        dato_enviado = {direccion,dato,identificador};
        transaccion.dato_enviado = dato_enviado;
        while (espera < transaccion.t_retardo) begin
          @(posedge vif.clk_i)
            espera = espera +1;
        end
        transaccion.t_envio  = $time;
        tpo_spec = enviar;
        transaccion.tipo = tpo_spec;
        transaccion.print();
        agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
        agente_sb_mbx.put(transaccion);
    end

end
```

- Envío de una cantidad aleatoria de paquetes aleatorios por parte de una sola terminal cualquiera a sí misma.

```
autoenvio: begin //Caso de esquina en el que una terminal se envía a sí misma datos
    identificador = $urandom_range(0,drivers-1); //Define el identificador de la terminal que se va a autoenviar datos
    num_transacciones = $urandom_range(1,profundidad); //Define un número aleatorio de transacciones
    for(int i = 0; i< num_transacciones; i++)begin
      espera = 0;
      transaccion = new;
      transaccion.randomize();
      direccion = identificador; //Define que la dirección a la que va a enviar va a ser a sí misma
      transaccion.terminal_envio = identificador;
      dato = $random;
      dato_enviado = {direccion,dato,identificador};
      transaccion.dato_enviado = dato_enviado;
      while (espera < transaccion.t_retardo) begin
        @(posedge vif.clk_i)
          espera = espera +1;
      end
      transaccion.t_envio  = $time;
      tpo_spec = enviar;
      transaccion.tipo = tpo_spec;
      transaccion.print();
      agente_drv_mbx[transaccion.terminal_envio].put(transaccion);
      agente_sb_mbx.put(transaccion);
  end
end
```

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.

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

    test_sb_mailbox.get(transaccion_test);

    case (transaccion_test)

        reporte: begin
            $display("Mae si era una transacciones de reporte");
            bw = 0;
            tiempo = 0;
            linea = "";
            linea_agregar = "";
            informacion = {};


            for (int i=0; i < verificadas.size(); ++i) begin

                $display("[%i]", i );
                transaccion_auxiliar = new();
                transaccion_auxiliar = verificadas[i];
                tiempo = tiempo + transaccion_auxiliar.latencia;
                $display("\n%h,%g,%g,%g,%g,%g,%g,%g",

                transaccion_auxiliar.pckg,
                transaccion_auxiliar.t_envio,
                transaccion_auxiliar.t_recibido,
                transaccion_auxiliar.terminal_envio,
                transaccion_auxiliar.terminal_recibido,
                transaccion_auxiliar.latencia,
                transaccion_auxiliar.prof,
                transaccion_auxiliar.drv);

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

```
    transaccion_auxiliar.pckg,
    transaccion_auxiliar.t_envio,
    transaccion_auxiliar.t_recibido,
    transaccion_auxiliar.terminal_envio,
    transaccion_auxiliar.terminal_recibido,
    transaccion_auxiliar.latencia,
    transaccion_auxiliar.prof,
    transaccion_auxiliar.drv

    );

    informacion.push_back(linea_agregar);



end

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

for (int i=0; i<informacion.size(); ++i) begin

    $fwrite(archivo_1, "%s", informacion[i]);

end

$fclose(archivo_1);

tpromedio = tiempo / verificadas.size();

bw = (width) / (8 * tpromedio );
```

```
        $display("Para una prueba con Terminales: [%g], Profundidad: [%g], Ancho de palabra: [%g], Tiempo promedio: [%g], Ancho de banda: [%g] \n
        archivo_2 = $fopen("Reporte_Anchos_de_banda_Tiempo_promedio.csv", "a" );
        $sformat (linea, "\n%g,%g,%g,%g,%g", drivers, profundidad, width, tpromedio, bw);
        $fwrite(archivo_2, "%s", linea);
        $fclose(archivo_2);
        $finish;

    end
    default: begin
        $display("No se recibio ninguna instrucción de reporte valida desde el test");
    end
endcase
```

Resultados de las pruebas

Reporte de transacciones en .csv

```
pckg, tiempo_envio, tiempo_recibido, terminal_envio, terminal_recibido, latencia
, profunidad, drivers
0c53524e,25,635,14,12,610,1,16
0895e81c,10055,10525,12,8,470,1,16
017cd0dc,10515,11175,12,1,660,1,16
07dcd3dc,10575,11825,12,7,1250,1,16
0495e810,10075,10605,0,4,530,16,16
033e3010,10505,11255,0,3,750,16,16
0453524a,5,555,10,4,550,16,16
0a95e818,10065,10455,8,10,390,16,16
0a284658,10325,11105,8,10,780,16,16
0395e81a,10095,10485,10,3,390,16,16
0175212a,10475,11135,10,1,660,16,16
0dde9f9a,10915,11785,10,13,870,16,16
0e535245,15,455,5,14,440,16,16
0f95e81a,10055,10485,10,15,430,16,16
087cd0da,10465,11135,10,8,670,16,16
0cde9f9a,10755,11785,10,12,1030,16,16
~
```

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

```
File  Edit  View  Search  Terminal  Help
Drivers, Profundidad, Ancho, Tiempo_promedio, Ancho_de_banda
16,16,32,747,0
16,16,32,640,0
16,16,32,573,0
16,16,32,640,0
16,16,32,642,0
~
```