

UNIVERSIDAD DE COSTA RICA

ESCUELA DE INGENIERÍA ELÉCTRICA

IE-0523 - CIRCUITOS DIGITALES 2

---

Tarea #1

**Instalación de Herramientas del Curso**

---

*Autor:*

Arnaldo Chacón Madrigal  
B41768

*Profesor:*

Jorge Soto



Ciudad Universitaria Rodrigo Facio, Costa Rica

30 de agosto de 2018

## I. TIEMPOS REQUERIDOS

Cuadro I: Tiempos invertidos en cada Tarea

Tarea	Búsqueda de Información	Estudio de Información	Ejecución	Confección del reporte
Tiempo invertido	3h	2h 30min	2h	1h 50min

## II. DESCRIPCIÓN DE LAS HERRAMIENTAS

### II-A. *Icarus Verilog:*

Verilog es un lenguaje de descripción de hardware (HDL). Verilog posee distintos niveles de abstracción: es posible describir un circuito como conexiones de compuertas lógicas o a un nivel de comportamiento (behavioral). Los diseños en Verilog pueden ser sintetizados y fabricados, implementados en una FPGA o simulados en un computador. [3].

Icarus Verilog es una herramienta de simulación y síntesis de Verilog. Funciona como un compilador, compilando código fuente escrito en Verilog (IEEE-1364) en algún formato de destino.[1]

El código en Verilog permite describir el comportamiento de un circuito digital y simularlo a través de Icarus.

Para la simulación por lotes, el compilador puede generar una forma intermedia llamada ensamblaje vvp. Esta forma intermedia se ejecuta con el comando “ vvp ”. Para la síntesis, el compilador genera listas de conexiones en el formato deseado. [4].

Dentro del módulo probador, es posible crear un archivo .vcd, el cual contiene los datos del proceso, esto para simularlo en GTKwave.

### II-B. *GTKWave:*

GTKWave es un completo visor de onda basado en GTK+, para Unix, Windows y otras plataformas. Que lee archivos LXT, LXT2, VZT, FST, y los archivos GHW, así como archivos estándar Verilog VCD / EVcd y permite su visualización en pantalla.

GTKWave es el visor recomendado por desarrollador el Icarus Verilog y uno de los mejores visores de onda de software libre.

### II-C. *Yosys:*

Es un framework para la síntesis de Verilog RTL. L. Actualmente cuenta con un amplio soporte de Verilog-2005 y proporciona un conjunto básico de algoritmos de síntesis para diversos dominios de aplicación. Características seleccionadas y aplicaciones típicas: [2]

- Procesa casi cualquier diseño sintetizable Verilog-2005
- Conversión de Verilog a BLIF / EDIF / BTOR / SMT-LIB / RTL simple Verilog / etc
- Métodos formales integrados para comprobar propiedades y equivalencia
- Asignación a bibliotecas de celdas estándar ASIC (en formato Liberty File)
- Mapeo de FPGAs Xilinx de 7 Series y Lattice iCE40 Fundación y / o front-end para flujos personalizados

### III. EMPLEO DE LAS HERRAMIENTAS CON UN MULTIPLEXOR 2-1

Para emplear y probar las herramientas, se crea el modelo conductual y un modelo estructural de un mux 2-1. Para el mismo, se crea un probador para comprobar el funcionamiento correcto de ambos modelos. Además de un banco de pruebas para interconectar cada uno de los módulos.

El código de verilog correspondiente al mux, se presenta a continuación. Como el fin es mostrar el correcto empleo de las herramientas, no se entrará en detalles respecto al código.

```

mux_cond.v
// Multiplexor 2-1. Descripción Conductual.
// Cuando c=1, out=a; y cuando c=0, out=b
module mux_cond (out, c, a, b);
    //salidas del Mux
    output out;
    // Entradas al Mux
    input c, a, b;
    reg out;
    always @(*) //asterisco representa
    //todas las entradas que pueden modificar la salida
    out= (~c&b) | (c&a);
endmodule

```

(a) Modelo Conductual

```

mux_estruc.v
// Mux 2-1. Descripción Estructural.
// Cuando c=1, out=a; y cuando c=0, out=b
module mux_estruc (out, c, a, b);
    // salidas del mux
    output out;
    input c, a, b;
    wire n1, n0;
    // nodos internos
    and(n0, b, ~c);
    and(n1, a, c);
    or(out, n1, n0);
endmodule

