

Universidad de Costa Rica

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

IE0523 – Circuitos Digitales II

I ciclo 2024

Reporte Tarea 1

Calculadora de 8 bits con Makefile, y utilizando Yosys.

Alexa López Marcos B94353

Profesor: Ana Eugenia Sánchez Villalobos.

12 de Mayo del 2024

1. Ejecución

Para realizar una correcta ejecución del programa se facilita el siguiente archivo tipo makefile. Para poder ejecutar este archivo desde la terminal de Linux se debe de :

- Abrir la terminal de Linux.
- Posicionarse en la carpeta en la que se encuentra el archivo makefile con los archivos de los módulos del programa.
- Colocar en la terminal **make Calculadora**

Este archivo tipo makefile se utilizara para automatizar la ejecución del programa y contiene las líneas necesarias para ejecutar el programa desde la terminal, se colocan dentro de una bandera del archivo tipo makefile, para que al ejecutar en la terminal se generen los archivos necesarios para ejecutar y abrir GTKWave para ver el comportamiento de las señales.

Dicho makefile contiene:

```
1 Calculadora:
2     iverilog -o tb1.vvp testbench.v
3     vvp tb1.vvp
4     gtkwave tb1.vcd
5     rm -rf tb1.vvp tb1.vcd
6 Calculadora_yosys:
7     iverilog -o tb.vvp testbench_yosys.v
8     vvp tb.vvp
9     gtkwave tb.vcd
10    rm -rf tb.vvp tb.vcd
11
12 clean:
13     rm -rf tb.vvp tb.vcd
14     rm -rf tb1.vvp tb1.vcd
```

Con esto, ya no es necesario ejecutar una a una las líneas necesarias para ejecutar el programa, solamente se ejecutaría **make Calculadora** desde la terminal para ejecutar el programa original y **make Calculadora_yosys** para ejecutar el programa con el código generado por Yosys.

2. Análisis

El programa está compuesto de 3 módulos distintos: **DUT**, **TESTER** y **TESTBENCH**.

2.1. DUT

El archivo llamado **Calculadora.v** corresponde al módulo DUT, en el cual se especifican las condiciones que deben de cumplir las entradas para ejecutar las salidas.

2.1.1. Entradas:

En este caso, este módulo contiene como entradas las señales:

- Sin restricción de bits: **clk** (clock), **rst** (reset) y **en** (enable).
- Del tamaño de un bit: **MODO**
- Del tamaño de 8 bits: **a** y **b**

2.1.2. Salidas:

En este caso, este módulo contiene como salida la señal **c** de un tamaño de 8 bits, la cual es el resultado de la operación realizada por la calculadora.

2.2. Registros:

En este caso, este módulo no utiliza registros.

2.3. Parámetros:

En este caso, este módulo posee como parámetros:

- `suma = 2'b00`
- `resta = 2'b01`
- `multiplica = 2'b10`
- `left_shift = 2'b11`

2.4. TESTER

El archivo llamado **tester.v** corresponde a módulo TESTER, el cual se utiliza para definir el comportamiento de las entradas del circuito, en este módulo en específico las entradas y salidas se definen inversas a como se encuentran definidas en el DUT.

Para comprobar el correcto funcionamiento de los modos se hicieron las siguientes pruebas.

Primera parte de pruebas

- **suma (a + b):** con
 - `a = 8'b10010001` (145 expresado en 8 bits)
 - `b = 8'b00101010` (42 expresado en 8 bits)

dando como resultado $c = 8'b10111011$ (187 en decimal)

■ **resta (a - b):** con

- $a = 8'b01111111$ (127 expresado en 8 bits)
- $b = 8'b00111111$ (63 expresado en 8 bits)

dando como resultado $c = 8'b1000000$ (64 en decimal)

■ **multiplica (a * b):** con

- $a = 8'b00111111$ (63 expresado en 8 bits)
- $b = 8'b00000100$ (4 expresado en 8 bits)

dando como resultado $c = 8'b11111100$ (252 en decimal)

■ **left_shift (a<<1):** con

- $a = 8'b00000110$ (6 expresado en 8 bits)
- $b = 8'b00000000$ (0 expresado en 8 bits)

dando como resultado $c = 8'b00001100$ (12 en decimal)

Segunda parte de pruebas

■ **suma (a + b):** con

- $a = 8'b01111111$ (127 expresado en 8 bits)
- $b = 8'b00111111$ (63 expresado en 8 bits)

dando como resultado $c = 8'b10111110$ (190 en decimal)

