

Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
Circuitos Digitales II
Prof. Jorge Soto
II Ciclo 2020
IE-0523

Tarea #1

1. Parte 1:Tiempo

1.1. Segunda parte:

Buscar información: Del Icarus Verilog con GTKwave se obtuvo en 2 minutos y del Yosys en 40 segundos.

Estudiar la información para decidir qué hacer: Aproximadamente 20 segundos, ya que solo era una línea de código

Ejecutar lo que decidió hacer: Aproximadamente 10 segundos instalar cada programa ya que solo era copiar el código

1.2. Tercera parte:

Buscar información: Del Icarus Verilog y GTKwave se obtuvo en 4 minutos y del Yosys en 12 minutos, debido a que busqué videos en Youtube.

Estudiar la información para decidir qué hacer: Del Icarus Verilog y GTKwave 12:40 minutos porque investigué mucho de módulos de Verilog y de Yosys 7 minutos.

Ejecutar lo que decidió hacer: Del Icarus Verilog y GTKwave tarde 10 minutos y Yosys duré 4 minutos.

1.3. Cuarta parte:

Buscar información: Acerca el makefile en general duré 5 minutos, sin embargo buscando el comando -p necesario para automatizar yosys me demoré 1 hora.

Estudiar la información para decidir qué hacer: 10 minutos

Ejecutar lo que decidió hacer: Del Icarus Verilog y GTKwave tarde 2 minutos y Yosys duré 10 minutos.

1.4. Quinta parte:

Buscar información: Duré 5 minutos encontrando información del comando sed

Estudiar la información para decidir qué hacer: 7 minutos entendiendo el comando

Ejecutar lo que decidió hacer: Ejecutándolo duré 5 minutos

1.5. Sexta parte:

Buscar información: No duré nada debido a que me basé en el video de mediación virtual

Estudiar la información para decidir qué hacer: 7 minutos, mientras iba viendo el video iba entendiendo

Ejecutar lo que decidió hacer: 10 minutos implementando lo del video

1.6. Total:

Buscar información: 1 hora con 28 minutos

Estudiar la información para decidir qué hacer: 43 minutos

Ejecutar lo que decidió hacer: 41 minutos

Confeccionar el reporte: 1 hora

2. Parte 2: Instalación

Se instaló GTKwave utilizando el código `sudo apt-get install -y gtkwave`,¹ en cuanto a yosys fue `sudo apt-get install -y yosys`.

3. Parte 3: Descripción y funcionamiento

3.1. Icarus Verilog:

Este programa corresponde a un compilador desarrollado para Linux de lenguaje Verilog. Este se obtiene mediante la licencia GNU GPL. Para el funcionamiento se probó el archivo BancoPrueba.v. primeramente se compila y se crea un archivo mediante el código `iverilog -o tarea1 BancoPrueba.v`. Posteriormente se ejecuta la simulación mediante el comando `vvp tarea1`. La simulación aparece en la figura 1.¹

```

10 1 z z 0 1 1
12 0 z z 0 1 1
14 1 z z 1 0 0
16 0 z z 1 0 0
18 1 z z 1 0 1
20 0 z z 1 0 1
22 1 z z 1 1 0
24 0 z z 1 1 0
26 1 z z 1 1 1
28 0 z z 1 1 1
30 1 z z 0 0 0
32 0 z z 0 0 0
34 1 z z 0 0 0
alberto@alberto-VirtualBox:~/Descargas/Alarma_ejemplo_1$ iverilog -o tarea1 BancoPrueba.v
alberto@alberto-VirtualBox:~/Descargas/Alarma_ejemplo_1$ vvp tarea1
VCD info: dumpfile alarma.vcd opened for output.
      clk,      sAlr_estr,      sAlr_cond,      sLuz,      sPrta,      sIgn
0 0 0 x 0 0 0 0 0
1 0 0 0 0 0 0 0
2 1 0 0 0 0 0 1
4 0 0 0 0 0 0 1
6 1 0 0 0 0 1 0
8 0 0 0 0 0 1 0
10 1 0 0 0 0 1 1
12 0 0 0 0 0 1 1
14 1 0 0 0 1 0 0
16 0 0 0 0 1 0 0
18 1 0 0 0 1 0 1
20 0 0 0 0 1 0 1
22 1 0 0 1 1 1 0
24 0 1 1 1 1 1 0
26 1 1 0 0 1 1 1
28 0 0 0 0 1 1 1
30 1 0 0 0 0 0 0
32 0 0 0 0 0 0 0
34 1 0 0 0 0 0 0