```

(b) Modelo Estructural

Figura 1: Modelos del MUX 2-1.

```

mux_prob.v
//Modulo probador.
module mux_prob (out_cond, out_estruc, a, b, c);
    input out_estruc;
    input out_cond;

    output reg a;
    output reg b;
    output reg c;

    reg clk;

    initial begin
        $dumpfile("multiplexor.vcd");// Nombre de archivo del "dump"
        $dumpvars; // Directiva para "dumpear" variables
        // Mensaje que se imprime en consola una vez
        $display ("a\tb\tc\tout_estruc\tout_cond");
        $monitor("%b\t%b\t%b\t%b\t%b", a, b, c, out_estruc, out_cond);
        a=0; b=1; c=0;
        #10 a=1; b=0; c=0;
        #10 c=1;
        #10 a=0; b=1; c=1;
        #10 $finish;
    end
    //Reloj
    initial clk <=0;
    always #2 clk <= ~clk;
endmodule

```

(a) Módulo Probador

```

mux_bancoPrueba.v
//Banco de pruebas multiplexor
timescale 1ns/100ps
// escala unidad temporal (valor de "1") / precisión
// incluye de archivos de verilog
// Pueden omitirse y llamarse desde el testbench
`include "mux_cond.v"
`include "mux_estruc.v"
`include "mux_prob.v"

module mux_bancoPrueba ();
    wire out_estruc, out_cond, a, b, c;
    mux_cond m_cond(out_cond, a, b, c);
    mux_estruc m_estruc(out_estruc, a, b, c);
    mux_prob m_mux_prob(out_estruc, out_cond, a, b, c);
endmodule

```

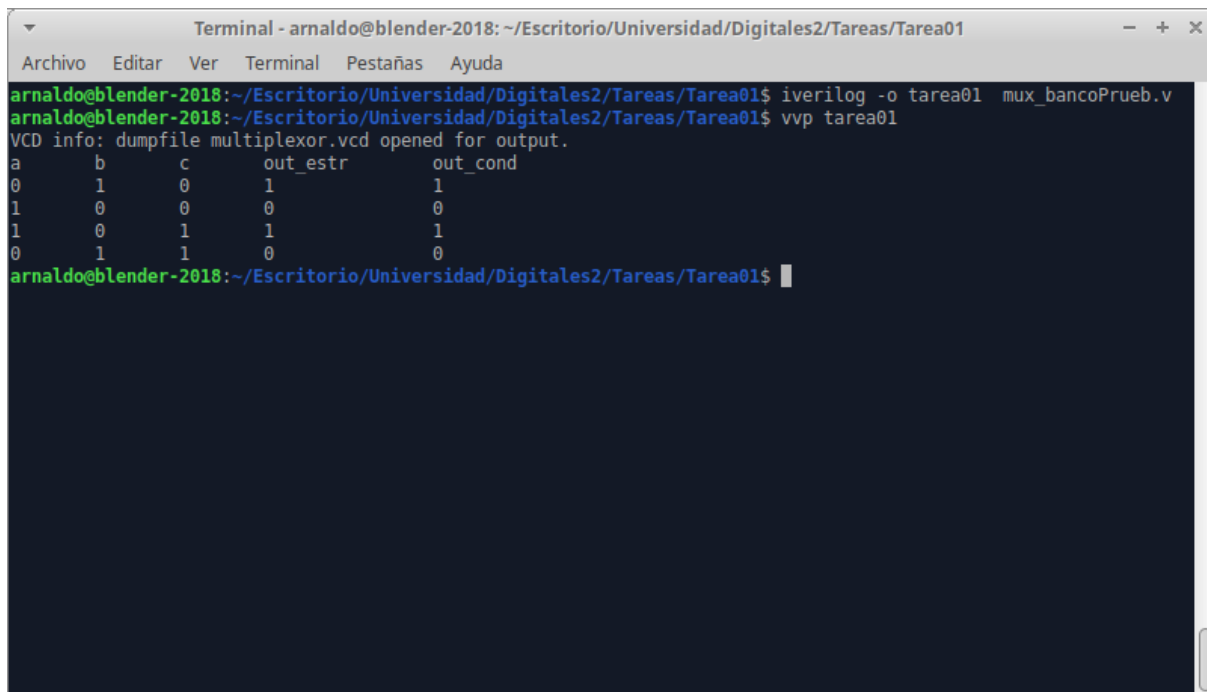
(b) Banco de Pruebas

Figura 2: Probador y Banco de pruebas para los Modelos.