■ **resta (a - b):** con

- $a = 8'b10010001$ (145 expresado en 8 bits)
- $b = 8'b00101010$ (42 expresado en 8 bits)

dando como resultado $c = 8'b1100111$ (103 en decimal)

■ **multiplica (a * b):** con

- $a = 8'b00111101$ (61 expresado en 8 bits)
- $b = 8'b00000010$ (2 expresado en 8 bits)

dando como resultado $c = 8'b1111010$ (122 en decimal)

■ **left_shift (a<<1):** con

- $a = 8'b00100110$ (38 expresado en 8 bits)
- $b = 8'b00000000$ (0 expresado en 8 bits)

dando como resultado $c = 8'b1001100$ (76 en decimal)

2.5. TESTBENCH

El archivo llamado **testbech.v** corresponde al módulo TESTBENCH, en el que se importan los módulos DUT y TESTER para poder realizar las conexiones por medio de cables de las salidas del DUT con las entradas del TESTER y las salidas del TESTER con las entradas el DUT, esto hace que se cierre el circuito y se pueda realizar la simulación.

3. Resultados

Como parte de los resultados, se va a mostrar el comportamiento para cada una de las partes simuladas, con el uso de la herramienta GTKWave obtenemos las siguientes figuras:

3.1. Simulación Completa (Binario)

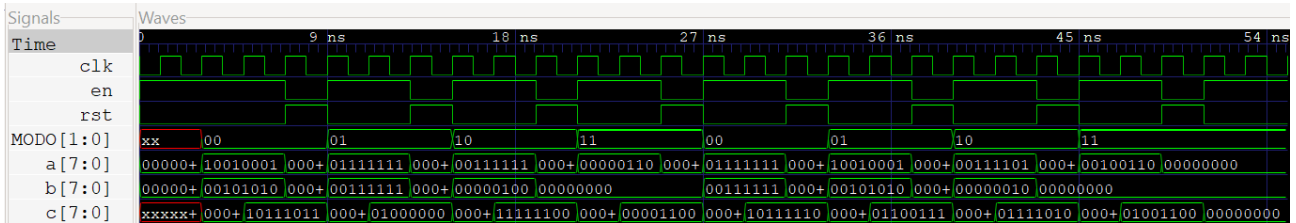


Figura 1: Imagen de la simulación completa realizada en GTKWave para la simulación completa (mostrando resultados en binario).

3.2. Simulación Completa (Decimal)

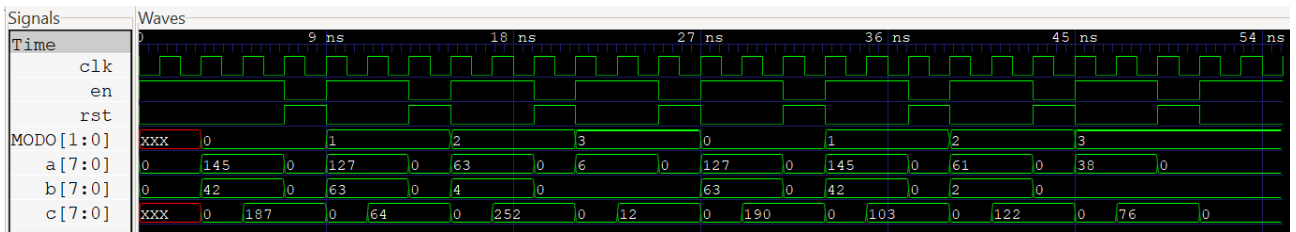


Figura 2: Imagen de la simulación completa realizada en GTKWave para la simulación completa (mostrando resultados en decimal).

3.3. Simulación Completa Yosys (Binario)

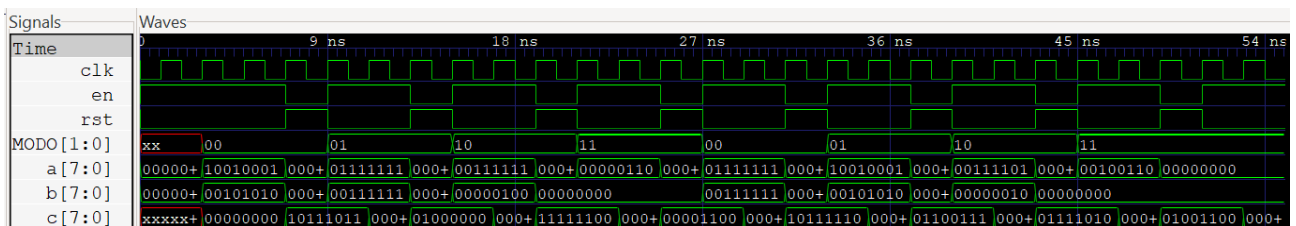


Figura 3: Imagen de la simulación completa realizada en GTKWave con el código generado por Yosys para la simulación completa (mostrando resultados en binario).

