Verificación de circuitos digitales con software libre

Eduardo Vázquez Díaz
lalohao@gmail.com

Contenido

***Resumen*—Se creó una maquina virtual con *Ubuntu Desktop 16.04.1* LTS en un contenedor utilizando el software de virtualizacion `Qemu`, donde posteriormente se instaló `verilator` y `gtkwave`; a partir de este sistema se exponen algunas técnicas para simular circuitos con verilog/C++, además de visualizar las ondas generadas de manera gráfica.**

I. Introducción

La importancia de probar los circuitos antes de ser llevados al silicio puede representar millones de dolares, sin contar el tiempo invertido en el diseño, y el que se necesitará volver a invertir para arreglarlo.

En 1990 el lenguaje de descripción de hardware mas usado era VHDL, a pesar de que solo tenia constructores básicos para probar (TestBench) los circuitos . Los diseños empezaban a crecer y nuevo software comercial se creaba para compensar la falta de funciones. Algunas empresas invertían horas de trabajo para crear su propio sistema y no pagar los miles de dolares en licencias, una de ellas llevó a la creación de Accelera que fue la base de SystemVerilog [4].

De la misma manera surgió `Verilator`, un simulador potente de Verilog HDL que además es software libre, este compila el código y lo optimiza para ser simulado rápidamente [1], en algunos casos es incluso mas veloz que los simuladores comerciales [2].

I-A. Virtualización

La maquina virtual permite encapsular el sistema de verificación de circuitos en un contenedor que no será afectado (y que no afectará) la maquina utilizada, esto elimina errores que podrian ser causados al tener instalado software que utilice configuraciones con variables globales (PATHS) como ocurre con `HSPICE` y `Questa SIM` por ejemplo.

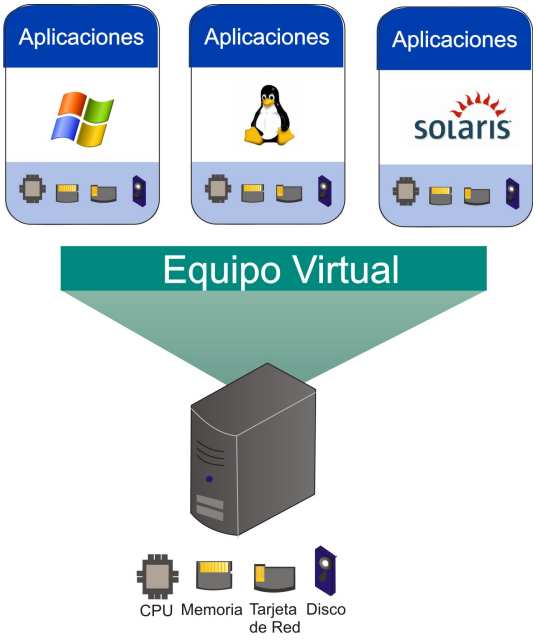

Figura 1. Las maquinas virtuales pueden o no conectarse entre ellas o hacia la red externa y pueden ser de diferentes arquitecturas y sistemas operativos independientemente del sistema anfitrion.

Se le dice anfitrión a la maquina donde se encuentran los contenedores virtuales, en este caso la anfitriona usa *Arch Linux*, pero esto no afecta a los contenedores ya que estan aislados, por lo que esto se puede aplicar de igual manera en Windows o Mac.

II. Desarrollo

Para instalar la maquina virtual se utiliza la linea de comandos, creando el disco duro virtual llamado **verif**, con un tamaño de 10Gb, posteriormente se carga el archivo `ISO` de Ubuntu en el disco duro creado, con 2G de memoria RAM y se procede a instalar el sistema operativo de forma normal.

```
qemu-img create -f raw verif 10G
qemu-system-x86_64 -cdrom ubuntu.iso \
                   --boot order=d -drive \
                   file=verif,format=raw \
                   -m 2G
```

Una vez instalado el sistema operativo se puede iniciar de la siguiente forma:

```
qemu-system-x86_64 --boot order=d \
   -drive file=verif,format=raw -m 2G
```

Posteriormente en la maquina virtual (Ubuntu) se instala el software necesario para simular [3], al igual que el editor de texto de su preferencia para modificar los archivos :

```
sudo apt-get install git make autoconf \
     g++ flex bison verilator gtkwave
```

REFERENCIAS

[1] https://www.veripool.org/wiki/verilator Last accessed Sat Jan 28 12:01:25 2017.

[2] https://www.veripool.org/wiki/veripool/Verilog_Simulator_Benchmarks Last accessed Wed Feb 1 20:41:50 2017.

[3] https://www.veripool.org/projects/verilator/wiki/Installing Last accessed Sat Jan 28 12:25:07 2017.

[4] Chris Spear. *SystemVerilog for verification a guide to learning the testbench language features*. Springer, 2008.