### III-A. Icarus Verilog

Se realiza una prueba de compilación y ejecución del programa anterior. El cual mientras que la señal de selección  $c$  sea igual a 1, la salida será igual a  $a$ . Con  $c$  igual a 0, la salida será  $b$ . Se utiliza un archivo testbench o probador para comprobar los resultados obtenidos. Se imprime en pantalla los resultados.

Se muestra en la siguiente figura el proceso:



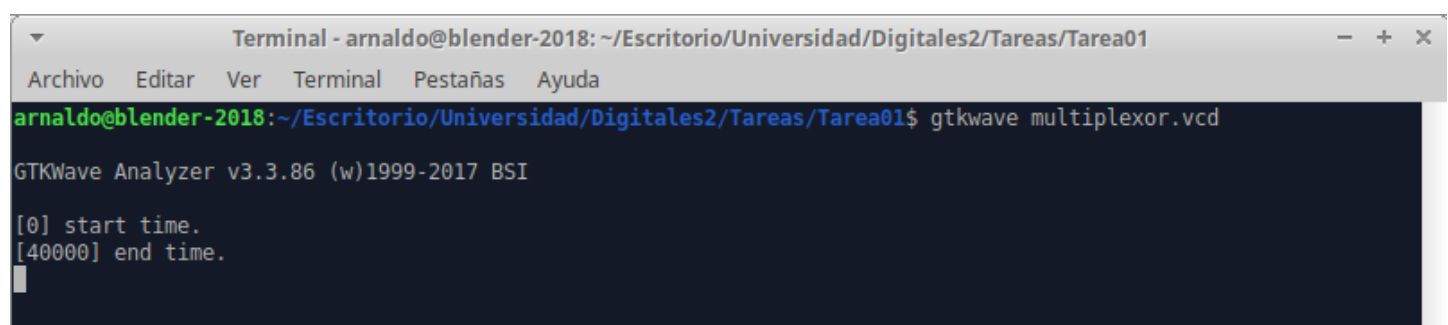
```
Terminal - arnaldo@blender-2018: ~/Escritorio/Universidad/Digitales2/Tareas/Tarea01
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
arnaldo@blender-2018:~/Escritorio/Universidad/Digitales2/Tareas/Tarea01$ iverilog -o tarea01 mux_bancoPrueb.v
arnaldo@blender-2018:~/Escritorio/Universidad/Digitales2/Tareas/Tarea01$ vvp tarea01
VCD info: dumpfile multiplexor.vcd opened for output.
a      b      c      out_estr      out_cond
0      1      0      1              1
1      0      0      0              0
1      0      1      1              1
0      1      1      0              0
arnaldo@blender-2018:~/Escritorio/Universidad/Digitales2/Tareas/Tarea01$
```

Figura 3: Compilación con Icarus Verilog

Se logra obtener el resultado esperado, como se muestra en la imagen anterior.

### III-B. GTKWave

En la siguiente imagen se observa que también se abre un archivo `.vcd` creado en el proceso de compilación con GTKWave. Se muestra la visualización del archivo `.v` de verilog que se puede realizar con GTKWave:



```
Terminal - arnaldo@blender-2018: ~/Escritorio/Universidad/Digitales2/Tareas/Tarea01
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda
arnaldo@blender-2018:~/Escritorio/Universidad/Digitales2/Tareas/Tarea01$ gtwave multiplexor.vcd
GTKWave Analyzer v3.3.86 (w)1999-2017 BSI
[0] start time.
[40000] end time.
arnaldo@blender-2018:~/Escritorio/Universidad/Digitales2/Tareas/Tarea01$
```