```

Figura 1: Icarus Verilog

3.2. GTKwave:

Gtkwave es un simulador que ayuda a verificar el funcionamiento de un diseño. Los formatos de salida de los archivos de Verilog son difíciles de comprender, por ende mediante GTKWave permite entender de manera sencilla generando gráficas de tiempo para cada señal. Para mostrar lo que hace esta herramienta se utilizó el comando **gtkwave alarma.vcd**, note que se probó con el ejemplo alarma. Posteriormente se seleccionan las señales deseadas y se puede ver como la figura 2.¹

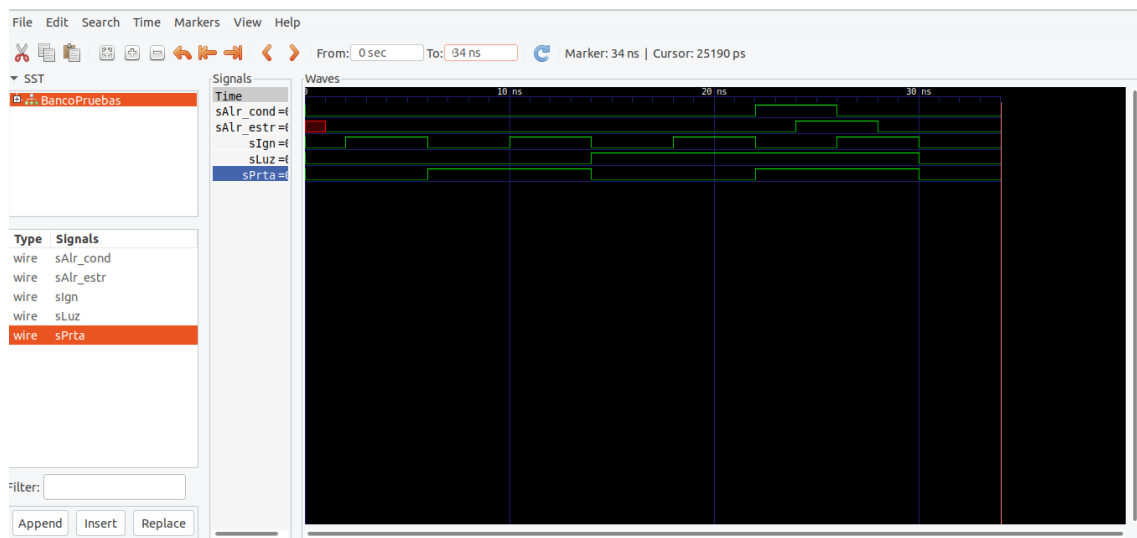


Figura 2: GTKwave

3.3. Yosys:

Yosys es un software compatible con Verilog, el cual posee varios algoritmos básicos de síntesis para este mismo. Este es gratuito y se puede adaptar a cualquier trabajo de síntesis, ya sea con algoritmos existentes, con scripts de síntesis e incluso se pueden agregar nuevos algoritmos para extender la base de código. Para ejecutar el programa primeramente se pone **yosys** para abrir el programa en terminal, seguidamente **read_verilog alarma_desc_conductual.v** y por último **show**. Una vez completado aparece un diagrama como el de la figura 3.²