3.4. Simulación Completa Yosys (Decimal)

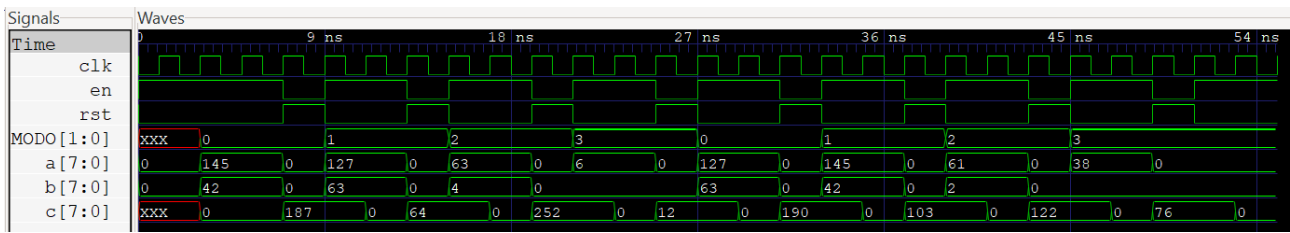


Figura 4: Imagen de la simulación completa realizada en GTKWave con el código generado por Yosys para la simulación completa (mostrando resultados en decimal).

3.5. Comparación de simulaciones

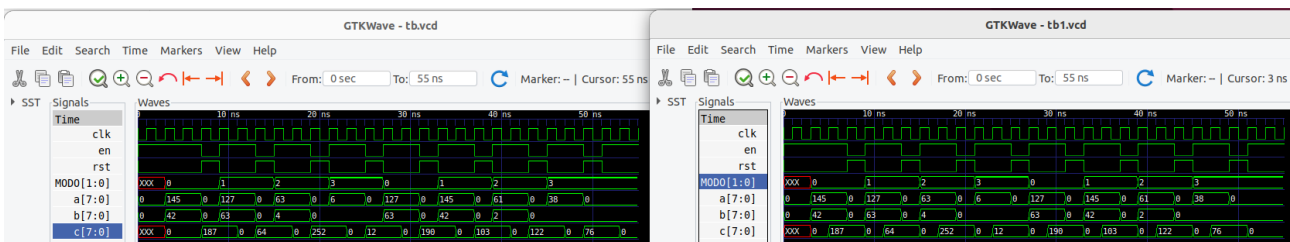
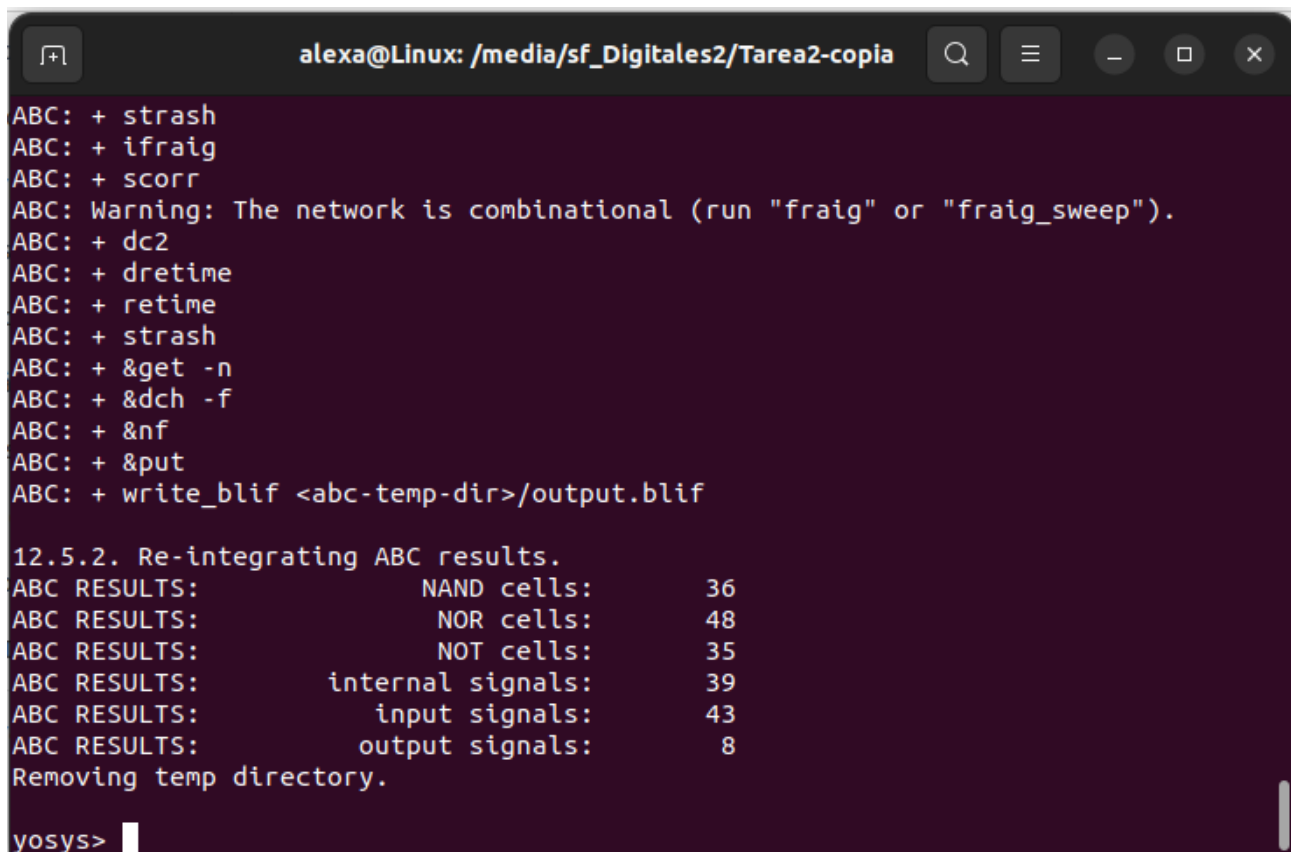