Figura 4: Visualización en GTKwave

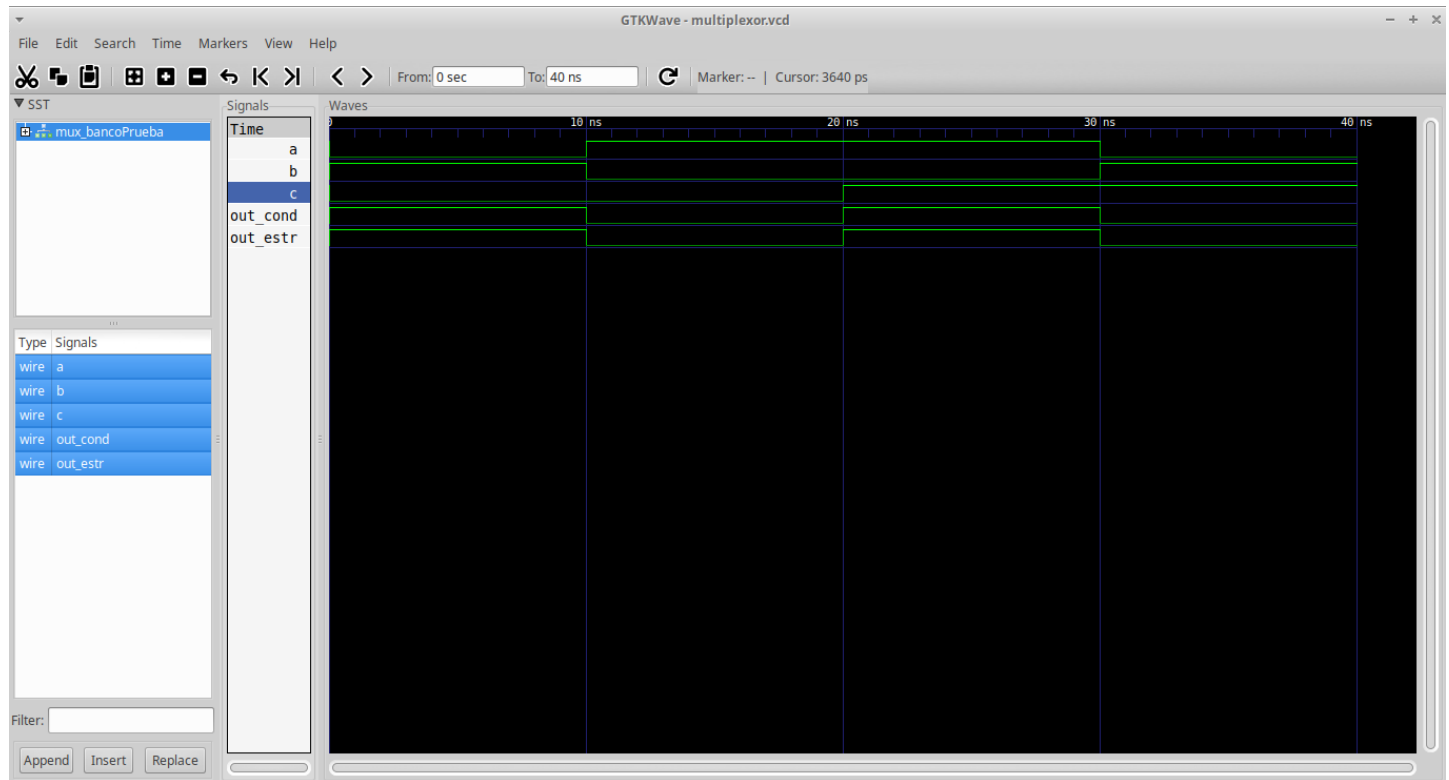


Figura 5: Visualización en GTKwave

En la figura anterior observan con más claridad los resultados, mostrando que cuando la señal c está en alto, la señal de salida corresponde al valor de a.

### III-C. Yosys

Se realiza la lectura de un archivo .v empleando este software. Sea crea un archivo .dot que al transformarlo en una imagen o un pdf se puede observar el diagrama de flujo correspondiente a la lógica del circuito descrito en verilog.

```

Terminal - arnaldo@blender-2018: ~/Escritorio/Universidad/Digitales2/Tareas/Tai - + x
Archivo  Editor  Ver  Terminal  Pestañas  Ayuda

| OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. |
|-----|
Yosys 0.7 (git sha1 61f6811, gcc 6.2.0-11ubuntu1 -02 -fdebug-prefix-map=/build/
yosys-0IL3SR/yosys-0.7=. -fstack-protector-strong -fPIE -Os)

yosys> read_verilog mux_estruc.v
1. Executing Verilog-2005 frontend.
Parsing Verilog input from 'mux_estruc.v' to AST representation.
Generating RTLIL representation for module '\mux_estruc'.
Successfully finished Verilog frontend.

yosys> show

2. Generating Graphviz representation of design.
Writing dot description to '/home/arnaldo/.yosys_show.dot'.
Dumping module mux_estruc to page 1.