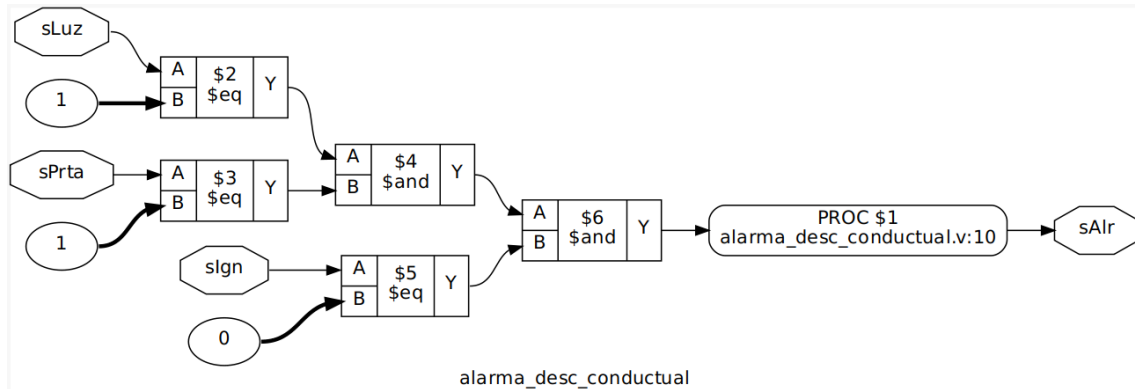


Figura 3: Yosys

4. Parte 4: Makefile

Para la automatización se crea el siguiente makefile(incluyendo parte 5):³

```

1 iverilog1:
2     iverilog -o tarea1 BancoPrueba.v
3
4 iverilog2:
5     vvp tarea1
6
7 iverilog: iverilog1 iverilog2
8
9
10 yosys:
11     yosys -p "read_verilog alarma_desc_conductual.v" -p "hierarchy -check
12         -top alarma_desc_conductual" -p show
13
14 gtkwave:
15     gtkwave alarma.vcd
16
17 sed:
18     sed -i 's/alarma_desc_conductual/alarma_desc_conductualsynth/'
19     copiaalarma_desc_conductual.v

```

5. Parte 5: uso de Sed

Primeramente se hace una copia del archivo `alarma_desc_conductual.v` mediante `cp` `alarma_desc_conductual.v` `copiaalarma_desc_conductual.v`. Posteriormente se le agrega el término `synth` al modulo de la copia mediante el código `sed -i 's/alarma_desc_conductual/ alarma_desc_conductualsynth/' copiaalarma_desc_conductual.v`, además se agrega este código al Makefile. Esto se puede ver en la figura 3.⁴

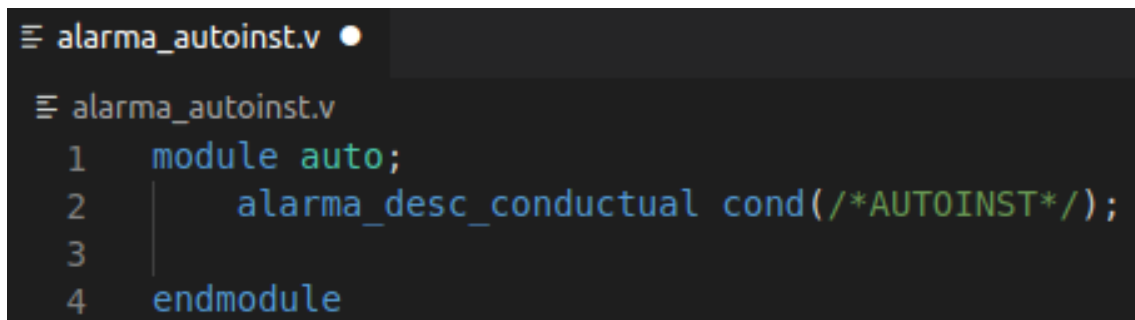
```
alberto@alberto-VirtualBox:~/Descargas/Alarma_ejemplo_1$ cp alarma_desc_conductual.v copiaalarma_desc_conductual.v
alberto@alberto-VirtualBox:~/Descargas/Alarma_ejemplo_1$ cat copiaalarma_desc_conductual.v
module alarma_desc_conductual (
    output reg    sAlr,    // sAlr de tipo reg, almacena el valor
    input         sLuz,    // No se indica el tipo de sLuz, wire implícito
    input         sPrta,   // No se indica el tipo de sPrta, wire implícito
    input         sIgn);   // No se indica el tipo de sIgn, wire implícito
    // En las descripciones conductuales, los puertos de entrada son wires.
    // Los puertos de salida pueden ser wires o regs, dependiendo de la
    // implementación.

    always @ (*) begin // always combinacional, bloque de procedimiento/comportamiento
        // (*) Lista de sensibilidad, entra al "always" ante cualquier cambio
        // en (sLuz or sPrta or sIgn)
        if (sLuz == 1 & sPrta == 1 & sIgn == 0)
            sAlr = 1;    // Asignación bloqueante (=)
        else
            sAlr = 0;    // Asignación bloqueante (=)
        end
    end
endmodule
alberto@alberto-VirtualBox:~/Descargas/Alarma_ejemplo_1$ make sed
sed -i 's/alarma_desc_conductual/ alarma_desc_conductualsynth/' copiaalarma_desc_conductual.v
alberto@alberto-VirtualBox:~/Descargas/Alarma_ejemplo_1$ cat copiaalarma_desc_conductual.v
module alarma_desc_conductualsynth (
    output reg    sAlr,    // sAlr de tipo reg, almacena el valor
    input         sLuz,    // No se indica el tipo de sLuz, wire implícito
    input         sPrta,   // No se indica el tipo de sPrta, wire implícito
    input         sIgn);   // No se indica el tipo de sIgn, wire implícito
    // En las descripciones conductuales, los puertos de entrada son wires.
    // Los puertos de salida pueden ser wires o regs, dependiendo de la
    // implementación.
```