Figura 5: Imagen de comparación de las simulaciones realizadas con el código original y el código realizado por Yosys.

3.6. Cantidad de compuertas con Yosys



```
alex@Linux: /media/sf_Digitales2/Tarea2-copia
ABC: + strash
ABC: + ifraig
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + dc2
ABC: + dretime
ABC: + retime
ABC: + strash
ABC: + &get -n
ABC: + &dch -f
ABC: + &nf
ABC: + &put
ABC: + write_blif <abc-temp-dir>/output.blif

12.5.2. Re-integrating ABC results.
ABC RESULTS:      NAND cells:      36
ABC RESULTS:      NOR cells:       48
ABC RESULTS:      NOT cells:       35
ABC RESULTS:      internal signals: 39
ABC RESULTS:      input signals:   43
ABC RESULTS:      output signals:   8
Removing temp directory.

yosys>
```

Figura 6: Imagen de la terminal en que se muestran la cantidad y el tipo de compuertas utilizadas.

4. Conclusiones

Ya que, al ejecutar **make Calculadora_yosys** se obtiene el GTKwave del código generado por Yosys mostrado en la figura 3 y al ejecutar **make Calculadora** se obtiene el GTKwave generado por el DUT original mostrado en la figura 1. Al comparar estos 2 GTKwave como se muestra en la figura 5 se observa que ambos GTKwave se comportan de la misma forma, lo que demuestra que el código original es equivalente al código generado por Yosys.

5. Anexos

```
yosys> select Calculadora

yosys*> show

14. Generating Graphviz representation of design
Writing dot description to `/home/alexa/.yosys_s
Dumping module Calculadora to page 1.
Exec: { test -f '/home/alexa/.yosys_show.dot.pid
s_show.dot.pid'; } || ( echo $$ >&3; exec xdot '
3> '/home/alexa/.yosys_show.dot.pid' &

yosys*> show -format pdf -prefix ImagenYosys

15. Generating Graphviz representation of design
Writing dot description to `ImagenYosys.dot'.
Dumping module Calculadora to page 1.
Exec: dot -Tpdf 'ImagenYosys.dot' > 'ImagenYosys
.new' 'ImagenYosys.pdf'

yosys*> █
```

Imagen de la terminal en que se muestran los comandos para obtener la imagen generada por Yosys.