```

Figura 6: Yosys Ejecución

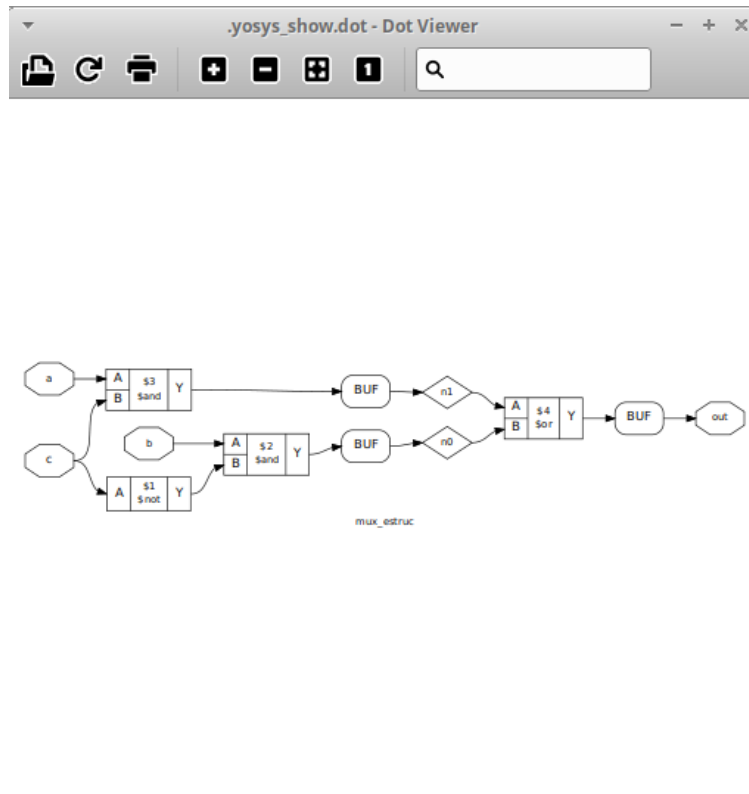


Figura 7: Yosys Resultado

#### IV. MAKEFILE

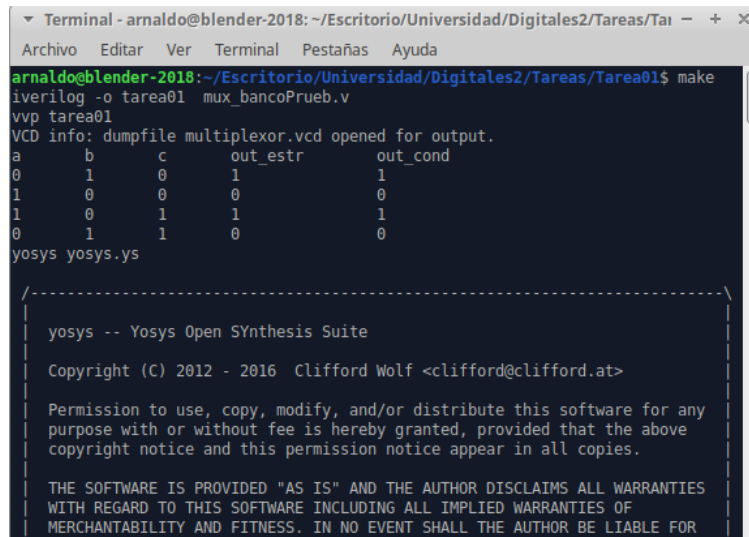
Por último, se procede a crear un archivo makefile con el fin de automatizar el proceso de compilación y ejecución.

A continuación, se muestra el archivo, junto con un archivo .ys para yosys, el cual se emplea para ejecutar los comandos de síntesis y escritura desde dicha herramienta.