Figura 4: Makefile sed

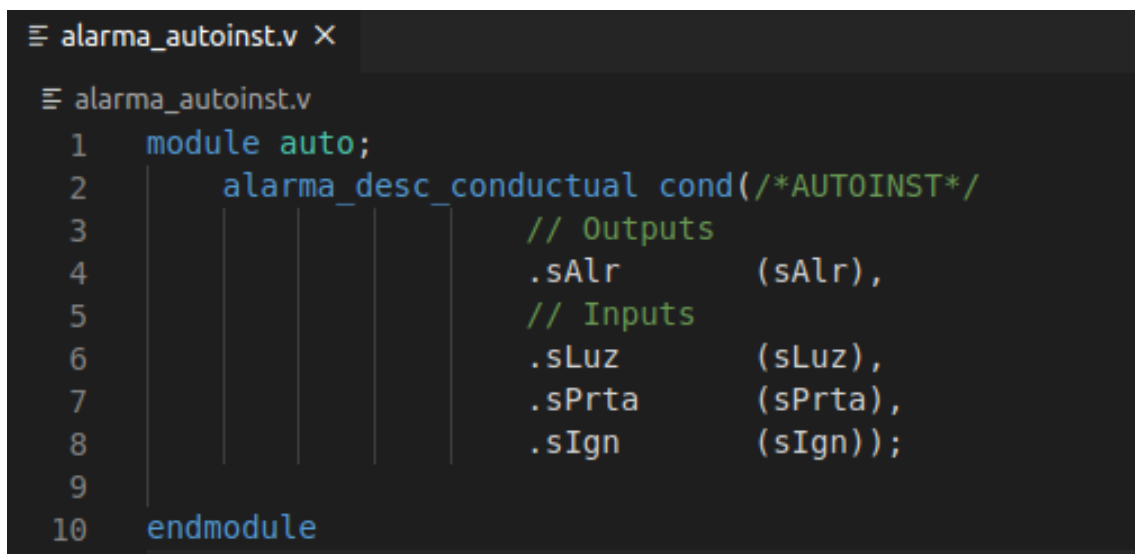
6. Parte 6: uso de Autoinst

Primeramente se descarga emacs mediante el código `sudo apt-get install emacs`. Posteriormente se crea un nuevo archivo llamado `alarma_autoinst.v` y a este archivo se instancia con el archivo `alarma_desc_conductual.v`. Posteriormente en `alarma_autoinst.v` se coloca el código `alarma_autoinst.v` `alarma_autoinst.v` (`/*AUTOINST*/`); en el modulo `auto`, de modo que el archivo quede como en la figura 5. Ahora se coloca en la terminal `emacs -batch alarma_autoinst.v -f verilog_batch-auto`. Entonces cuando se vuelve a abrir el archivo, este queda conectado de manera automática como se ve en la figura 6.⁵



```
≡ alarma_autoinst.v ●
≡ alarma_autoinst.v
1  module auto;
2      alarma_desc_conductual cond(/*AUTOINST*/);
3
4  endmodule
```

Figura 5: Archivo antes de autoinst



```
≡ alarma_autoinst.v X
≡ alarma_autoinst.v
1  module auto;
2      alarma_desc_conductual cond(/*AUTOINST*/
3      // Outputs
4      .sAlr      (sAlr),
5      // Inputs
6      .sLuz      (sLuz),
7      .sPrta     (sPrta),
8      .sIgn      (sIgn));
9
10 endmodule
```

Figura 6: Archivo después de autoinst

Referencias

- [1] Linuxito, “Primeros pasos con icarus verilog en windows.” <https://www.linuxito.com/windows/1088-primeros-pasos-con-icarus-verilog-en-windows>, 08 2014.
- [2] D. Naranjo, “Yosys: un marco open source para herramientas de síntesis verilog.” <https://www.linuxadictos.com/yosys-un-marco-open-source-para-herramientas-de-sintesis-verilog.html>, 01 2020.
- [3] E. Hernandis, “Como hacer un makefile.” <https://hernandis.me/2017/03/20/como-hacer-un-makefile.html>, 03 2020.
- [4] D. . Bits, “Uso del comando sed en linux y unix con ejemplos.” <https://www.ochobitshacenunbyte.com/2019/05/28/uso-del-comando-sed-en-linux-y-unix-con-ejemplos/>, 04 2019.
- [5] James. Video disponible en mediación virtual de autoinst, 08 2020.