Makefile	yosys.ys
<code>all: verilog vvp yosys gtkw</code>	<code>1 read verilog mux estruc.v</code>
<code>verilog:</code>	<code>2 hierarchy; proc; opt; techmap; opt</code>
<code>iverilog -o tarea01 mux_bancoPrueb.v</code>	<code>3 write verilog mux_syn.v</code>
<code>vvp:</code>	<code>4 show</code>
<code>vvp tarea01</code>	<code>5</code>
<code>gtkw:</code>	
<code>gtkwave multiplexor.vcd</code>	
<code>yosys:</code>	
<code>yosys yosys.ys</code>	
<code>clean:</code>	
<code>rm multiplexor.vcd tarea01 mux_syn.v</code>	

Figura 8: Archivo Makefile de automatización.

Y la ejecución del mismo mediante el comando make.



```
Terminal - arnaldo@blender-2018: ~/Escritorio/Universidad/Digitales2/Tareas/Tarea01
Archivo Editar Ver Terminal Pestañas Ayuda
arnaldo@blender-2018:~/Escritorio/Universidad/Digitales2/Tareas/Tarea01$ make
iverilog -o tarea01 mux_bancoPrueb.v
vvp tarea01
VCD info: dumpfile multiplexor.vcd opened for output.
a      b      c      out_estr      out_cond
0      1      0      1              1
1      0      0      0              0
1      0      1      1              1
0      1      1      0              0
yosys yosys.ys

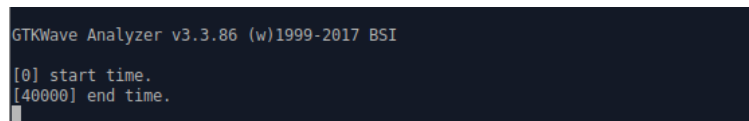
-----
yosys -- Yosys Open SYNthesis Suite

Copyright (C) 2012 - 2016 Clifford Wolf <clifford@clifford.at>

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
```

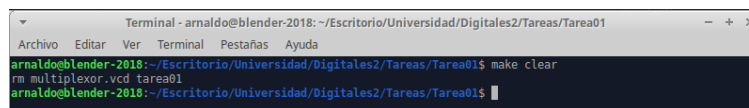
Figura 9: Compilación y ejecución mediante el comando make. Ejecutando también los comandos de yosys contenidos en el archivo.



```
GTKWave Analyzer v3.3.86 (w)1999-2017 BSI

[0] start time.
[40000] end time.
```

Figura 10: Se incluye en el makefile, la ejecución de GTKwave.



```
Terminal - arnaldo@blender-2018: ~/Escritorio/Universidad/Digitales2/Tareas/Tarea01
Archivo Editar Ver Terminal Pestañas Ayuda
arnaldo@blender-2018:~/Escritorio/Universidad/Digitales2/Tareas/Tarea01$ make clear
rm multiplexor.vcd tarea01
arnaldo@blender-2018:~/Escritorio/Universidad/Digitales2/Tareas/Tarea01$
```

Figura 11: Se eliminan los archivos creados en el proceso de ejecución.

## REFERENCIAS

- [1] Williams, S. (n.d.). Icarus Verilog. [online] Iverilog.icarus.com. Available at: <http://iverilog.icarus.com/> [Accessed 27 Aug. 2018].
- [2] Clifford.at. (n.d.). Yosys Open SYnthesis Suite. [online] Available at: <http://www.clifford.at/yosys/> [Accessed 27 Aug. 2018].
- [3] Estevez, C. (2011). Guia de Instalación y Uso de Icarus Verilog y GtkWave. [online]. Available at: [https://www.u-cursos.cl/ingenieria/2011/1/EL4102/1/material\\_docente/bajar?id\\_material=352936](https://www.u-cursos.cl/ingenieria/2011/1/EL4102/1/material_docente/bajar?id_material=352936) [Accessed 28 Aug. 2018].
- [4] Icarus Verilog (Página Oficial). Available at: <http://iverilog.icarus.com/> [Accessed 28 Aug. 2018].
- [5] sourceforge .(2018) Welcome to GTKWave. Available at: <http://gtkwave.sourceforge.net/> [Accessed 28 Aug. 2018].